Improved Simulated Viewing of Holographic Imagery

1st Nicholas Wells
Department of Computer Science
Memorial University
St. John's, Canada
nwwells@mun.ca

2nd Devender Singh
Department of Computer Science
Memorial University
St. John's, Canada
devenders@mun.ca

3rd Haoran Wang
Communication Department
Soochow University
Suzhou, China
haoranw23@mun.ca

3rd Amilcar Soares
Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada
amilcarsj@mun.ca

5th Matthew Hamilton
Department of Computer Science
Memorial University
St. John's, Canada
mhamilton@mun.ca

6th Oscar Meruvia-Pastor
Department of Computer Science
Memorial University
St. John's, Canada
oscar@mun.ca

***Abstract*—Holographic displays are now being developed and becoming usable technologies. Developing content for such displays now presents a challenge, but open-source tools to develop these technologies are lacking. This paper further extends our light field simulator that allows for the simulation of viewing of light field images on simulated displays, including improvements to user interaction. Our simulator employs a ray tracing pinhole camera to simulate radial and tangential distortion models found in real-world cameras. We further added a Gaussian blur model to model the point spread function found within light field displays. Further, our simulator provides automated testing for both new display and light field designs through our developed view-based report system. This system allows the automated collection and analysis of perceived views of different light field displays from user provided positions. Overall, our simulator provides future researchers with a fast and standardized testing environment.**

***Index Terms*—light field rendering, quality evaluation of immersive video, display simulation**

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 can we 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 including: how do we evaluate such designs as indistinguishable from the real-world? 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, but instead any view they may desire using such displays. As a result, images and video for immersive displays now require 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 immersive content 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.

This paper builds upon our previous work [1], where we proposed and developed a light field simulator as to allow the viewing of immersive videos within VR in real-time. In that work, we created a light field simulation software. This paper presents a continuation of our previous work, including the incorporation of distortions found in real-world cameras and the light leakage phenomenon that exists within physical light field displays. Additionally, it is designed with an intuitive graphical user interface (GUI) to provide users with the means to visualize camera simulations and the automated collection and analysis of perceived views of different light field displays, providing future researchers with tools to better develop future light field compression and display designs.

The main contributions of this work are:

- Incorporation of camera distortions into the light field simulator providing approximations to real-world camera viewers.
- The incorporation of a Gaussian blur-based model within a light field to better simulate the light leakage phenomenon found in physical displays.
- We provide an improved testing procedure and control system to provide future researchers with a standardized testing procedure for more efficient and repeatable testing of light field display and compression designs.

II. Background

A. Light Fields

A light field is a fundamental representation of light in three-dimensional space, consisting of both spatial and angular

dimensions, as defined by Levoy [2] in 1996. With the fast growth of computing capacity, light fields are becoming increasingly important in diverse fields, such as depth estimation, content editing, image quality, scene reconstruction and view synthesis, and industrial products [3].

B. Light Fields Displays

In modern light field displays, a continuous light field is reconstructed from a set of 2D camera images, using a structure called hogels, where each hogel represents a group of pixels corresponding to different angular directions of light fields. Moreover, real-world displays are subject to various physical phenomena that impact their appearance and performance. One notable phenomenon is the Gaussian blur of pixels, as referenced in [5], which occurs due to the finite size of display elements and optical imperfections. This phenomenon can affect the perceived sharpness and clarity of displayed content. These displays are designed to provide an immersive and high-fidelity viewing experience. Such displays are characterized by their ability to recreate the natural propagation of light and thus deliver an authentic visual experience.

III. Related Work (Literature Review)

Currently, there exist a variety of simulation tools that are applied towards the design of light field displays. We have previously developed a light field simulator which we are extending in this paper [1]. In our previous work we demonstrated how we developed a ray-tracing-based rendering engine, that provided perspective views of a light field to a viewer in a standard camera and VR configuration these images were then rendered relative to single-camera viewing. In this previous work we also discuss a method for generating light field images using octane render, we adapt this method for generating light fields for this work. One concern with our previous work is the lack of ability to simulate flaws found in the real world, as a result simulated views generated from our previous work cannot be used as a one-to-one comparison to how they would be viewed in the real-world.

