A Low-Latency 360° Streaming System for VR Applications

Ben Civjan
bcivjan2@illinois.edu
University of Illinois Urbana-Champaign
Champaign, Illinois, USA

Abstract

Real-time streaming of 360° video to virtual reality headsets presents unique challenges in maintaining low latency, high frame rates, and perceptually high video quality. In this work, we design and implement an open-source end-to-end 360° video streaming pipeline, enabling video capture, encoding, transmission, and display on a VR headset. The system is fully documented and made publicly available to facilitate future research and development in immersive streaming. We conduct extensive evaluations of the pipeline across a variety of real-world network environments, collecting key performance metrics including frame rate (FPS) and end-to-end latency. To gain fine-grained insight into system behavior, we decompose latency into network transmission and display components. Our results provide a detailed characterization of the performance trade-offs involved in 360° VR streaming and offer a foundation for future optimization of immersive video delivery systems.

1 Introduction

With the rapid advancement of virtual reality (VR) technologies, immersive experiences such as live 360° video streaming are becoming increasingly common across domains like remote collaboration [8, 18], education [7], healthcare [1], and the metaverse [17]. However, delivering high-quality, low-latency 360° video to VR headsets in real time remains a complex challenge. These systems must not only ensure a smooth and responsive visual experience but also handle the significant computational and network demands of high-resolution panoramic video.

Development of real-time 360° video streaming systems is further complicated by limited compatibility with VR headsets and a lack of existing open-source implementations. This makes it difficult for researchers to build upon prior work or integrate components easily into their own systems, hindering experimentation and progress in the field.

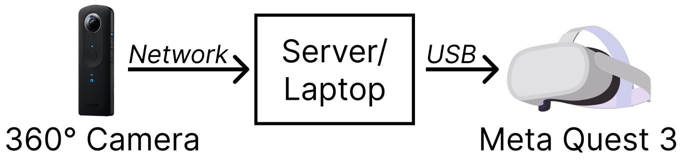

Figure 1: High-level overview of our system's functionality. We want to minimize latency and maximize frame throughput from 360° camera to VR device.

In this work, we design, implement, and evaluate a complete end-to-end video streaming pipeline, shown in Figure 1, that transmits live video from a 360° camera to a VR headset. This allows a user to view live video in an entirely immersive way, as if they are located where the camera is. The user can turn their head and look up or down just as they could if they were there in reality. Our system emphasizes reproducibility and transparency: we open-source the full implementation to enable future researchers to build upon our work. The pipeline includes major stages of streaming: capture, encoding, transmission, decoding, and rendering.

We perform an in-depth analysis of system performance by collecting frame rate (FPS) and end-to-end latency metrics from multiple locations. To better understand latency behavior, we decompose it into its core components, measuring the time spent in transmission over the network and time to display on the headset. This breakdown allows us to identify performance bottlenecks and analyze how different stages contribute to overall delay.

Our contributions are summarized as follows:

(1) A system that streams 360° camera footage in real time to a Meta Quest 3, allowing the user

to view the environment as if they are in the location of the camera. The code is made available at https://github.com/bencivjan/360-video-streaming.
(2) Detailed evaluation of the network and display latency of the system from three different locations.
(3) An examination of the fps throughput of the system using a variety of encoding bitrates.

2 Background and Related Work

360° video, also known as immersive or spherical video, is a type of visual media that captures a scene in all directions. It typically uses an array of cameras or a specialized omnidirectional camera that allows viewers to look around in any direction as if they were physically present in the environment. Unlike traditional video, which frames and directs the viewer's attention, 360° video offers a much wider visual field mapped onto a sphere [20]. This immersive quality makes it especially valuable for applications such as virtual reality and remote presence [8, 17], where spatial context enhances user engagement and realism. Viewers can interact with 360° video through a VR headset for a fully immersive experience or use a touchscreen or mouse to pan around the scene in a 2D player.

360° video introduces unique challenges in stitching, encoding, and streaming, particularly due to its high resolution and wide field of view requirements. Despite these challenges, advancements in real-time processing, camera hardware, and network streaming are rapidly making 360° video a practical tool, enabling more natural and embodied experiences across a growing range of devices and platforms.

