Semantic-Aware Virtual Reality Video Streaming

Yue Leng*
UIUC
yueleng2@illinois.edu

Chi-Chun Chen*
University of Rochester
cchen120@ur.rochester.edu

Qiuyue Sun
University of Rochester
qsun15@u.rochester.edu

Jian Huang
UIUC
jianh@illinois.edu

Yuhao Zhu
University of Rochester
yzhu@rochester.edu

ABSTRACT

Virtual reality (VR) technologies have huge potential to enable radically new applications, among which spherical panoramic (a.k.a., 360°) video streaming is on the verge of hitting the critical mass. Current VR systems treat 360° VR content as plain RGB pixels, similar to conventional planar frames, resulting in significant waste in data transfer and client-side processing. In this paper, we make the case that next-generation VR platforms can take advantage of semantics information inherent to VR content so as to improve the streaming and processing efficiency. To that end, we present SVR, a semantic-aware VR system that utilizes the object information in VR frames for content indexing and streaming. SVR exploits the key observation that end-users' viewing behaviors tend to be object-oriented. Instead of streaming entire frames, SVR delivers miniature frames that cover only the tracked visual objects in VR videos. We implement SVR prototype with a real hardware board and demonstrate that it achieves up to 34% network bandwidth reduction along with 21% device power saving.

1 INTRODUCTION

Virtual Reality (VR) has been generating profound social impact in transformative ways. For instance, immersive VR experience is shown to reduce patient pain [6] more effectively than traditional medical treatments and is seen as a promising alternative to addictive painkillers like opioids [9].

Among all VR use cases, a particularly promising one is 360° VR video streaming. 360° videos present a spherical panoramic view of the scene and are usually viewed through dedicated VR devices such as a head-mounted display (HMD). As users change the viewing orientation, the display will render different parts of the scene, creating an immersive experience for users.

The most salient characteristic of VR video streaming is that users' current viewable area is only a small portion of a full spherical frame [19]. The viewable area is characterized by the Field-of-View (FOV), which is an intrinsic parameter of a VR HMD that captures the degrees of horizontal and vertical viewing angles provided by the HMD. For instance, under a typical 120° × 90° FOV, the viewable area is one-sixth of the full spherical frame.

Off-the-shelf VR delivery systems, e.g., YouTube and Facebook, inherit the techniques used in delivering conventional planar videos. Specifically, they always fetch full frames, both areas inside and outside of the FOV. Although this strategy accelerates the deployment of VR videos by reusing the existing video delivery infrastructure, it leads to two major system-level inefficiencies.

First, transferring VR videos from the cloud to client devices imposes massive bandwidth requirement. For instance, streaming VR videos for a perceived 720p resolution on a VR device with a 120° × 90° FOV requires a network bandwidth that is 6× higher than that of streaming a conventional planar video under the same perceived resolution. The bandwidth requirement will only grow as users demand better user experience.

Second, rendering 360° videos on VR devices in real-time consumes excessive power, presenting a practical challenge to the energy- and thermal-constrained mobile VR devices [24]. Our measurements show that rendering 720p VR videos in 30 frames per second (FPS) consumes over 4 W power, which is twice as much power than rendering conventional planar videos and exceeds the Thermal Design Point (TDP) of typical mobile devices [24].

Recent VR video delivery research [21, 26, 33] mostly focuses on reducing the network bandwidth using view-dependent streaming. For instance, VisualCloud [26] decomposes each frame into multiple tiles and transmits tiles that are predicted to be within a user's FOV in high resolution and transmits other tiles in lower resolution. However, view-dependent streaming increases the device power need since tiles from the same frame have to be stitched together before rendering, further exacerbating the power issue.

This paper proposes a comprehensive VR delivery system design that simultaneously reduces the network bandwidth requirement and power consumption on VR device. We achieve so by transferring and processing only the necessary pixels that fall within a user's FOV. Our major insight is that we can provide accurate prediction of user FOVs by *leveraging semantics information inherent in VR content that is previously ignored.* As a first-attempt toward semantics-aware VR streaming, we focus on one particular form of semantics information: visual objects. Based on real user data study, we find that VR users tend to focus on visual objects in their FOVs, and track the same set of objects across frames. Therefore, user FOVs mostly align with object trajectories within a VR video.

Harnessing this insight, we propose a semantic-aware VR (SVR) system that extracts visual object information in VR content to assist VR video streaming. Leveraging recent advances in computer vision, the cloud component of SVR extracts object semantics from VR videos upon ingestion. Object information is stored as metadata associated with the original VR video. SVR leverages the object information to create a miniature video that contains only the user FOVs as well as to perform essential processing that originally has to be executed on the VR client. Upon client request, the cloud service streams the miniature video with object semantics. In this way, SVR minimizes data volume while simplifying client-side processing.

*Yue Leng and Chi-Chun Chen are co-primary authors.

