Viewing of Immersive Video on Extended Reality Platforms

Nicholas Wells
Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada
nwwells@mun.ca

Aatman Rangrej
Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada
aurangrej@mun.ca

Amilcar Soares
Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada
amilcarsj@mun.ca

Matthew Hamilton
Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada
mhamilton@mun.ca

***Abstract*—Given the current broad availability of immersive head-mounted displays (HMDs), such as those for virtual and augmented reality (AR/VR) and stand-alone 3D displays, immersive cinema becomes both possible and compelling. In this paper, we present a system to allow the display of immersive video within an extended reality (XR) environment. This system allows for the targeting of immersive video onto various AR/VR/XR platforms; Accomplished by integrating a previously-developed real-time light field rendering system based on hardware-accelerated ray tracing using NVIDIA's Optix API with the OpenXR standard's API. The result is an ability to render real-time views of immersive videos and interactively simulate light field displays, all within various AR/VR/XR systems. We demonstrate using the Meta Quest 2 VR headset. Future work includes developing compression schemes for light field video to support longer, high-quality, and streaming video at interactive rates.**

***Index Terms*—VR/AR, Light Field, Immersive Video**

I. Introduction

With the evolution of immersive displays, a new paradigm for media content has emerged. This new paradigm has created a new challenge: How to effectively create, store, and view immersive content in fast and efficient methods that allow scaling to a visual quality that cannot be distinguished from the real world? These requirements raise other questions, such as: What is the required quality of such content, and how can we evaluate different resolution configuration? In order to answer these questions we need to develop testing procedures which can simulate this immersive media on the displays they will be seen on.

These immersive displays include virtual reality (VR), augmented reality (AR) headsets, and holographic displays and can offer the ability for a viewer to be active in the video experience. A viewer is no longer required to view from a single camera point during a video, but instead any view they may desire using such displays. As a result, video for immersive displays now requires managing all the possible views of a viewer to be stored before it can be used.

The requirement for these new views to be stored when making videos now requires a new video format to be used. An immersive video format requires the ability to store all these new views in new ways that can be both created and view these videos with new software that can interact with current video creation tools.

In this work, we continue our previous work [1] on developing a light field simulator as a solution for viewing immersive videos within VR in real-time. Within that work, we created a light fields simulation software, which used light fields as a means to store immersive images created as light fields using commercial animation software (OctaneRender). These images were then rendered relative to single camera viewing using a our software's ray-tracing based rendering engine.

The work presented in this paper expands upon this original work, to now include the ability to generate and view light field videos instead of just single frames and include the ability to now view these immersive videos within an extended reality environment using the OpenXR API. This work can be found at https://github.com/hamiltonmj/LF-Render

The main contributions of this work are:

- We show a simple pipeline to generating light field videos and displaying them within a pure ray-traced environment.
- We present the limitation of storing immersive video in light field frames directly when attempting to render using conventional rendering techniques.
- We provide a baseline cross-platform program for viewing immersive media on multiple immersive technologies for design testing of both scenes and light field designs.

II. Background

A. Immersive Media

Immersive media is a general term to describe media content used within technologies designed to adapt to a viewer's interactions with the system. Many different technologies can fit into this general term, two in particular are essential to this paper: virtual reality/augmented reality (VR/AR) headsets and holographic displays. Each provides an immersive viewing experience by providing users with more interactions than standard 2D displays. In VR/AR displays, this includes the ability for the headsets to track head movements and adapt the displayed view around the moving head, and in holographic displays, multiple views are displayed at once to a viewer in which they can see a correct version of a scene based on where their head positions are from the display. More specific information about immersive media types can be found in [2].

B. Light Fields

Light fields are a term used to describe all the waves of light that intersect a volume of space. By quantifying the number of waves intersecting the volume and its shape, we can describe how to capture light. Conventional cameras quantify these parameters to capture a single view of the scene. While capturing immersive media, this restriction of a single view is no longer the goal. Displays that attempt to use light fields for immersive media redefine the volume and rays collected to capture multiple views. This approach is well described in many papers, and an initial introduction to the topic was provided by Levoy and Hanrahan [3]. Other ways to capture immersive media can be found in [4] and include multi-Camera Arrays, volumetric capture, etc.