We summarize some related areas of research below:

Tiling and Viewport-Adaptive Streaming. Tiling in 360° video describes the technique of dividing the spherical video frame into multiple independent segments (tiles) that can be processed and streamed separately. This approach enables viewport-adaptive streaming, where only the tiles within the viewer's current field of view are transmitted at high resolution, while peripheral areas receive lower quality data. By prioritizing resources for the actively viewed portion of the sphere, tiling significantly reduces bandwidth requirements compared to streaming the entire 360° environment at uniform resolution.

One work that implements such an approach is OpTile [19], a system that optimizes tiling strategies to reduce bandwidth consumption. Unlike traditional methods that stream the entire view or existing fixed-size tiling approaches, which reduce encoding efficiency, OpTile uses an integer linear programming approach to determine optimal, content-aware tile sizes. The system considers both storage costs and expected user viewing patterns to create non-uniform tile distributions.

Another work [12] presents an end-to-end adaptive streaming system for 8K 360° video that focuses on maximizing the perceived quality within a viewer's viewport in virtual reality applications. The system divides equirectangular frames into tiles and uses a novel extension to the DASH protocol to describe each tile's position and center in spherical coordinates. A viewport-aware bitrate selection algorithm assigns higher bitrates to tiles inside or near the viewport and gradually reduces quality with distance, ensuring efficient use of bandwidth.

Stitching. Stitching in 360° video refers to the process of seamlessly combining multiple camera feeds or video segments into a coherent spherical panorama. This technique involves aligning overlapping image regions, correcting for differences between adjacent views, then blending these boundaries to create an uninterrupted immersive environment. Stitching can be a primary cause of delay in real-time 360° video transmision. [2] addresses the unique challenge of real-time stitching in live 360 video streaming by introducing an event-driven, tile-based stitching system that dynamically adjusts to semantic relevance and computational constraints. The core insight is that stitching, not network transmission, is the primary bottleneck in live 360 pipelines. Through a hybrid of offline profiling and online greedy search, the system selects optimal tiling and scaling configurations per frame.

Viewport Prediction. Viewport prediction is extremely useful when coupled with viewport-adaptive streaming, because it allows the FoV adaption to function more efficiently. One such example is SEAWARE [13], a semantic-aware 360-degree video streaming framework that enhances view prediction by incorporating object-based information into the prediction pipeline. Unlike prior methods that rely solely on historical head movement data, SEAWARE offloads computationally expensive deep learning-based video analysis to the server, which identifies and tracks objects of interest across video segments. The client receives this semantic metadata and combines it with spatial tile descriptions to request future tiles more intelligently. By integrating with existing MPEG-DASH infrastructure and leveraging semantic cues, SEAWARE improves prediction accuracy and reduces bandwidth usage, thereby enhancing the overall quality of experience for viewers.

Codecs. Video codecs are essential for efficient 360° video transmission, as they significantly reduce the large data sizes associated with immersive video content while maintaining acceptable visual quality. Due to the high resolution and panoramic nature of 360° video, which captures scenes in every direction, raw footage is extremely bandwidth-intensive. Codecs like H.264 [6], H.265 [15], and AV1 [5] enable real-time streaming and playback by compressing video data in a way that balances quality, compression ratio, and computational complexity [21]. This is essential for the practical application of real-time 360° video streaming.

3 Motivation

Despite the breadth of work in this space, few efforts have provided open-source, end-to-end implementations of 360° video streaming systems that are integrated with a VR headset, where the user can observe the video entirely from the perspective of the camera. Our goal is to fill this gap by developing a modular, fully functioning, and open-source pipeline that supports live capture, transmission, and display in VR. This system aims to enable easier experimentation, accelerate research, and support emerging applications such as telepresence and immersive training.

4 System Design

The following section describes the components of the system in detail, as shown in Figure 2.

360° Camera. We use the Ricoh Theta [14], a consumer-grade 360° camera equipped with dual wide-angle lenses that capture overlapping fields of view. These images are stitched in real time to produce a full spherical image in equirectangular format. The camera is operated in live streaming mode, allowing it to be recognized by the host computer as a standard webcam. This enables frame capture directly from video input streams, simplifying integration into our real-time pipeline.