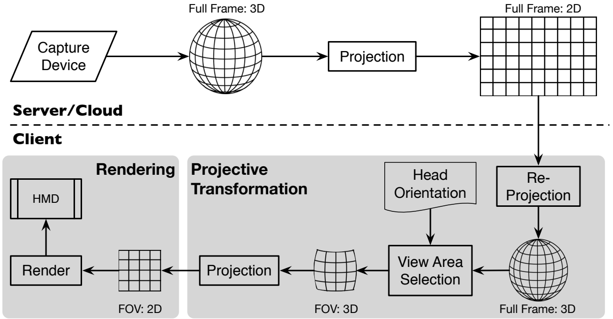


Fig. 1: **Illustration of today's VR video delivery system.**

We implement our design in a prototyped system, where the cloud service is hosted on an AWS instance and the client is deployed on an Nvidia TX2 development board to emulate real VR end users. Evaluating on a set of common VR videos based on real system measurement, SVR reduces the network bandwidth requirement by up to 34% while saving client-side power consumption by up to 21%. Overall, this paper makes the following contributions:

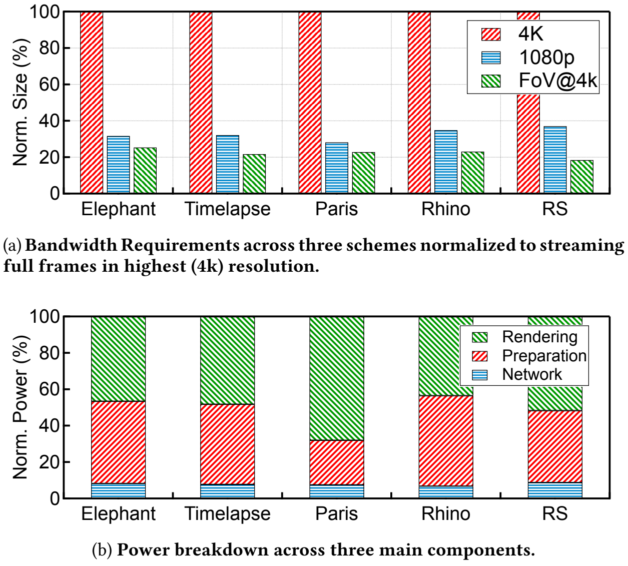
- We provide a comprehensive power analysis of VR devices. We show that frame projective transformation is a significant power overhead that is uniquely introduced by VR videos.
- We quantitatively demonstrate, for the first time, that VR users' viewing behavior strongly correlates with visual object movements in VR content based on real-user studies.
- We propose a semantics-aware VR (SVR) system that extracts and persists visual object semantics from VR content, and leverage the object semantics for efficient content indexing and streaming.
- We implement SVR on a real-world cloud-client setup and demonstrate significant reductions in network bandwidth requirement and mobile device power consumption.

2 VR VIDEO STREAMING BACKGROUND

Different from conventional planar video content, 360° VR content provides an immersive experience by encoding spherical panoramic views in videos frames. With more information provided in each frame, users can choose different focus point on the spherical panoramic surface, introducing a diversified viewing experience. Spherical frames are created by special capturing systems such as an omnidirectional camera [15, 27] or a multi-camera rig [3], and then delivered to a user's HMD for playback. This paper primarily focuses on the VR delivery and playback system, i.e., after a VR video is captured. This section introduces the necessary background.

Figure 1 shows a today's off-the-shelf VR video delivery system. When spherical frames are uploaded to a VR content provider such as YouTube or Facebook, spherical frames are projected to the conventional planar format using one of the various projection formats such as equirectangular projection [21], and thus form a conventional planar video. The spherical-to-planar projection is performed so as to leverage the well-established planar video compression and streaming algorithms. Upon user requests, the (projected) planar video is streamed to the client for display.

Once on the client device and decoded by the codec, each planar frame goes through a sequence of projective transformations before it is eventually rendered on the HMD. A planar frame is first re-projected back from the planar format to the spherical format, which is then used to produce the current viewing area, a.k.a., FOV, based on the head orientation sensor data. The FOV area then goes through another projection to the planar format that can then be can be viewed from user's perspective [2]. In modern VR client software, both the projective transformation and rendering are executed using a combination of CPU and GPU.



(a) **Bandwidth Requirements across three schemes normalized to streaming full frames in highest (4k) resolution.**

(b) **Power breakdown across three main components.**

Fig. 2: **Network bandwidth and power consumption bottlenecks in today's off-the-shelf VR solutions.**

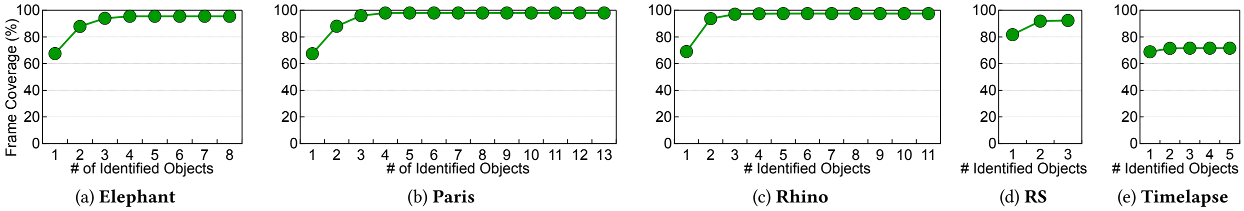
The design philosophy of today's VR delivery system is to reuse existing planar video streaming infrastructure as much as possible so as to accelerate the adoption of VR videos. For instance, although the FOV occupies only a small portion of a full frame, today's VR delivery systems always stream the entire video and rely on endpoint devices to project the frame and extract FOV to minimize disruptions to existing server infrastructure. As we show later, this strategy introduces tremendous inefficiencies, putting pressure on devices with limited resources.

