Light Field Photography Using A Controller Operated Camera

by

© Abdelrahman Abdelrahman

A Project Report submitted to the

School of Graduate Studies

in partial fulfillment of the

requirements for the degree of

Master of Science

Supervisor: Dr. Matthew Hamilton

Department of Computer Science

Memorial University of Newfoundland

April 2024

St. John's Newfoundland

Abstract

In this project, we design and implement a system for capturing light field images of a static scene. The system consists of a controller operated camera to capture photos of a scene from a number of vantage points and to generate a light field image from them. This setup can capture many images as needed, vertically and horizontally, on the scale of tens of thousands of images. These are then uploaded to MUN's cloud HPC computing facility CAIR for processing, where it is downscaled, re-mapped in terms of field of view, and compressed. This allows us to capture the many light field rays of a scene and be able to visualize it digitally in the light field renderer. We show the captured images in a light field renderer to validate the overall approach.

Contents

List of Figures

Chapter 1

Introduction

A light field image is a capture of the light rays that in a scene in many directions. In contrast to a regular image where it only captures light intensity entering a single point. The purpose of this image is to allow a 3-dimensional view of a scene as a light field render can render a scene from multiple angles via the information of the light rays [2]. These images can be used in virtual reality and augmented reality displays. The subject of light field rendering was introduced by Levoy and Hanrahan [3] and is based on the idea of representing the light field in 4 dimensions as shown in Figure 1.1. In practice, this would mean capturing many images from multiple positions in 2 dimensions (u, v), each generating a 2-dimensional image (t, s). An example of such images is shown in Figure 1.2.

The renderer we use is made by N. Wells et al. [2]. This renderer has been tested on virtual images generated by 3-dimensional simulator models, but it has never been

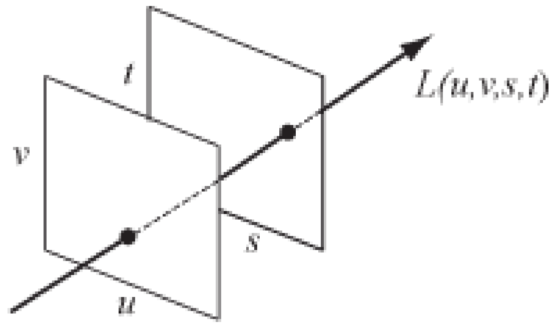
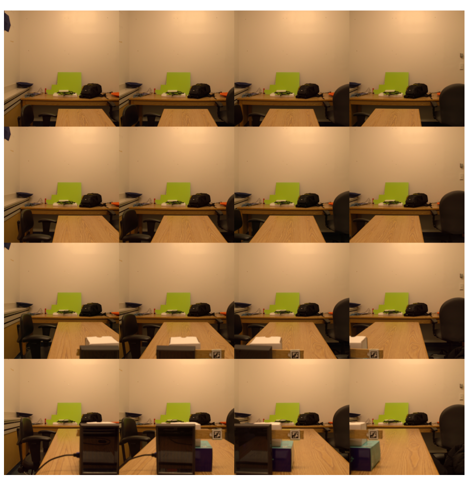
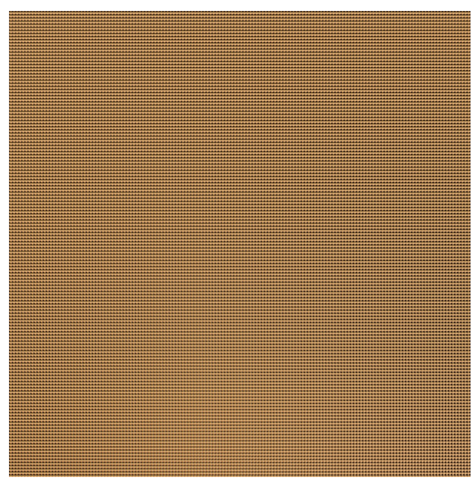


Figure 1.1: 4D parameters of a light field (Marc Levoy 2006 [1])



(a) 4x4 light field image



(b) 150x150 light field image

Figure 1.2: Examples of light field images

tested on images from the real world. This project aims to validate our approach and showcase the renderer working on real images.

Chapter 2

Related Works

There are many methods to capturing a light field image and the process we present is not novel [4]. Generally speaking, there are 3 methods to photograph a light field image.

2.1 Array of Cameras

This method include having an array of cameras aligned in a grid. This technique has the advantage of being able to capture dynamic scenes as opposed to only static images. One example of this implementation has been made by Wilburn et. al [5].

2.2 Array of Lenses

Another imaging technique is to use an array of lens that feed to one camera. The lens could be placed before the main lens of the camera, or in between the main lens and the camera's sensor [6]. This has the same advantage of being able to capture films of dynamic scenes, while being a cheaper option than using multiple sensors. The most famous implementation of this was made with the commercial Lytro camera [7].

2.3 Moving Camera

The imaging technique we present is a controller operated camera. This only has the capability of capturing a static scene, but it is cost-effective as it only needs 1 camera. It is however slow for capturing large dimensions of images. The performance of our technique is detailed in Section 4.3. This method was used in the digital Michelangelo project, where a 3D scan of Michelangelo was created [8].