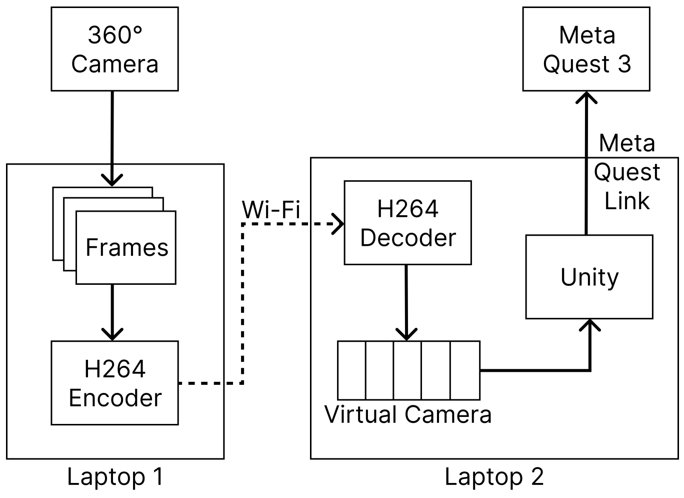

Figure 2: A detailed overview of the streaming system components.

H.264 Encoder and Decoder. Captured frames are encoded using the H.264 codec [6] with the ultrafast preset to minimize latency while maintaining acceptable compression. The encoded video stream is then transmitted to a receiving device over a local network. On the receiver side, the stream is decoded back into raw frames, which are then written to a virtual camera.

Virtual Camera. The virtual camera acts as a software-emulated video input device, allowing applications to access video frames as if they were coming from a physical webcam. It serves as a shared buffer between the decoding process and applications like Unity, which can read from the virtual camera in real time. This design allows Unity to ingest the 360° frames as standard input.

Unity and Meta Quest Link. Unity [16] renders the video by applying an equirectangular projection to the incoming 360° frames using a custom fragment shader. This shader maps the flat, 2D equirectangular format into a spherical representation, which the Unity camera can then render for VR. Meta Quest Link is used to connect the computer to the Meta Quest 3 [9] headset over USB, enabling Unity to stream rendered frames directly into the headset with low latency. This setup allows the user to view the real-time 360° video stream in an immersive VR environment.

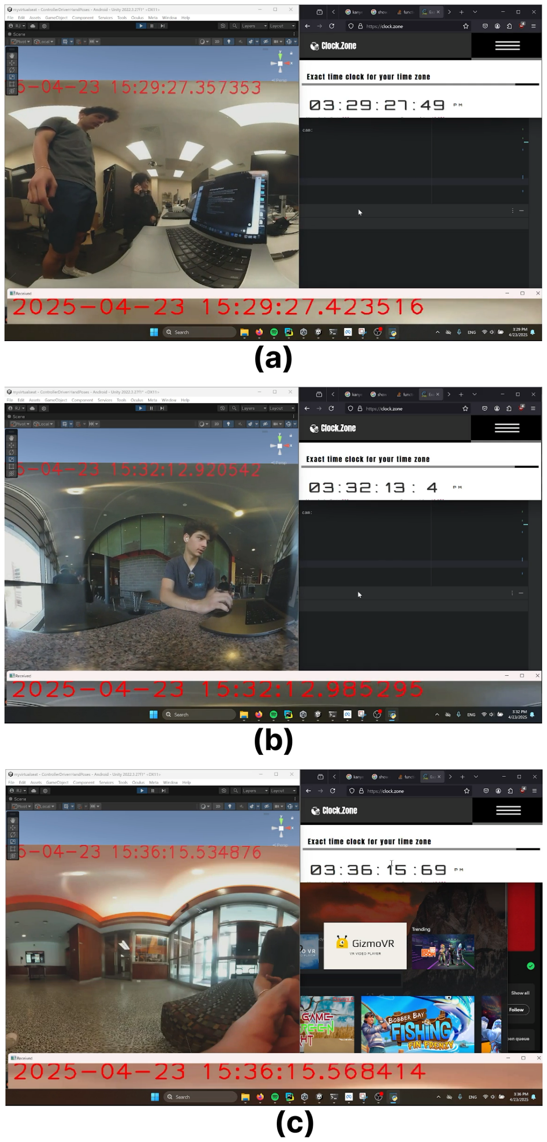

Figure 3: Our evaluation methodology for latency. We overlay timestamps on the images to determine the precise time a frame is sent, received, and displayed. We evaluate in the Monet lab (a), Siebel lobby (b), and a neighboring building (c).