3 BOTTLENECKS AND OPPORTUNITIES

This section identifies and quantifies two main bottlenecks in VR video delivery: network bandwidth and device power consumption. Using a large-scale user study (§ 3.1), we show that existing solutions waste network bandwidth (§ 3.2) and device processing power (§ 3.3) by transmitting and processing unnecessary pixels. We then show that exploiting VR video semantics, especially visual objects, offers an opportunity to mitigate both bottlenecks (§ 3.4).

3.1 Methodology

We conduct studies on a recently published VR video dataset [19], which consists of head movement traces from 59 real users viewing five different 360° VR videos. The dataset records the real-time head movements, through which we obtain each user's FOV in every



(a) **Elephant** (b) **Paris** (c) **Rhino** (d) **RS** (e) **Timelapse**

Fig. 3: **Measurement study on a real-world VR video dataset consisted of 59 users viewing five different 360° videos. Each $< x, y >$ point represents the percentage of frames (y) in which at least one of the x identified/detected objects appears in users' FOV. Results indicate that users' attention centers around visual objects in VR content.**

frame. The dataset is collected using the Razer Open Source Virtual Reality (OSVR) HDK2 HMD with a typical FOV of 110° × 110° [10].

We use the NVIDIA Jetson TX2 board [7] as the client computing platform since it represents the hardware that high-end VR devices possess. We directly measure TX2's power consumption using the Texas Instruments INA 3221 voltage monitor IC on TX2. We use YouTube as a representative off-the-shelf VR delivery system.

3.2 Network Bandwidth Characterizations

Today's off-the-shelf VR delivery systems transmit pixels out of user's FOV, and thus introduce bandwidth inefficiencies. We quantify the network bandwidth requirement using the total amount of data transferred as a proxy. Figure 2a compares the total amount of transferred data between three schemes: (1) streaming full frames under the highest resolution (4K) for the best viewing experience, (2) streaming full frames under a lower-resolution (1080p) to tradeoff bandwidth requirement for user experience, and (3) streaming only the FOV part of each frame, but in the highest resolution (4K).

Lowering the frame resolution leads to about 67% bandwidth reduction, which however comes at a cost of user experience degradation [18]. In contrast, streaming only the FOVs in the highest resolution reduces bandwidth by 78% (110° × 110° FOV is about one-fifth of the full frame size) with no impact on viewing experience.

FINDING 1: *Off-the-shelf VR delivery solutions (e.g., YouTube) transfer redundant pixels that are outside of users' FOV, leading to 78% bandwidth waste. Lowering the resolution improves the bandwidth usage, but still transmits more data that is necessary.*

3.3 Device Power Characterizations

Playing VR videos consumes excessive power on the client-side device, which is particularly problematic for HMDs and smartphones that are energy and thermal constrained. In our measurements, the device consistently draws over 4 W power across all five videos. The power consumptions are obtained when streaming VR videos at a 720p (1280×720) resolution for the FOV (4K, or 3840×2160 for the full frame), which is the minimal requirement for an immersive VR experience. As a comparison, the Thermal Design Point (TDP), i.e., the power that the cooling system is designed to sustainably dissipate, of a mobile device is around 3.5 W [24], clearly indicating the need to reduce power consumption of VR playback.

We breakdown the device power consumption of VR playback into three parts: network, projective transformation (PT), and rendering. Recall from Figure 1 that PT is the step that is unique to VR video playback. It generates planar FOV frames from spherical full frames before the former are rendered on the display. The power breakdown is shown in Figure 2b.

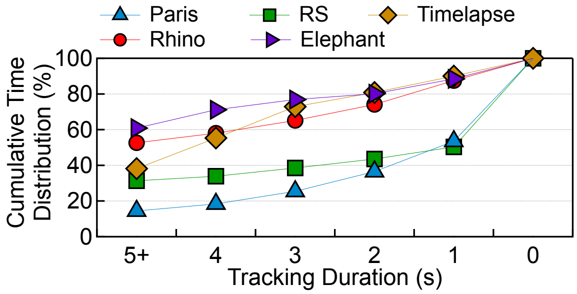


Fig. 4: **Cumulative distribution of tracking time durations.**

We find that rendering and PT are the two major power consumers. Although rendering is a necessary step for video streaming that occurs even for conventional planar videos, the frame projective transformation step is a pure overhead, 40% on average, introduced by VR videos. In particular, the preparation step processes out-of-sight pixels that are not in users' FOVs but are transmitted as an artifact of today's VR delivery system. Removing the PT step and directly rendering the FOV frames would significantly reduce the device power consumption.