Another work developing a light field simulator has been proposed by Hamilton et al [7]. This simulator focused on providing modelling the properties of a light field display including the simulation of how a viewer would perceive light leakage using a point spread function. Although light leakage has been included in this paper, Hamilton et al's 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 will result.

In terms of comparing light field images, when dealing with light field images, characterized by a vast number of hogels and pixels, the application of traditional pixel-based comparisons cannot accurately replicate how humans perceive visual information [7]. In this paper, we have focused on addressing these challenges.

IV. Simulating Real World Cameras

In our previous work, we focused on simulating how a real-world light field display would present information to a user [1]. Building upon this foundation, our current work introduces significant enhancements. We have expanded the capabilities of our light field simulator to incorporate two crucial elements: viewer distortions, simulating those encountered in real-world cameras, and point spread function, addressing the "light leakage" problem in light field display.

A. Distortions in Pinhole Camera: Simulating Radial and Tangential Distortion

Camera distortions alter projected images from their true geometry due to imperfections in the lens system. It can manifest in various forms, primarily as radial and tangential distortions. Our simulator utilizes a ray-tracing pinhole camera to replicate the radial and tangential distortion models commonly observed in real-world cameras. The simulator aims to replicate real-world camera behavior by simulating these distortions. To achieve a more accurate representation of real-world distortions, it leverages distortion coefficients obtained from OpenCV's camera calibration process, which incorporates Zhang's calibration models [8].

In our simulator's pinhole camera ray generation process, distortion coefficients $k1$, $k2$, $k3$ for radial distortion, and $p1$, $p2$ for tangential distortion) are applied to the camera rays. These coefficients alter the paths of the rays to simulate real-world distortion effects radial and tangential distortion. We embed these distortion formulas into the generation of the camera rays. These generated rays are augmented to align with the distortions found in a real-camera, returning a distorted view of the light field. This implementation further ensures that the simulator effectively applies distortion to the camera rays, incorporating both radial and tangential effects to produce more realistic rendered images.

Radial distortion emerges from lens curvature, inducing the curvature of straight lines in captured images. The formula for Radial Distortion used in our simulator is as follows:

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{1}$$

$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2}$$

It is characterized by coefficients $k1$, $k2$, and $k3$, which serve as correction modellers for various levels of distortion. While $k1$ rectifies fundamental radial distortion, $k2$ compensates for more intricate lens aberrations, and $k3$ addresses higher-order radial distortion effects. x, and y represent the radial distance of a ray from the center of the distortion, usually the optical center of the camera [9].

Tangential distortion arises when there is a misalignment between the lens and image sensor, leading to image skewing. The formula for tangential distortion used in our simulator is as follows:

$$x_{distorted} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \tag{3}$$

$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \tag{4}$$

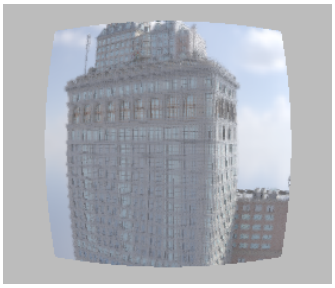

Fig. 1. radial distortion

Fig. 2. Tangential distortion

Tangential distortion is described using two values, p1 and p2. P1 helps reduce distortion caused by objects not being perfectly parallel to the lens, and p2 helps reduce distortion that occurs when objects are not exactly perpendicular to p1. [**?**]. The images below show radial and tangential distortions.

B. Gaussian Function

As previously mentioned in the background section, the light field image is comprised of large quantities of hogels. Those hogels consist of multiple directional rays representing a spread of diverse angles. To achieve accurate image reconstruction from these hogels, each lens was positioned over individual hogels, to precisely converge incident light rays onto their corresponding pixels. This arrangement requires directional pixels to be closely packed beneath the lenses to accommodate the necessary multitude of directional perspectives. Consequently, some errors may arise due to this mechanism. As described in Ramachand's work [5], light field displays can have light leakage problem because of lenticular arrangement, resulting in a reduction of image sharpness during light field reconstruction on the 3D display.