5 Implementation

We implement the system using OpenCV [11] for video capture and processing, PyAV [4] for encoding and decoding, and PyVirtualCam [10] for virtual camera output. The system is composed of a client and a server that communicate over a TCP socket, enabling end-to-end streaming of encoded video from source to display.

On the client side, video frames are captured either from a webcam or a video file using OpenCV's VideoCapture API. The resolution and frame rate are automatically retrieved to configure the pipeline. Each frame is converted to RGB format and encoded using PyAV, with customizable parameters such as bitrate and encoder preset. The resulting encoded packets are transmitted to the server over a socket connection, with each packet prefixed by a 4-byte header to indicate its length and ensure correct reassembly on the server side.

The server listens for incoming connections and spawns a new thread per client. Upon receiving a packet, it reads the header to determine the packet size, retrieves the full data, and decodes it using PyAV. The decoded frames are streamed to a virtual camera via PyVirtualCam. This integration allows the streamed video to be accessed by Unity.

The frames from the virtual camera are mapped into an equirectangular projection in Unity using a custom fragment shader. The Unity scene is transmitted using a USB cable to the Meta Quest 3 using the Meta Quest Link application.

6 Evaluation Methodology

For debugging and synchronization, we overlay timestamps onto each video frame using OpenCV. To ensure accuracy, we use the website *clock.zone* [3], which is synchronized to an atomic clock, as a consistent time reference. We then screen record three key timestamps during streaming: (1) the timestamp on the frame immediately after decoding, (2) the frame as displayed within Unity, and (3) the current atomic time shown on the clock website. This setup allowed us to measure the latency from the moment a frame was timestamped on the sender to when it appeared in the VR headset. To evaluate performance in different network conditions, we conduct this process in three locations: inside the Monet lab, in the Siebel Center lobby, and in a neighboring building. These steps are illustrated in Figure 3.

To evaluate fps throughput, we stream live camera footage for about two minutes from the 360° camera to the server located in the same room. The only variable is the encoding bitrate parameter in the H.264 encoder.

7 Results

The results of our latency experiments are shown in Figure 4. In all locations, network latency (the time