FINDING 2: *VR devices consume excessive power. Over 40% of the power consumption is attributed to the frame PT step that processes redundant pixels outside of users' FOV.*

3.4 Object-Oriented Viewing Behaviors

Existing VR solutions are inefficient mainly because they transfer and process redundant pixels outside of FOV. This section motivates the feasibility of transferring and processing *only the absolutely necessary pixels* in VR content with little impact on user experience. Our key insight is that users tend to focus on objects in VR content, and object movement provides a proxy for predicting user FOVs.

We use the state-of-art object detection tool [35] and manually adjust the objects annotation in the VR video dataset to quantify the correlation between users' FOV and visual objects in VR content in Figure 3. We reach two key conclusions.

First, users tend to pay attention to objects in VR videos. Even if only one object is identified in the video, users' FOVs cover that single object in 40% (`Paris` and `Timelapse`) to 60% (`Elephant`, `Rhino`, `RS`) of the frames. As the number of identified objects increase, the

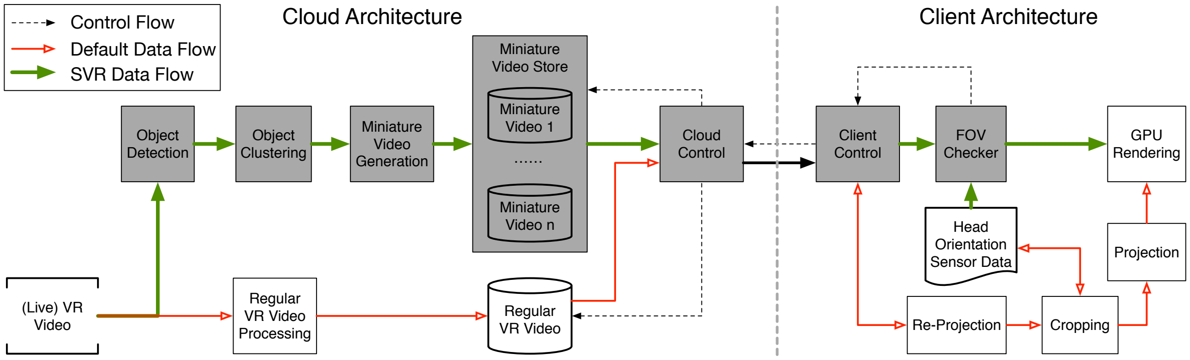


Fig. 5: **SVR overview. Augmentations to existing VR platforms are shaded.**

percentage of frames in which users focus on identified objects increase to at least 80%, and reaches almost 100% in cases of `Elephant` and `Rhino`. This indicates that frame areas that contain a group of visual objects are likely to be watched by end-users, and therefore streaming those frame areas will likely satisfy users' needs.

We further confirm that users track the same set of objects across frame rather than frequently switching objects. Specifically, we measure the time durations during which users keep tracking the movement of the same object, and show the results in Figure 4 as a cumulative distribution plot. Each $< x, y >$ point in the figure denotes the percentage of time (y) during which users track an object for at least a particular time duration (x). On average, users spend about 47% of time tracking an object for at least 5 seconds.

Second, the near 100% FOV coverage in many videos as the number of identified objects increases indicates that the cloud can effectively predict user FOVs solely based on the VR video content (i.e., objects) without sophisticated client-side machine learning models. This observation frees the resource-constrained VR clients from predicting FOVs and simplifies the client design.

FINDING 3: *VR users tend to track object movement. Users' FOVs can thus be predicted and tracked by object trajectories without using specifically-trained head-movement models.*

4 SVR DESIGN

SVR's design objective is to minimize the network bandwidth requirement and device power consumption of VR video delivery with little impact on user-experience. To that end, SVR attempts to transmit and process only the absolutely necessary pixels that are in users' FOV. We first provide an SVR overview by describing its design principles and the various components on both the cloud and client sides (§ 4.1). We then discuss in detail the cloud VR service (§ 4.2) and client-side playback software augmentations (§ 4.3).

4.1 Overview

Today's VR platforms fall short because they treat VR content as plain RGB pixels while largely ignoring the rich semantics information in the video. Video semantics, however, can assist VR content streaming and processing [37]. SVR focuses on one particular form of semantics information: visual object. By leveraging key object information, SVR reduces the data communication volume while simplifying the VR client processing.

The system architecture of SVR is illustrated in Figure 5. The cloud architecture of SVR has two major components: a static and offline analysis component, and a dynamic and runtime serving component. The static component extracts visual objects from the uploaded VR video and generates different miniature videos, each of which corresponds to one trajectory of an object cluster.

Critically, each miniature video frame is already converted from the spherical format to the planar format and contains only the user's FOV. Thus, miniature videos are much smaller in size compared to the original video and can be directly rendered on the client device, bypassing the power-hungry projective transformation step.

The dynamic component, at runtime, streams miniature videos to the client. In cases where the client requests a wrong miniature video initially, which we define as a *FOV-miss*, the dynamic component corrects it by transferring the original full frames, essentially falling back to the normal VR streaming mode. We augment the new miniature video with metadata that corresponds to the head orientation for each frame. The miniature video along with associated metadata explicitly contains essential object semantics of the original video, and thus can ease future object-based processing.