To be more specific, the corresponding pixels "leak" light to their neighboring directional pixels. Therefore, observers viewing these pixels perceive a blended representation of the depicted angles. Hamilton et al. [7] demonstrated the feasibility of modelling this light leakage through the application of a Gaussian based model to windows centered on the directional pixels.

The Gaussian function provides a weight for each neighboring pixel, emphasizing pixels closer to the target pixel while gradually reducing the influence of farther pixels. This weighting mechanism means that we sum together the values of all these neighboring pixels to derive the final RGB color value for the pixel under consideration. This approach ensures that the pixel's color is not only influenced by its immediate neighbors but also takes into account the broader context of the surrounding pixel. Mathematically, the reconstructed pixel RGB value(P) can be defined as:

$$P_{i,j} = \sum_{x=-n}^{n} \sum_{y=-n}^{n} W_{x,y} \cdot P_{i+x,j+y} \tag{5}$$

where x and y represent the horizontal and vertical offsets, respectively, with respect to the target pixel (i, j), n is the size of the neighborhood around the target pixel that is considered for calculating the reconstructed pixel value, W is the weight assigned to the neighboring pixel at position $(i + x, j + y)$, and σ represents the standard deviation of the Gaussian distribution.

In our project, we have further incorporated a slide displayed on the screen to facilitate the adjustment of key parameters, specifically sigma and window size (referring to the size of neighboring pixels). This deliberate inclusion serves to empower users to manipulate the Gaussian blur effect according to their specific preferences.

V. Testing Procedures

A. Testing Procedure Control System

The Testing Procedure Control System in the light field simulator primarily facilitates the saving and restoring of light field states, offering users the ability to manage, return to, and easily switch between simulator configurations and camera setups. This system facilitates effortless transitions between simulator configurations and camera setups. By saving and opening different states, users can swiftly revert to their previous simulator state. This feature expedites users' ability to return to specific simulator setups, enhancing the efficiency of experiments and analyses by automating tasks for reproducibility. Notably, users are spared the need to manage multiple files; they can either append states to the current file or save the state to a new file, preserving both the current camera and simulator configurations.

Saving Simulator States: Users have two convenient methods for saving configurations. The "Save to New File" option generates dedicated JSON files for each state, ensuring planned organization. Alternatively, the "Add Camera State" function incorporates additional states into existing JSON files, simplifying file management and enhancing user-friendliness.

Accessing Saved States: Retrieving saved simulator states is effortless through the open simulator state GUI. Users can easily select the "Load JSON Struct File" option, streamlining the process for better usability. Encouraging clear and descriptive naming conventions further facilitates user identification and satisfaction as the naming scheme should contain relevant details about the camera setup and scenario for easy identification of the files.

This system enhances the user experience by seamlessly integrating into the simulator's workflow. It provides researchers, designers, and developers with a versatile platform for exploration, evaluation, and analysis, thereby contributing to a more efficient and user-friendly research environment.

B. Generate Image Report

Generate image report is a critical function designed to evaluate the similarity between various light field images. To enhance testing efficiency and minimize human error in the experimentation process, we have implemented automated, repeatable testing procedures for evaluating future light fields and displays.

This process involves capturing light field pictures under varying camera parameters and subsequently comparing them with other light field images in the same camera condition. in our project, we employ configuration files to record camera parameters, which we refer to as 'camera states'. These camera states encompass information concerning camera parameters, such as the orientation within a 3D scene, distortion coefficients, field of view (fov) etc. By storing various camera configuration in the camera configuration file, the project will capture images in different states. Subsequently, a comparative analysis can be conducted on these images, assessing their MSE and PSNR. The outcomes of this analysis will then be written into a csv report to allow further analysis.