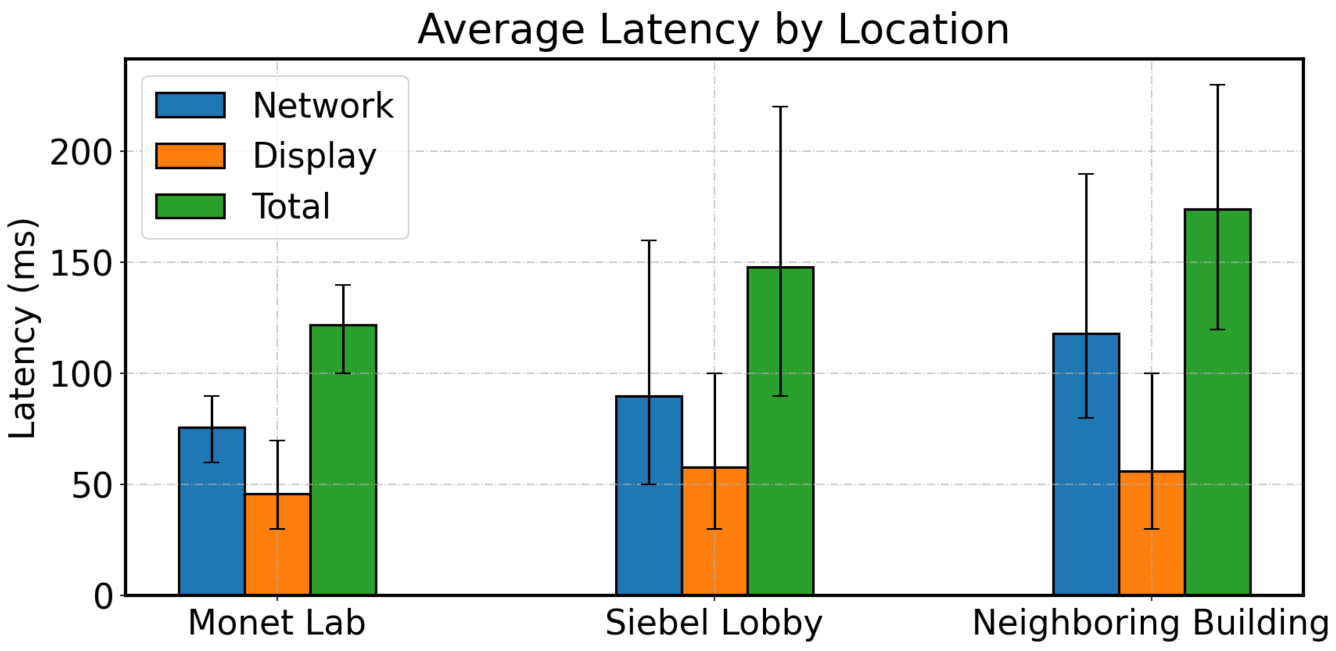

Figure 4: Latency measurements at 10 Mbps encoding bitrate.

from frame encoding on the client to decoding by the server) dominates the total latency, indicating that network conditions and transfer speeds are the primary bottlenecks. This latency increases with distance and/or network complexity: the Monet Lab, being in the same room as the server, exhibited the lowest average receive latency (76 ms), while the neighboring building had the highest (118 ms). The Siebel lobby, which was in between each, had an average latency of 90 ms. This aligns with expectations, as increased distance and the potential use of external routing can introduce additional delay.

Display latency (the time from frame receipt to rendering in Unity) remained relatively stable across all locations, ranging from 46–58 ms. This suggests that the server-side processing and virtual camera output are not significantly impacted by network location.

The Monet Lab achieved the best performance with an average total latency around 122 ms, followed by the Siebel Lobby (148 ms), and the neighboring building (174 ms). Error bars representing min and max latency indicate higher variability in more remote or complex network environments, particularly in the Siebel Lobby and neighboring building, highlighting potential instability in network performance or routing.

Figure 5 shows the relationship between bitrate and average frames per second during 360° video streaming. Interestingly, the system maintains a stable fps of approximately 17.9 at low bitrates of 0.1 Mbps and 1 Mbps. However, as the bitrate increases beyond 10 Mbps, the average FPS begins to decline, reaching 15.67 FPS at 50 Mbps and 15.08 FPS at 100 Mbps. This trend shows that higher encoding bitrates introduce additional overhead. These results highlight a performance sweet spot around 1 Mbps, where the system achieves both high visual quality and smooth frame delivery.

8 Discussion

In this section we discuss challenges encountered during the project and some future applications of this work.

8.1 Challenges

A major challenge encountered during development and testing was the incompatibility of VR devices with Linux, which made system-level integration and end-to-end testing much more difficult. The first iteration of the H.264 encoder/decoder ran reliably and easily on Linux, but had to be changed to be compatible with Windows.