Chapter 3

Methodology

3.1 Camera operator

This is the imaging technique we present, a controller operated camera system. The system can move the camera freely in a 2-dimensional plane. Figure 3.1 showcases the system being used. The system uses the following hardware:

- Nikon D3200 DSLR Camera
- Optics Focus Controller with an XY Stage
- Computer

3.1.1 Calibration

Before any capture is done with the system, a calibration process is done to get the camera's configuration and properties. The most important of those properties is

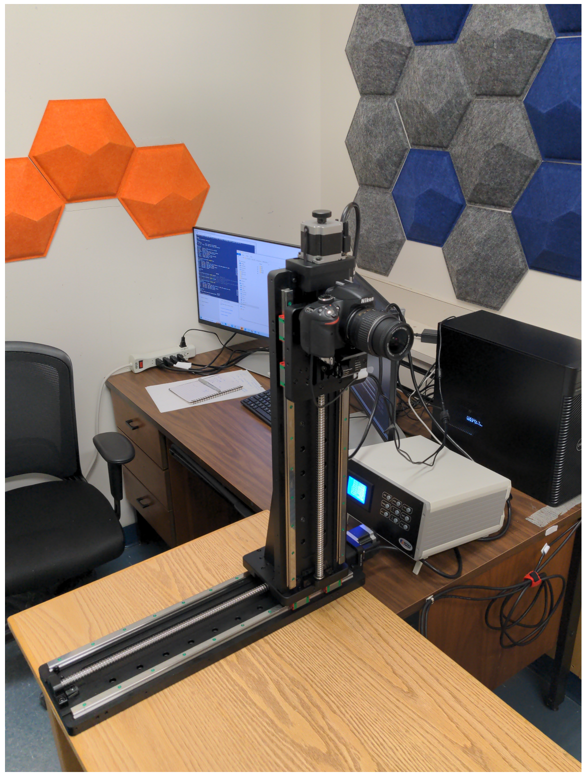


Figure 3.1: Camera operator system

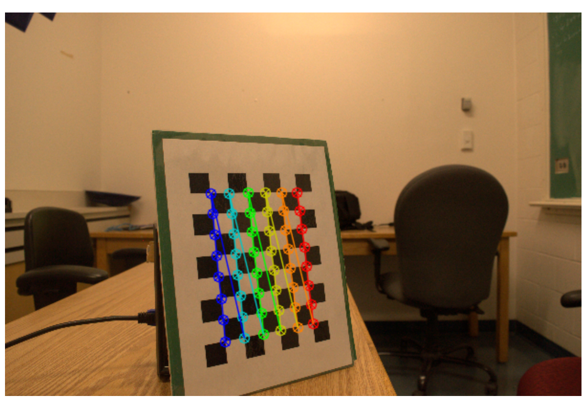


Figure 3.2: Corners detected in the checkerboard calibration process

the angle of view, as this is information the light field render needs to render its projections correctly. We will be using the checkerboard calibration technique used by OpenCV to perform this calibration [9]. Figure 3.2 shows the corners detected in a checkerboard calibration process. With the corner positions, the camera's angle of view could be deduced.

3.1.2 Capture process

The camera system needs to take photographs across all grid points in the 2-dimensional plane. In order to cover all those points efficiently, the camera system moves the camera in a "zigzag" pattern and stops at each point to capture an image. Figure 3.3 shows the capture movement pattern of the system as it covers the grid of points. This figure only shows the system taking a 4×4 set of images. In practice, a good light field image would need a set of more than 100 images in each dimension, e.g. 150×150 or more.

The system can be programmed to take any number of images horizontally and vertically, the controller's precision allows the system to take up to 128,000 images horizontally and vertically.

These images are named after each point the photo is taken from. For example, if the photo is taken in the position "1,2" according to the programmed pattern, the image will be saved as "1-2.nef".