To enhance the variety of these camera configurations, we have also developed functions aimed at generating diverse camera locations. These camera states can be distributed in through different methods, including random, normal, and uniform distributions. Users also have the flexibility to define their custom camera state distributions using a JSON file within our project framework.

Furthermore, to support automated testing, our project have the option of employing a non-GUI approach for execution. This method involves using text-based commands to run our image report procedure on a large scale number of camera and light field configurations. It's a useful feature that gives users more control and flexibility when attempting to test different light field and camera properties quickly and .

C. Interactive Functionalities and Keyboard Control UI

In addition to Testing Procedure Control System, our simulator boasts a wide array of interactive functionalities designed to enhance user experience. Among these features is a Keyboard Control User Interface (UI) equipped with a key mapping system tailored for diverse functionalities. This mapping system is further supported with the ability to remap the controls when building the application from the source. Through this UI, we can offer future researchers with convenient and fast access to basic analysis tools including the ability to alter the Gaussian blur parameters during run-time and dynamic camera movement of the viewing camera.

VI. Results

All the results were obtained using a computer equipped with an NVIDIA RTX 3060 GPU and 16GB of VRAM. During program execution, the VRAM usage hovers around 11GB, with the GPU operating at 39%. Our light field image size is 1.45GB when compressed in PNG format, with a resolution of 307200 x 307200 pixels. With approximately 200 camera states, the refresh rate of our program's interactive window is approximately 750 frames per second (fps). Notably, our automated image comparison process for light field images of the same size and location took about 120 minutes. In the following sections, we will delve into the details of distortion and Gaussian blur effects.

In the context of a light field configuration, the simulator attributes are defined as follows:type: "lightfield", position: [0, 0, 0], orientation: [23, 23, 21]. Regarding the light field attributes: "Physical Display Dimensions" are set to [0.5, 0.5] and "Holographic Display Dimensions" are configured as [512, 512]. Detailed parameters for camera location, Gaussian blur and distortion are provided in the following table.

Fig. 3. distorted image

Fig. 4. undistorted Image

A. Lens Distortion Effect

To assess the effectiveness of our distortion methodology within the Light Field Simulator, we conducted a validation process. This validation involved a comparative analysis between the distorted images produced by the simulator and their corresponding undistorted counterparts, generated using the OpenCV formula.

Figure (3) displays the images that represent our simulator's radial and tangential distortion processes, with exaggerated coefficients. In practice large distortion coefficients would result in substantial distortions, to match real-world cameras operating the required distortion coefficients are very small. These images serve as a visual reference for the distortion effects applied within the simulator.

We present the results of the validation process in Figure (4), which showcases the outcome of this comparative analysis. The image in Figure (4) highlights a perfectly undistorted representation, especially when the distortion coefficients are minimal. These distortion coefficients need to be kept to a minimum, as real-world camera distortions are typically minimal [10].

B. Gaussian Blur Effect

The results of the Gaussian blur model are clearly displayed in these two images. Figure 5 presents the image without Gaussian blur, while Figure 6 illustrates the outcome of applying Gaussian Blur. It is evident that the sharpness of Figure 6 has notably decreased.

VII. Conclusion

In this paper, we have expanded the scope and functionality of our light field simulator. Key features of this work include the introduction of radial and tangential distortion models observed in real-world cameras being applied to the camera found within the simulator. In addition, we have integrated a Gaussian blur model, as an algorithm to replicate the light leakage characteristic of light field displays. Further, we have introduced an automated testing system that facilitates the