III. Related Work

The concept of creating immersive videos has been explored in other works. One approach to address this problem is by computing the view of a scene in real-time. This is accomplished using mathematical formulations of how light would respond within the real world to a computer-modeled scene, this approach can be better understood in [5]. One work that attempts to follow the real-time rendering principle is [6]. These processes, however, require increased computational power and rendering time the closer to reality it attempts to get Eventually to a point where creating a real-world equivalent becomes impossible in real-time.

As a result, other methods for rendering immersive scenes have been suggested; the method introduced by Broxton et al. [7] suggests using a layered Mesh to represent the scene, allowing a viewer bounded within a sphere to view an immersive video in real-time. This approach limits the viewer to within the boundaries of a sphere, so the ability to scale this method to larger and more general immersive scenes fails. Another problem with this method is the ability of this method to represent depth within an immersive video. The depth being represented is impacted by the number of layered meshes used. As a result, replicating depth found in the real world would produce many meshes, eventually impacting performance.

Another area focusing on displaying immersive videos is simulators designed to replicate the views a real person would view digitally. One such work focused on this is [8]. This simulator does not use a pure ray-traced-based approach to simulate an immersive scene and, as a result, produces inaccurate simulations of how singular light waves interact within the environment.

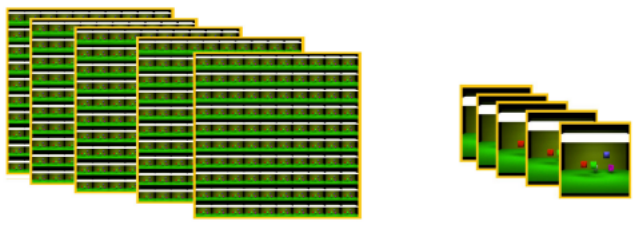

Fig. 1. Left: light field Animation frames, Right: Conventional animation Frames

This work advances the literature on the visualization of immersive videos by providing a complete open-source pipeline for both the production and testing of computer generated, light field based, immersive media. Which Can be specifically tailored for different immersive display technologies, opening up the ability to for the design and testing of many configurations for future research. Unlike the approaches previously discussed, our approach can be used within current media pipelines to generate immersive videos in unbounded environments and also provide ray traced paths to simulate how light actually travels in the real world. As a result of this our work can also be used for testing many different configurations of immersive media designs in with the ability of offering a standard testing environment.

IV. Generating Light Field Video

A. Rendering

The first step in accomplishing our goals was the creation of a dataset to test on. This dataset would contain a light field video which we have rendered. This rendering was done so by modifying our original work's rendering system. This modification included the ability to now capture multiple light field video frames from a scene, instead of the original single light field captured. This modification was accomplished using Maya, a commercially available rendering software with otoys Octane render plugin. The adoption of this software allowed us to add our custom OSL camera shader, which could capture a light field. To then capture a light field video, we would apply a similar format to animation for standard 2d video, involving setting up the objects of the scene at each animation frame, see figure 1 for a comparison between 2D video and Light Field Video. The only modification we apply to the animation is replacing the regular-use camera with our custom OSL camera. This allows us to capture a light field in the animation frame. We would then just render each frame like done with standard 2d animation at each time frame, now using our OSL camera.

B. Storage

With our new rendering system for creating light fields, we now discuss how we store these light field videos within our system. Within our simple testing, we keep each frame stored as a simple PNG file that has been labeled relative to the animation frame it represents *lightField_AnimationFrame.png*. With more complex video, compression of such images would be necessary, and standard video compression schemes could be applied such as HEVC [9].

C. Limitations

With the design of our rendering system, the ability to generate light field videos at any size and scale is possible. Our work does not impose restrictions on the light fields being generated and as such, they can scale to any size application and video length desired by the user. The rendering time although will become an impeding problem as light field quality increases. This can be shown with the simple demo scenes rendered for this paper. The rendering time of these demos took approximately 14 hours, with simple geometry and reflection. When scaled to more complex scenes with increased quality and size of the video, this rendering time will increase.