Once the miniature video together with its associated metadata is on the client side and before a miniature frame is rendered to the display, the VR client compares the desired FOV indicated by the head movement motion sensor with the metadata associated with the

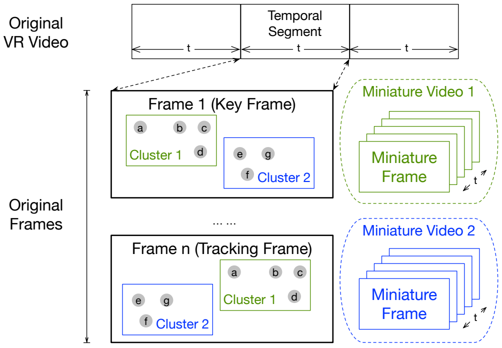


Fig. 6: **Miniature video creation.**

frame. If the two match, i.e., a *FOV-hit*, the client renders the frame on the display. Otherwise, the client system requests the original video segment from the cloud. To ease the penalty for requesting original segments, there might be a resolution adjust period for transmitting full segments and smooth laggy experience. We expect that the former case is far more common, thus significantly reducing the overall bandwidth and client power consumption.

4.2 Cloud Service Architecture

Users tend to track object movement. In addition, they tend to track not one object, but most often a group of objects. This motivates the fundamental idea behind the cloud architecture design: extract object information and group objects into different clusters. Each cluster contains a unique set of objects that users tend to watch together. By tracking the same cluster of objects across frames, we can accurately predict the FOV of end-users.

Miniature Video Figure 6 provides an overview of how a VR video is processed in the cloud by SVR. First, we decompose the video into multiple t-second temporal segments. We categorize the frames of each video segment into two types: *key frame* and *tracking frame*. A key frame is always the first frame of a segment; it is a frame in which objects are explicitly detected and clustered. Objects within the same cluster are then tracked across subsequent tracking frames, effectively creating a trajectory of the object cluster.

SVR creates a miniature frame for each cluster. The miniature frame has a size that matches the end-user's FOV on the client VR device and is converted from the spherical format to the planar format that can be directly rendered on the client device upon a FOV-hit. In the end, the server creates a miniature video from all the miniature frames of the same object cluster. If multiple clusters exist in the original video, multiple miniature videos are created.

Temporal Segmentation The size of the temporal segment determines the granularity of miniature videos. In one extreme design, one could create a miniature video for the entire video. This design would lead to high video compression rate [16], but is less flexible in that any single FOV-miss would lead to re-streaming of the entire original video. In contrast, we divide the original video into many segments, each of which has its own miniature videos. In this way, only one segment of the original video needs to be transmitted upon a FOV-miss.

SVR Store The miniature videos is stored in log-structured manner. We place the associated metadata in a separate log rather than mixing them with frame data. This allows us to decouple metadata with normal video encoding, and thus simplify the system design as well as miniature management. SVR generates the miniature videos statically beforehand to reduce runtime latency, but this incurs storage overhead. An alternative design is to generate miniature videos on-demand when receiving user requests. We believe that sacrificing storage for lower latency is a desirable trade-off as the storage becomes cheaper [5] and more scalable while users demand higher viewing experience. The exact storage overhead is proportional to the number of miniature videos created, which is in turn proportional to the number of object clusters in a frame. As users' view region of interest is usually covered within certain area such as horizontal area [19], we empirically find that choosing a cluster number of three leads to decent savings with reasonable cloud storage overhead and user-view coverage.

Handling Client Requests SVR differentiate between two types of client requests: requests for miniature videos and requests for the original video. The former is made at the beginning of each video segment when the client decides what object cluster the user is most likely interested in, and then SVR will serve the corresponding miniature video from the cloud. The latter request is made when a FOV-miss happens, upon which SVR serves the original segment.

4.3 Client Architecture

On the VR client, for each (miniature) frame that will be rendered, the playback application will check the real-time head pose and compare it against the associated metadata of the frame. If the desired FOV indicated by the current head pose is covered by the corresponding miniature frame (FOV-hit), the miniature frame can be directly rendered onto the display. Otherwise (FOV-miss), the client will request the entire original video segment that contains the correct frame. It might initially seem to be wasteful to stream an entire segment although only the missing frame is needed. However, this strategy is more bandwidth-friendly because video compression ratio is much higher than image compression ratio [16]. It is worth noting that upon a FOV-miss, the current frame that induces the miss is dropped so as to minimize the impact on user experience.

FOV Thresholding An important requirement of the client architecture makes is to reduce the FOV-misses in order to minimize network transmission and local processing power. We propose an optimization that significantly reduces the FOV-miss ratio by leveraging the unique characteristics of human perception systems.

We observe that, although at a given moment the miniature frame may not perfectly overlap with the user's FOV, the "missing" pixels in the non-overlap region are necessarily at the edges of the user's FOV. Numerous studies on human peripheral vision have shown that human vision acuity declines steeply when contents are at the edges of the visual field [14, 17]. Therefore, the missing pixels can be approximated without impacting user experience while avoiding unnecessary FOV-misses. An FOV-miss is generated only when the non-overlap region size is above a certain threshold.