TABLE I
PARAMETERS OF CAMERA LOCATION, DISTORTION AND GAUSSIAN BLUR

Figure Name	eye	lookat	up	fov	res	Distortion					Gaussian Blur	
						k1	k2	k3	p1	p2	sigma	windowSize
Fig 1	[0, 0, 0.559]	[0, 0, 0]	[0, 1, 0]	60	[600, 600]	0.6	0.4	0.3	0	0	0.67	5
Fig 2	[0, 0, 0.559]	[0, 0, 0]	[0, 1, 0]	60	[600, 600]	0	0	0	0.2	0.1	0.67	5
Fig 3	[0, 0, 0.559]	[0, 0, 0]	[0, 1, 0]	60	[600, 600]	0.6	0.3	0.2	0.2	0.01	0.67	5
Fig 4	[0, 0, 0.559]	[0, 0, 0]	[0, 1, 0]	60	[600, 600]	0	0	0	0	0	0.67	5
Fig 5	[0, 0, 0.559]	[0, 0, 0]	[0, 1, 0]	60	[600, 600]	0	0	0	0	0	0.408	1
Fig 6	[0, 0, 0.559]	[0, 0, 0]	[0, 1, 0]	60	[600, 600]	0	0	0	0	0	0.946	7

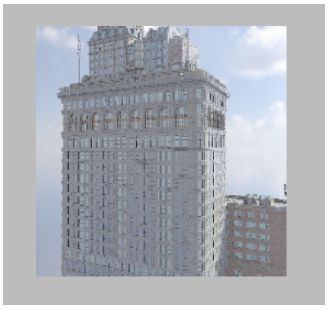

Fig. 5. raw picture

Fig. 6. picture after Gaussian blur

evaluation of new display and light field designs by automating the collection and analysis of perceived views from user-specified positions. We also discuss further developments to the user interface (UI) which presents better tools for future researcher use. This paper showcases a continuing effort to build an open-source light field display simulator that allows researchers access to tools for advancing the development of holographic displays and the exploration of immersive graphics.

VIII. ACKNOWLEDEMENTS

We acknowledge support for this project from NSERC, Mitacs Globalink program and the Memorial University Faculty of Science Undergraduate Research Award (SURA).

REFERENCES

[1] N.Wells, A. Rangrej, A. Soares, and M. Hamilton. "Viewing of Immersive Video on Extended Reality Platforms." The 31st Newfoundland Electrical and Computer Engineering Conference (NECEC), St. John's, Canada.

[2] Marc Levoy and Pat Hanrahan. 1996. Light field rendering. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96). Association for Computing Machinery, New York, NY, USA, 31–42. https://doi.org/10.1145/237170.237199

[3] Zhou, S., Zhu, T., Shi, K., Li, Y., Zheng, W., & Yong, J. (2021). Review of light field technologies. Visual computing for industry, biomedicine, and art, 4(1), 29. https://doi.org/10.1186/s42492-021-00096-8

[4] S. H. Peter Shirley Trevor David Black, 'Ray Tracing in One Weekend'. Aug-2023.

[5] V. Ramachandra, K. Hirakawa, M. Zwicker, and T. Nguyen, "Spatioangular Prefiltering for Multiview 3D Displays," IEEE Trans. Visual. Comput. Graphics, vol. 17, no. 5, pp. 642–654, May 2011, doi: 10.1109/TVCG.2010.86.

[6] Y. Xu, K. Maeno, H. Nagahara, A. Shimada, and R. Taniguchi, "Light field distortion feature for transparent object classification," Computer Vision and Image Understanding, vol. 139, pp. 122–135, Oct. 2015, doi: 10.1016/j.cviu.2015.02.009.

[7] M. Hamilton, C. Rumbolt, T. Butyn, D. Benoit, R. Lockyer, and M. Troke, "P-91: Light Field Display Simulator for Experience and Quality Evaluation," SID Symposium Digest of Technical Papers, vol. 49, no. 1, pp. 1523–1526, 2018, doi: 10.1002/sdtp.12268.

[8] Z. Zhang, "A flexible new technique for camera calibration," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330–1334, Nov. 2000, doi: 10.1109/34.888718.

[9] D. Brown, 'Decentering distortion of lenses', 1966.

[10] R. Szeliski, Computer Vision: Algorithms and Applications. in Texts in Computer Science. London: Springer, 2011. doi: 10.1007/978-1-84882-935-0.