It is worth noting that the rendering times for these demos do not represent a minimum rendering time for a scene of that size, the complexity of the scene and rendering parameters set before rendering hold a great amount of power in increasing/decreasing the performance of the system. But in order to render light field videos at the same quality as 2D rendering the size and quality of the light field video will greatly increase the rendering time required.

V. Real-time Light Field Video Viewer

A. Basics

Our light field video viewer is an extension of our previously created light field viewer. This previous software created a light field viewing simulator using NVIDIA's Optix API. This application creates a plane in which we can display a light field onto, in which a single camera can move around and the resulting view from the camera is taken as a viewer looking at the light field at that location. Example views from this camera can be seen in figure 2. This system can accept light fields with any parameters although hardware constraints can limit this, and the viewing camera can be modified based on viewing requirements. This simulator is also able to render 2D images, as well.

The new work done with our simulator is the ability to load entire light field videos into the simulator and display them. This is accomplished by loading all light field frames of the video into the system RAM as individual light fields, during the start up of the program. We then move all these frames into the graphics card memory (VRAM) and launch the application. Once launched we now track the running time of the application so we change the displayed light field to represent the currently required animation frame based on the running time of our simulator. Once the video animation has been completed our light field simulator will loop over the video. The ability to also view conventional 2D video is also present within this new design.

The ability to change the viewed video during run-time is also available and the same process as at start up will occur, first loading the video into ram then to VRAM and then the program will continue to run.

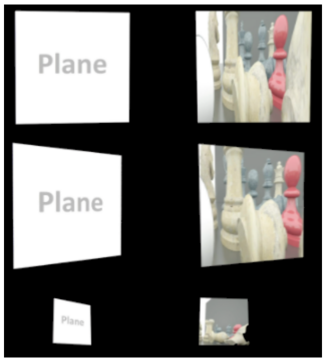

Fig. 2. Rendered Views of Light field by moving Camera around. Scene rendered with OTOY OctaneRender software.

B. Memory Requirements

The largest constraint to our current light field viewer is that of both the RAM and VRAM required to store a light field video. The need for it to be in VRAM although is a higher concern compared with the system ram, as the cost of VRAM is much higher. As a result, we are constrained to making light field videos that will fit entirely within a system's VRAM.

This is an ever-present challenge within our work as the size of even a single light field frame of high quality and viewing angle, which could compare with standard HD resolutions, would still be far too large to fit in any consumer-grade hardware. A naive equation for determining the number of pixels for a specific light field is:

$$LFSize = M_x * M_y * N_x * N_y \tag{1}$$

where M_x,M_y represent the spatial resolution of a single view (spatial resolution) and N_x,N_y represent the number of views in each direction (Angular resolution). The equation for standard 2D video can be derived from this equation by applying that N_x and N_y = 1 as a result it is clear that any light field frame will be $N_x * N_y$ larger than a standard 2D video. This problem then extends when we want to include enough frames for an entire video. The equation for video becomes:

$$LFVideoSize = LFSize * fps * L \tag{2}$$

Where L is the length of the video in seconds, and fps is the frame rate of the video per second. By applying these equations to a concrete example we can quickly discover the physical problem of rendering light field videos, compared with rendering 2D video. We will take a 30-second video with a frame rate of 60 would with a standard HD quality for our example. For a none light field display would be:

$$= (1920 * 1080 * 1 * 1) * 60 * 30 = 3.7 \ billion \ pixels \tag{3}$$

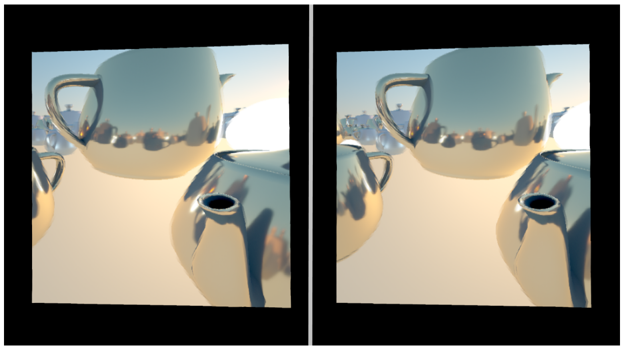

Fig. 3. Left and Right eye view in VR

Compared with using a light field with both N_x and N_y = 60 (This does not represent high Angular resolution):

$$= (1920 * 1080 * 60 * 60) * 60 * 30 = 13.4\ trillion\ pixels \quad (4)$$

As can then be seen a light field video requires orders of magnitude more space than a conventional 2D display and we cannot replicate light fields at any scale or quality to compare. So within our current design, there is a quality ceiling that we will hit when attempting to store a light field video that is larger than the VRAM of the system.