Specifically, the client approximately reconstructs the missing pixels by extrapolating the edge pixels in the transmitted miniature frames. This strategy has the advantage of being computationally efficient. We plan to explore more sophisticated approaches such as reconstruction using motion estimation [28].

5 IMPLEMENTATION DETAILS

We build an end-to-end SVR system which is distributed across a VR cloud server and a playback client. In particular, we choose the NVIDIA Jetson TX2 board [7] as the client computing platform as it represents the hardware that high-end VR devices possess. We compare against the real user head movement trace [19] and faithfully generate the FOV-hit/miss statistics, from which we obtain performance and device power consumption.

Object Extraction The VR server uses convolutional neural network (CNN)-based framework for object detection for its superior accuracy. In our implementation, we choose YOLOv2 [13, 35], which achieves state-of-the-art detection accuracy.

Object Clustering The server uses the classic k-means algorithm [25] for object clustering. Currently, the clustering considers only the object location with the intuition that users tend to watch objects that are close to each other within segments. In future work, we will investigate more advanced clustering techniques that can leverage richer object semantics such as object category.

Temporal Segmentation We statically set the segment length to 20 frames, which roughly match the Group of Pictures (GOP) size in video compression [36] and thus retains reasonable compression ratio. We plan to explore adaptive temporal decomposition methods.

6 PRELIMINARY RESULTS

We evaluate SVR using three metrics: network bandwidth saving, power reduction, and user experience degradation. The baseline we compare against is the YouTube VR platform (as described in Figure 1), which represents today's off-the-shelf VR video delivery systems [21]. Although we are still optimizing SVR design, our preliminary evaluation has shown promising results.

Bandwidth Savings We quantify the bandwidth savings in Figure 7a, which shows the bandwidth saving of SVR compared to the baseline system that always streams full frames.SVR reduces the bandwidth requirement by up to 34% and 28% on average.

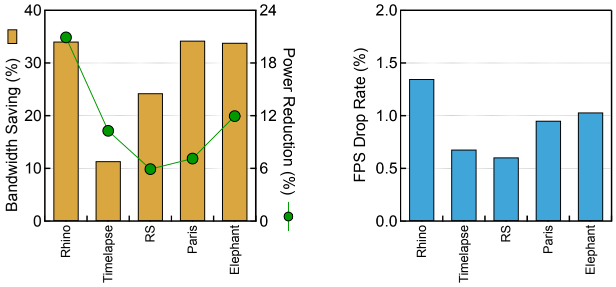
Power Reductions We overlay the average power consumption reduction of the VR client on the right y-axis in Figure 7a. SVR achieves on average 11% and up to 21% power reduction. The power reduction not only increases the VR viewing time, but also reduces the heat dissipation and thus provides a better viewing experience.

User Experience Impact We use FPS drop rate as a first-order proxy to evaluate user experience. Figure 7b shows the frame drop rate averaged across 59 users. Overall, SVR introduces less than 1% FPS drop, indicating negligible loss of user experience. In future work, we plan to perform subjective user study to further confirm our quantitative conclusion.

Sensitivity Study We study SVR's sensitivity to the FOV threshold, which is a client-side parameter that determines when a FOV-miss is generated (§ 4.3). Figure 8a and Figure 8b show how the device power consumption and bandwidth saving varies as the FOV threshold increases from 80% to 100%, respectively. For comparison purposes, we overlay the baseline power consumption in Figure 8a.

Not surprisingly, as FOV threshold increases the power consumption increases and the bandwidth saving decreases as more FOV misses are generated. We note that even with 100% overlap threshold SVR is still able to outperform the baseline. In future work we will investigate advanced mechanisms to control the threshold.

We also compare SVR with VisualCloud [26], which represents the recent view-guided VR systems that use machine learning (ML) techniques to predict user FOVs. According to the results reported in VisualCloud, SVR achieves a similar bandwidth saving. However, SVR's main advantage is that it also reduces the power consumption of client devices dramatically. This is because SVR removes the power-intensive PT step from the critical path (see § 3.3). Unlike the ML-based VR systems that require significant effort on model training and powerful computing resources for intelligent processing, SVR presents a lightweight and power-efficient solution by leveraging the inherent semantic information in VR content. Future work will conduct a more comprehensive comparison to further quantify SVR's advantages over view-guided, ML-based VR systems.



