An Interactive and Immersive Remote Education Platform based on Commodity Devices

Jiangong Chen* Feng Qian † Bin Li*
*Department of ECBE, University of Rhode Island, Kingston, Rhode Island, USA
†Department of CSE, University of Minnesota - Twin Cities, Minneapolis, Minnesota, USA
{jiangong_chen, binli}@uri.edu, fengqian@umn.edu

***Abstract*—Virtual reality (VR) holds a great potential to provide interactive and immersive learning experiences for students in remote education by using existing mobile devices, which is extremely meaningful during the current pandemic. In such a VR application, satisfactory user experience requires: 1) high-resolution panoramic image rendering; 2) high frame rate; 3) synchronization among users. This requires that either mobile devices perform fast image rendering or today's wireless network can support multi-Gbps traffic with extremely low delay, neither of which is the case in current practice. In this demo, we develop a platform for interactive and immersive remote education based on commodity devices, where a server performs rendering to ensure that the rendered images have high-resolution (2560×1440 pixels) and are displayed at a high frame rate (60 frames per second) on the client-side. We further leverage motion prediction to overcome the diverse round-trip time (RTT) between a server and users and ensure synchronization among users (average 9.2 ms frame latency difference among users), which improves at least 60% and 20% compared to the existing local-rendering and server-rendering methods, respectively.**

I. Introduction

Virtual reality (VR) has a great potential to provide more interactive and immersive experiences for students in remote education than traditional video conferencing platforms such as Zoom and Webex. Indeed, when a teacher introduces the solar system to students, students can visualize various planets in a 3D manner and see the same perspective as the teacher does. When a student has a question regarding a particular part of the solar system, he/she can point to it for further explanations. While VR-based remote education seems appealing, it comes with new requirements and challenges. To provide the best immersive user experience, the VR-based remote education system should provide 1) high-quality panoramic image rendering: users want to ensure a resolution with 2560×1440 pixels and have at least 60 frames-per-second (FPS); 2) synchronization among users: views should be properly synchronized across participating users to support smooth interactions among them.

Existing mobile devices only support low-quality immersive applications due to their constrained CPU/GPU capabilities. Systems such as Furion [1] and Firefly [2] offload compute-intensive rendering load to a powerful server, which wirelessly streams the rendered frames to the mobile device. As such, in this demo, we adopt the server-rendering approach and develop an interactive and immersive remote education platform based on commodity mobile devices. Since students and the teacher have different round trip times (RTT) between their devices and a server, it is difficult for them to see the same view simultaneously. We observe that if a teacher's motion is perfectly predictable and RTTs of all students and the teacher are deterministic, then the server can deliver rendered frames corresponding to each client's displayed time to ensure that all of them see the same view at the same time. However, a teacher's motion and RTTs cannot be accurately predicted in practice. Hence, we leverage motion prediction to improve synchronization performance among users compared with the traditional server-rendering approach.

This research has been supported in part by NSF grants: CNS-1717108, CNS-1815563, CNS-1942383, and CNS-1915122.

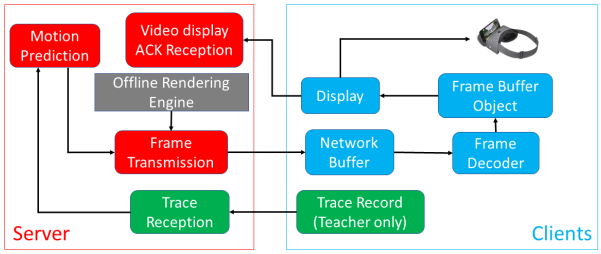


Fig. 1: System Architecture

II. System Architecture

In this section, we will introduce our system for interactive and immersive remote education. Our system consists mainly of three parts: the offline rendering engine, the server, and clients (see Fig. 1), which are explained in details next.

Offline Rendering Engine: We modify a commercial VR scene from [3] such that it is compatible with the offline rendering engine, which splits the VR world into grids. On each grid point, a panoramic frame is captured such that all of the possible views on that position are included. We use Equirectangular projection [4] to process the mega frames such that the frames are compressed using the H.264 codec to reduce the required bandwidth during the transmission. During the runtime, the frames will be swapped to the RAM and stored as a HashMap indexed by its position to be quickly distributed.