C. Basic Operation

In order to run our system, you simply need to provide the program with a light field and the specific parameters required to view it correctly: N_x, N_y, FOV (field of view) and the location of the PNG image storing the light field. To view a video the same parameters will need to be provided, except instead of providing a PNG light field image, you will pass the location of the folder containing the labeled PNG video frames, and also provide the fr (frame rate) of the video.

VI. Adapting to HMD-VR

The need for VR adaptation comes from the immersive experience Head Mounted Display (HMD) can provide by enabling the viewer to perceive the depth found within the light field. To adapt to VR, you need to use stereoscopic rendering. This type of rendering involves rendering the same scene twice. In VR these two renderings represent the left and right eye of a human. In applications, you position each camera to mimic the position of the left and right eye, to now capture the same perception as a human can, this effect then also creates the perception of depth.

We extended our previous work, to enable such Stereoscopic rendering, with the use of the OpenXR API. This API provides a standard system for integration with immersive display devices, allowing accessibility across different VR devices, without needing external device-specific knowledge. This then enables our system to be cross-platform enabled to any range of immersive display devices, which support the OpenXR standard.

Our new system accomplishes this stereoscopic rendering with the OpenXR standard, a simple modification to our previous work's view generation process. In our previous work, a single view was generated by taking the camera position and direction and creating a pinhole viewing camera from it, in this new work we simply now create 2 of these cameras, using the positions and orientations provided by OpenXR, (fig 3) shows the generate views from these new cameras.

Format	Video	Single Frame
FPS	886.9	888.6
State Update Time	0.0 ms	0.0 ms
Render Time	0.4 ms	0.4 ms
Display Time	0.6 ms	0.6 ms

TABLE I
Rendering performance of video and single frame

VII. Results

A. Testing Parameters

All results were found using a computer with an NVIDIA RTX 3090 GPU using simple test scenes, not created to facilitate hyper-realistic scenes or with consideration to specific depth design, based on the memory constraints of our system discussed above. To further reduce the impact of the memory constraint discussed above, videos have been rendered with horizontal parallax only (where N_y = 1) to again reduce the light field sizes for rendering. The test Videos used had a length of 7 seconds with 30 FPS. The parameters of the light fields for each frame contained a total resolution of (15360 X 512) pixels, with a 30° FOV.

B. Conventional 2D Displays

In our extention from the previous work we also keep some features useful for testing, this includes the ability to produce a view within the simulator which can be seen on a 2D conventional display. With the new complexity of working with video added, we have run tests to confirm the performance of our new additions have not impacted the running time of the simulator compared with its previous iteration. This test was accomplished by rendering a single frame light field and then a video and comparing their results, table I shows the comparison while running the simulator under both versions. Further results for single frames can be found in our previous work, although when rendering videos, the size of the video impacts the available resolution of the light field being displayed, so comparisons between each version use the lower-quality Light fields required for video.

C. Meta Quest 2

When integrating our system within VR, we began our testing using a Meta Quest 2 VR headset and concluded several results from this initial testing. The biggest conclusion drawn was that we could detect depth within the headset not perceived when viewing our simulator on a 2D display. Fig 4 locates the specific area in which depth was found. The