(a) **Client power savings and network bandwidth reductions.**

(b) **FPS drop rate.**

Fig. 7: **Mobile device power saving, bandwidth reduction, and impact on user experience, averaged across 59 users.**

7 RELATED WORK

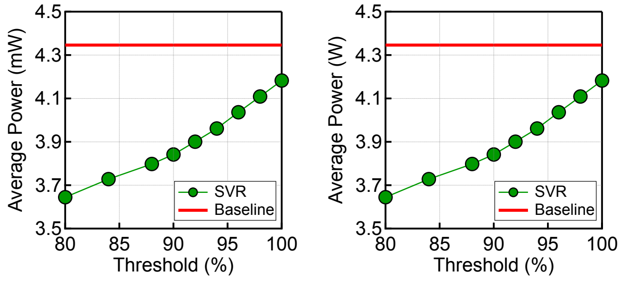
VR Content Delivery Today's VR systems such as YouTube and Facebook treat VR videos the same as ordinary planar videos while being agnostic to end-users' FOVs. To improve bandwidth efficiency, they typically apply conventional planar video streaming optimizations such as Dynamic Adaptive Streaming over HTTP (DASH) [38]. DASH lowers the resolution of an entire frame segment when the network is congested. In our experience of using YouTube VR, resolution is reduced frequently when streaming videos in a 720p FOV resolution (4k for full frames), degrading user experience.

Recently, researchers have started taking into account user FOV and investigating view-guided optimizations for serving VR videos [20, 23]. For instance, Haynes et al. [26] and Zare et al. [39] both propose to predict user head movement and thereby reducing the resolution of out-of-sight areas in a frame. Similarly, Khiem et al. [29, 34] proposes to predicts users' Region-of-Interest (ROI) and streams the ROIs with high resolution. Qian et al. [33], Fan et al. [22], and Liu et al. [31] propose to stream only the FOV tiles.

SVR can also be categorized as an view-guided approach. However, compared to existing FOV-based approaches, SVR has two key advantages. First, most of these schemes still require transferring the whole area of the frames. As a result, the power-hungry PT step is still a necessary step. In contrast, SVR offloads the PT step to the cloud and reduces the device power. Second, they all require training and running specific machine learning models to predict user FOVs. In contrast, SVR leverages semantics inherent in VR content to predict FOVs.

VR Content Representation Projecting from the spherical format to the planar format is a critical VR delivery component that affects network bandwidth efficiency and visual distortions. This paper assumes the most widely used Equirectangular projection. However, the SVR design is independent of the projection, and thus can be readily integrated with recently proposed projection formats such as CubeMap [12], Equi-Angular Cubemap [1], and Pyramid [8].

General VR Optimizations There has also been recent work on improving VR video capturing. Google [4], Facebook [3], and Samsung [11] have all released hardware and software to streamline the VR video capturing process. Recently, Mazumdar et al.,



(a) **Power sensitivity.**

(b) **Bandwidth sensitivity.**

Fig. 8: **Power and bandwidth reduction sensitivity to FOV threshold.**

proposes specialized hardware to accelerate bilateral solver[32], which is a compute bottleneck while capturing VR videos. Konrad et al., proposes a system that natively generates spherical VR videos while bypassing most of the compute-intensive processing in conventional VR capture systems[30]. Orthogonal to the capturing systems, SVR instead, focuses on the streaming and playback stage.

8 CONCLUSION AND PROSPECTIVES

Delivering 360° VR content presents a tall order for future cloud and mobile systems. We posit that the rich object semantics inherent in VR content offer critical information that help build efficient VR systems. This paper presents one such system called SVR, which extracts key object semantics from VR videos, and uses the object information to guide VR content streaming. Our prototype demonstrates significant bandwidth and processing power reduction with little loss of user-experience with regard to user focus and streaming quality.

In the long run, 360° video is just one form of the myriad of visual contents that are being generated and consumed. Computer systems researchers should fundamentally rethink how visual data is organized, managed, and processed. Distilling semantics information from visual data is a particular promising approach. Future developments should examine other forms of visual semantics and look beyond optimizing VR content streaming, but also processing, display, etc. We hope our work serves the first step in a promising new direction of research.

REFERENCES

[1] Bringing pixels front and center in vr video. https://blog.google/products/google-vr/bringing-pixels-front-and-center-vr-video/.

[2] Converting an equirectangular image to a perspective projection. http://paulbourke.net/miscellaneous/sphere2persp/.

[3] Facebook Surround 360. https://facebook360.fb.com/facebook-surround-360/.

[4] Google Jump VR. https://vr.google.com/jump/.

[5] Hard Drive Cost Per Gigabyte. https://www.backblaze.com/blog/hard-drive-cost-per-gigabyte/.

[6] Hospital-wide access to virtual reality alleviates pain and anxiety for pediatric patients. http://www.stanfordchildrens.org/en/about/news/releases/2017/virtual-reality-alleviates-pain-anxiety.

[7] Jetson TX2 Module. http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html.

[8] Next-generation video encoding techniques for 360 video and vr. https://code.facebook.com/posts/1126354007399553/next-generation-video-encoding-techniques-for-360-video-and-vr/.

[9] Opioids haven't solved chronic pain. maybe virtual reality can. https://www.wired.com/story/opioids-havent-solved-chronic-pain-maybe-virtual-reality-can/.

[10] Razer OSVR HDK2. https://versus.com/en/htc-vive-vs-razer-osvr-hdk2.

[11] Samsung Project Beyond. http://thinktankteam.info/beyond/.

[12] Under the hood: Building 360 video. https://code.facebook.com/posts/1638767863078802.