Server: The server is responsible for delivering both the pose (from the teacher client) and the rendered frame to the clients. To alleviate asynchronization caused by diverse RTTs of clients, we use motion prediction to deliver future frames

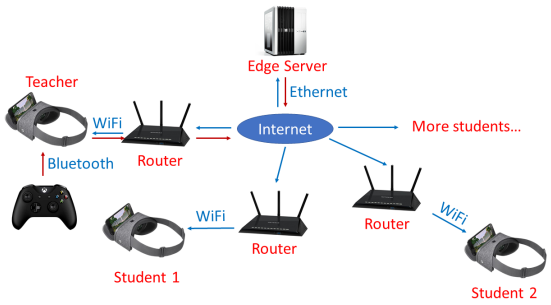


Fig. 2: Platform of the prototype

to clients. We observe that in remote education, the teacher usually stops in some places for a long time to introduce more details, and thus we predict the future positions as the same as the last position. Concerning the orientation, we use the Autoregressive Process (AR) model (see [5]) to predict the future orientation. To quantify the performance of synchronization between the teacher client and student clients, we compare the display time of each frame received by the teacher client and each student client. However, due to different system time (could be hundreds of milliseconds) in each client, it is inaccurate to directly compare them on the client-side. As such, we collect the data on the server-side by recording the time when it receives the display ACK (which will be introduced shortly) from a client. The actual display time can then be calculated by

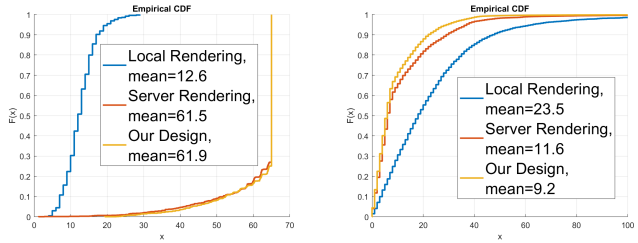
$$t_{\text{display}} = t_{\text{ACK}} - T_{\text{one-way}}$$

where $T_{\text{one-way}}$ is the one-way RTT which is estimated at the beginning of each network connection.

Clients: The client always waits for the frame from the server and puts the received frame into a network buffer pool. Then the frame decoder extracts compressed frames from the network buffer pool and decodes them using Android Media Codec [6]. To alleviate the impact of dynamic network bandwidth, we maintain a frame buffer implemented by OpenGL Frame Buffer Object. Once a panoramic frame is ready, it will be passed to the OpenGL thread where it would be projected to a specific FoV according to the received trace and stored in the frame buffer. The capacity of the buffer is set to only 5 frames to ensure real-time interactivity. Once the frame buffer is full, the oldest 40% of the frames will be released. When it is time to display a frame, the client displays the frames from the frame buffer in order and sends a display ACK to the server which includes the pose of the displayed frame. Different from the student clients, the application designed for the teacher client needs to sample the current pose, and send it to the server periodically.

III. Demonstration

The platform of our prototype is shown in Fig. 2. In our system, the teacher client controls the teaching content by using the Xbox controller and shares the same view with student clients through Google Pixels. We compare the performance of our system design with two existing methods: (1) local rendering, and (2) server rendering. We record a trace played in the teacher client using the controller and replay it to demonstrate the superior performance of our design compared to two existing designs. The demo video is available at [7].



(a) frame rate (b) frame latency difference

Fig. 3: Performance comparison.

We mainly focus on the following two metrics: FPS and frame latency difference, where the latter is defined as the display time difference of similar frames between the teacher client and the student clients. We conduct experiments on five clients under ordinary network bandwidth and add extra RTTs by Linux TC [8] on those clients which are $0, 10, 20, 30, 40$ ms separately. We can observe from Fig. 3 that our system achieves 60 FPS on average and reduces the average frame latency difference by at least 60% and 20% compared to existing local-rendering and server-rendering methods, respectively.

IV. Conclusion

In this demo, we develop a VR-based remote education platform that provides interactive and immersive learning experiences for students using commercial off-the-shelf mobile devices. Specifically, the server delivers high-resolution (2560×1440 pixels) panoramic content to the clients that is displayed at a high frame rate (60 FPS). Furthermore, we utilize motion prediction to mitigate the impact of heterogeneous RTTs between clients and the server and achieve at least 60% and 20% synchronization improvement compared to existing local-rendering and server-rendering methods, respectively.

References

[1] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, N. Dai, and H.-S. Lee, "Furion: Engineering high-quality immersive virtual reality on today's mobile devices," *IEEE Transactions on Mobile Computing*, 2019.

[2] X. Liu, C. Vlachou, M. Yang, F. Qian, L. Zhou, C. Wang, L. Zhu, K.-H. Kim, G. Parmer, Q. Chen *et al.*, "Firefly: Untethered multi-user VR for commodity mobile devices," in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, 2020, pp. 943–957.

[3] "Solar System Unity Asset." [Online]. Available: https://assetstore.unity.com/packages/templates/packs/solar-system-16139

[4] "Equirectangular Projection." [Online]. Available: http://mathworld.wolfram.com/EquirectangularProjection.html

[5] S. M. Kay, *Fundamentals of statistical signal processing*. Prentice Hall PTR, 1993.

[6] "Android MediaCodec API." [Online]. Available: https://developer.android.com/reference/android/media/MediaCodec.html

[7] "Demo Video." [Online]. Available: https://www.ele.uri.edu/~jiangongchen/demo/VRDemo.mp4

[8] "Linux TC." [Online]. Available: http://man7.org/linux/man-pages/man8/tc.8.html