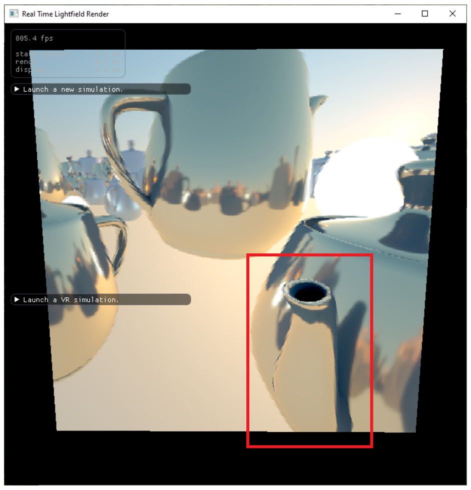

Fig. 4. Location of perceived depth

ability to see the spout of the teapot as a 3d object within the 2D plane proves the ability of our new method to deliver immersive views which cannot be perceived on a 2D display. This depth although was only perceived when working in a static image, when testing on video this depth effect was not present.

We pose several reasons why this may be the case, first, the low resolution required by the video requires us to lower the resolution of the light field, to such a resolution that depth cannot be detected, this problem has been explained in other works such as [10]. Another possible conclusion is the quality of the video scene used for demonstration may not facilitate the discovery of depth, when viewing our test videos, only contained 7 seconds of video in which a square moved around a room, this simplicity may have resulted in an inability for our testing to show a desirable depth.

D. Limitations

The results discussed for each hardware device show that we can display an immersive video on multiple immersive displays with simple effort, but this limited testing of devices prevents a general statement about the cross-platform usage of our software for displaying immersive video. Also, the light fields created for testing may not have differences based on the viewing environment independent from our software itself.

VIII. Conclusion

Within this work, we have shown a simple and adaptable framework that generates immersive video using light fields, which can be displayed in real-time to immersive display devices. We have shown how this process can create these immersive videos within modern commercial animation software using an OSL-Camera and can be adapted to various hardware specifications with ease. We also show how to then display these immersive videos using our own viewing software created with NVIDIA OPTIX's ray-tracing API and the OpenXR API standard. We then describe how our viewing software runs within actual hardware devices in real time.

IX. Future Work

This current work shows how we can generate and render small light field videos and display them both in simulation and in VR, the next step is to further increase the quality of the generated light fields by investigating better approaches to rendering and storing them to remove the large storage burden for even the small and simple demonstration videos we used within this paper. By applying more sophisticated approaches to storage, we will enable the ability to create light field movies at the cinematic scale, and quality and offer new experiences to viewers never experienced before.

References

[1] N. Wells and M. Hamilton, "Towards Immersive Cinematic Video for Immersive Displays," in *The 30th Annual Newfoundland Electrical and Computer Engineering Conference*, St.Johns's, Canada, 2021, pp. 0–4.

[2] M. Domański, O. Stankiewicz, K. Wegner, and T. Grajek, "Immersive visual media - MPEG-I: 360 video, virtual navigation and beyond," in *International Conference on Systems, Signals, and Image Processing*. IEEE Computer Society, jun 2017.

[3] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*. New York, New York, USA: ACM Press, 1996, pp. 31–42. [Online]. Available: http://portal.acm.org/citation.cfm?doid=237170.237199

[4] Immersive IDEA Live Action Working Group Report, "Photographic Live Action Capture for Immersive Media," Digital Experiences Alliance, Tech. Rep., 2021.

[5] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering:From Theory To Implementation*, 2018.

[6] P. Lall, S. Borac, D. Richardson, M. Pharr, and M. Ernst, "View-Region Optimized Image-Based Scene Simplification," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 1, no. 2, pp. 1–22, aug 2018. [Online]. Available: https://dl.acm.org/doi/10.1145/3233311

[7] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. Duvall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec, "Immersive light field video with a layered mesh representation," *ACM Transactions on Graphics*, vol. 39, no. 4, jul 2020.

[8] M. Hamilton, C. Rumbolt, T. Butyn, D. Benoit, R. Lockyer, and M. Troke, "Light Field Display Simulator for Experience and Quality Evaluation," Avalon Holographics, ST. John's, Tech. Rep., 2018.

[9] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards-including high efficiency video coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669–1684, 2012.

[10] M. Zwicker, W. Matusik, F. Durand, H. Pfister, and C. Forlines, "Antialiasing for automultiscopic 3D displays," *ACM SIGGRAPH 2006: Sketches, SIGGRAPH '06*, 2006.