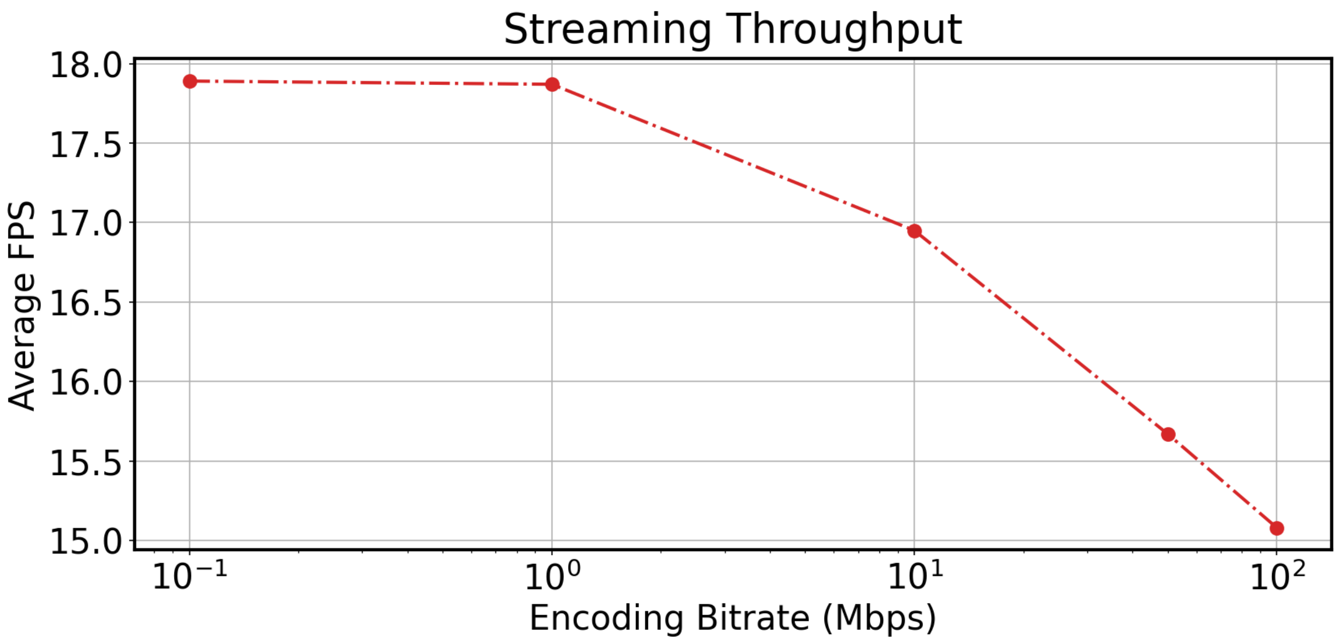

Figure 5: Average FPS throughput while streaming from 360° camera to server.

This is because most commercial VR devices, including the Meta Quest series, offer limited or no official support for Linux-based platforms. This creates friction for developers and researchers who rely on Linux for low-level system control, performance profiling, or experimentation with novel streaming protocols.

Additionally, attempts to use Meta Quest Link, which is required to connect to the Meta Quest 3, on a Windows system equipped with an NVIDIA RTX 4060 GPU were unsuccessful, despite official documentation stating compatibility. Debugging was complicated by the opaque nature of the Meta software stack and limited diagnostic feedback, making it unclear where the issue came from.

These challenges highlight the broader difficulty of integrating commodity VR hardware into experimental systems, where cross-platform support and hardware compatibility often lag behind marketing claims.

8.2 Future Work and Applications

This system serves not only as a real-time 360° video streaming solution, but also as a flexible platform for exploring a range of research directions and application domains. Its modular and extensible design makes it well-suited for studies in low-latency video transmission, adaptive media streaming, and human-centered immersive communication.

One immediate application is immersive teleconferencing, where the system can stream live 360° video from a camera to a VR headset, enabling participants to naturally look around and experience spatial presence. This creates opportunities to study how immersive environments affect user engagement, situational awareness, and collaboration in virtual meetings compared to traditional video calls.

Another future direction is using this system to support training simulations and remote observation, such as firefighter or surgical training. By streaming real-world environments to remote observers in real time, researchers can explore how immersive perspectives impact learning outcomes, attention, and decision-making under pressure.

The system can serve as a starting point for implementing and evaluating new codecs, transport protocols, or synchronization techniques for VR/AR systems. For example, the current TCP-based transport layer could be swapped for WebRTC or QUIC to investigate performance under lossy or mobile network conditions.

9 Conclusion

In this work, we presented a complete, open-source system for real-time 360° video streaming from a camera to a VR headset, enabling immersive remote viewing. By analyzing performance across different environments, we quantified the system's latency and throughput characteristics, revealing how encoding bitrate and network conditions affect the overall user experience. Our results shed light on key performance bottlenecks and demonstrate that smooth playback is achievable, making the system a practical tool for research and prototyping. This work lays a foundation for broader research of 360° streaming in VR applications.

Acknowledgments

I would like to thank Revan Ji for his contribution to the development and evaluation of this project, particularly with Unity and the VR device development.

References

[1] Gaurang Bansal, Karthik Rajgopal, Vinay Chamola, Zehui Xiong, and Dusit Niyato. 2022. Healthcare in metaverse: A survey on current metaverse applications in healthcare. *Ieee Access* 10 (2022), 119914–119946.

[2] Bo Chen, Zhisheng Yan, Haiming Jin, and Klara Nahrstedt. 2019. Event-driven stitching for tile-based live 360 video streaming. In *Proceedings of the 10th ACM Multimedia Systems Conference*. 1–12.