[13] YOLO: Real-Time Object Detection. https://pjreddie.com/darknet/yolo/.

[14] Abramov, I., Gordon, J., and Chan, H. Color appearance in the peripheral retina: effects of stimulus size. *JOSA A 8*, 2 (1991), 404–414.

[15] Anderson, R., Gallup, D., Barron, J. T., Kontkanen, J., Snavely, N., Hernández, C., Agarwal, S., and Seitz, S. M. Jump: virtual reality video. *ACM Transactions on Graphics (TOG) 35*, 6 (2016), 198.

[16] AXIS Communications. *An explanation of video compression techniques.* White Paper.

[17] Besharse, J., and Bok, D. *The retina and its disorders.* Academic Press, 2011.

[18] Bowman, D. A., and McMahan, R. P. Virtual reality: how much immersion is enough? *Computer 40*, 7 (2007).

[19] Corbillon, X., De Simone, F., and Simon, G. 360-degree video head movement dataset. In *Proceedings of the 8th ACM on Multimedia Systems Conference* (2017), ACM, pp. 199–204.

[20] D'Acunto, L., van den Berg, J., Thomas, E., and Niamut, O. Using mpeg dash srd for zoomable and navigable video. In *Proceedings of the 7th International Conference on Multimedia Systems* (2016), ACM, p. 34.

[21] El-Ganainy, T., and Hefeeda, M. Streaming virtual reality content.

[22] Fan, C.-L., Lee, J., Lo, W.-C., Huang, C.-Y., Chen, K.-T., and Hsu, C.-H. Fixation prediction for 360 video streaming in head-mounted virtual reality. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video* (2017), ACM, pp. 67–72.

[23] Gaddam, V. R., Riegler, M., Eg, R., Griwodz, C., and Halvorsen, P. Tiling in interactive panoramic video: Approaches and evaluation. *IEEE Transactions on Multimedia 18*, 9 (2016), 1819–1831.

[24] Halpern, M., Zhu, Y., and Reddi, V. J. Mobile CPU's Rise to Power: Quantifying the Impact of Generational Mobile CPU Design Trends on Performance, Energy, and User Satisfaction. In *IEEE International Symposium on High Performance Computer Architecture* (2016).

[25] Hartigan, J. A., and Wong, M. A. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 28*, 1 (1979), 100–108.

[26] Haynes, B., Minyaylov, A., Balazinska, M., Ceze, L., and Cheung, A. Visualcloud demonstration: A dbms for virtual reality. In *Proceedings of the 2017 ACM International Conference on Management of Data* (2017), ACM, pp. 1615–1618.

[27] Ishiguro, H., Yamamoto, M., and Tsuji, S. Omni-directional stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (1992), 257–262.

[28] Jakubowski, M., and Pastuszak, G. Block-based Motion Estimation Algorithms–A Survey. *Opto-Electronics Review* (2013).

[29] Khiem, N. Q. M., Ravindra, G., and Ooi, W. T. Adaptive encoding of zoomable video streams based on user access pattern. *Signal Processing: Image Communication 27*, 4 (2012), 360–377.

[30] Konrad, R., Dansereau, D. G., Masood, A., and Wetzstein, G. Spinvr: towards live-streaming 3d virtual reality video. *ACM Transactions on Graphics 36*, 6 (2017), 209.

[31] Liu, X., Xiao, Q., Gopalakrishnan, V., Han, B., Qian, F., and Varvello, M. 360 innovations for panoramic video streaming. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks* (2017), ACM, pp. 50–56.

[32] Mazumdar, A., Alaghi, A., Barron, J. T., Gallup, D., Ceze, L., Oskin, M., and Seitz, S. M. A hardware-friendly bilateral solver for real-time virtual reality video. In *Proceedings of High Performance Graphics* (2017), ACM, p. 13.

[33] Qian, F., Ji, L., Han, B., and Gopalakrishnan, V. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges* (2016), ACM, pp. 1–6.

[34] Quang Minh Khiem, N., Ravindra, G., Carlier, A., and Ooi, W. T. Supporting zoomable video streams with dynamic region-of-interest cropping. In *Proceedings of the first annual ACM SIGMM conference on Multimedia systems* (2010), ACM, pp. 259–270.

[35] Redmon, J., and Farhadi, A. YOLO9000: Better, Faster, Stronger.

[36] Richardson, I. E. *The H. 264 advanced video compression standard.* John Wiley & Sons, 2011.

[37] Sokołowski, J., and Walczak, K. Semantic modelling of user interactions in virtual reality environments. In *Technological Innovation for Resilient Systems: 9th Advanced Doctoral Conference on Computing, Electrical and Industrial Systems* (2018), Springer, pp. 18–27.

[38] Stockhammer, T. Dynamic adaptive streaming over http: standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems* (2011), ACM, pp. 133–144.

[39] Zare, A., Aminlou, A., Hannuksela, M. M., and Gabbouj, M. Hevc-compliant tile-based streaming of panoramic video for virtual reality applications. In *Proceedings of the 2016 ACM on Multimedia Conference* (2016), ACM, pp. 601–605.