[3] Clock.zone. 2024. Clock.zone: Online synchronized atomic clock. https://clock.zone/. Accessed: 2025-05-09.

[4] Barney Gale and contributors. 2024. PyAV: Pythonic bindings for FFmpeg. https://github.com/PyAV-Org/PyAV. Accessed: 2025-05-09.

[5] Jingning Han, Bohan Li, Debargha Mukherjee, Ching-Han Chiang, Adrian Grange, Cheng Chen, Hui Su, Sarah Parker, Sai Deng, Urvang Joshi, et al. 2021. A technical overview of AV1. *Proc. IEEE* 109, 9 (2021), 1435–1462.

[6] ITU-T. 2019. *Advanced Video Coding for Generic Audiovisual Services*. Technical Report H.264. International Telecommunication Union. Accessed: 2025-04-08.

[7] Dorota Kamińska, Tomasz Sapiński, Sławomir Wiak, Toomas Tikk, Rain Eric Haamer, Egils Avots, Ahmed Helmi, Cagri Ozcinar, and Gholamreza Anbarjafari. 2019. Virtual reality and its applications in education: Survey. *Information* 10, 10 (2019), 318.

[8] Tuomas Kantonen, Charles Woodward, and Neil Katz. 2010. Mixed reality in virtual world teleconferencing. In *2010 IEEE Virtual Reality Conference (VR)*. IEEE, 179–182.

[9] Meta Platforms, Inc. 2023. Meta Quest 3 VR Headset. https://www.meta.com/quest/quest-3/. Accessed: 2025-05-09.

[10] Takuma Mori and contributors. 2024. pyvirtualcam: Python library for virtual camera streaming. https://github.com/letmaik/pyvirtualcam. Accessed: 2025-05-09.

[11] OpenCV Team. 2024. OpenCV: Open Source Computer Vision Library. https://github.com/opencv/opencv. Accessed: 2025-05-09.

[12] Cagri Ozcinar, Ana De Abreu, and Aljosa Smolic. 2017. Viewport-aware adaptive 360 video streaming using tiles for virtual reality. In *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2174–2178.

[13] Jounsup Park, Mingyuan Wu, Kuan-Ying Lee, Bo Chen, Klara Nahrstedt, Michael Zink, and Ramesh Sitaraman. 2020. Seaware: Semantic aware view prediction system for 360-degree video streaming. In *2020 IEEE International Symposium on Multimedia (ISM)*. IEEE, 57–64.

[14] Ricoh Company, Ltd. 2017. Ricoh Theta 360 Camera. https://www.ricoh360.com/theta/. Accessed: 2025-05-09.

[15] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (2012), 1649–1668. doi:10.1109/TCSVT.2012.2221191

[16] Unity Technologies. 2024. Unity Real-Time Development Platform. https://unity.com. Accessed: 2025-05-09.

[17] Dapeng Wu, Zhigang Yang, Puning Zhang, Ruyan Wang, Boran Yang, and Xinqiang Ma. 2023. Virtual-reality interpromotion technology for metaverse: A survey. *IEEE Internet of Things Journal* 10, 18 (2023), 15788–15809.

[18] Mingyuan Wu, Ruifan Ji, Haozhen Zheng, Jiaxi Li, Beitong Tian, Bo Chen, Ruixiao Zhang, Jacob Chakareski, Michael Zink, Ramesh Sitaraman, et al. 2024. Scene Graph Driven Hybrid Interactive VR Teleconferencing. In *Proceedings of the 32nd ACM International Conference on Multimedia*. 11276–11278.

[19] Mengbai Xiao, Chao Zhou, Yao Liu, and Songqing Chen. 2017. Optile: Toward optimal tiling in 360-degree video streaming. In *Proceedings of the 25th ACM international conference on Multimedia*. 708–716.

[20] Abid Yaqoob, Ting Bi, and Gabriel-Miro Muntean. 2020. A survey on adaptive 360 video streaming: Solutions, challenges and opportunities. *IEEE Communications Surveys & Tutorials* 22, 4 (2020), 2801–2838.

[21] Ticao Zhang and Shiwen Mao. 2019. An overview of emerging video coding standards. *GetMobile: Mobile Computing and Communications* 22, 4 (2019), 13–20.