Each image takes approximately 20 MB of data. This demands a high amount

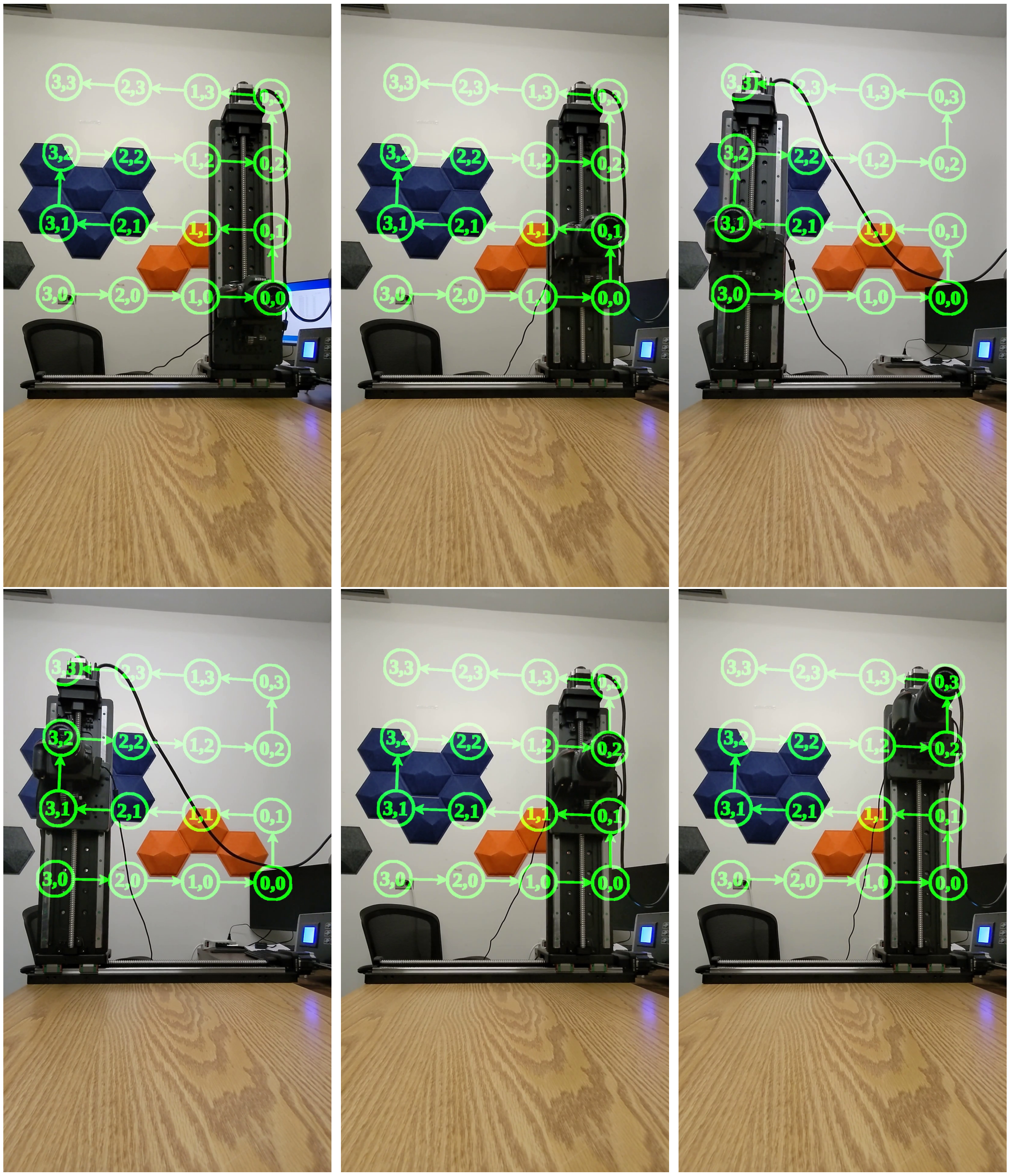


Figure 3.3: Camera system capture movement pattern for a 4×4 set of images. Points and movement pattern are drawn on the photos. Each point has its own co-ordinates that are saved for each image.

of storage for high sets of images. For example, a 150×150 set of images can take about 400 GB of storage. This is why we would need to upload those images to an HPC for further processing. In this project we will upload those images to Memorial University's Centre for Analytics, Informatics and Research computing centre, or CAIR for short, for this purpose.

3.2 Image Processor

A separate process will run on the HPC to process those images. The process will downscale, crop and then stitch the images together to form a single light field image similar to the images shown in Figure 1.2.

Downscaling and cropping are done in parallel and with the use of libvips [10], an image processing library specialized in processing large images. The downscale ratio can be set arbitrarily.

Cropping is done via a remap of the angle of view. Assuming f is the original angle of view and f' is the new arbitrary angle of view to be set, the cropping ratio can be deduced geometrically via Figure 3.4. Via trigonometry, the figure shows that (1)$\frac{l'}{2d} = tan(\frac{f'}{2})$ and (2)$\frac{l}{2d} = tan(\frac{f}{2})$. Via (1) and (2), the cropping ratio $\frac{l'}{l}$ can be deduced to be $\frac{tan(\frac{f'}{2})}{tan(\frac{f}{2})}$.

The images are then finally collected and stitched together in a single image. They are then archived in a ZIP file along with a metadata of the image dimensions and camera configuration in a JSON format. The process is illustrated in Figure 3.5

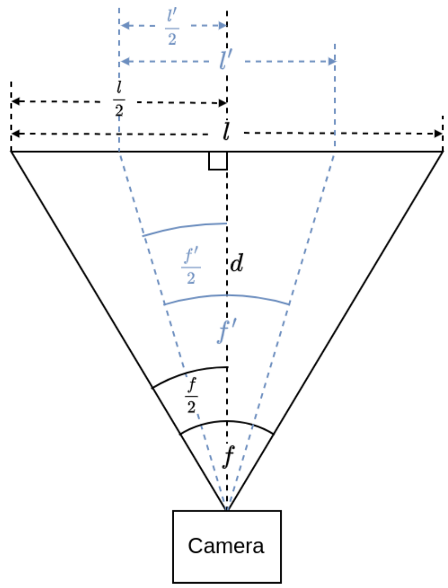


Figure 3.4: Cropping via a remapped angle of view geometry.

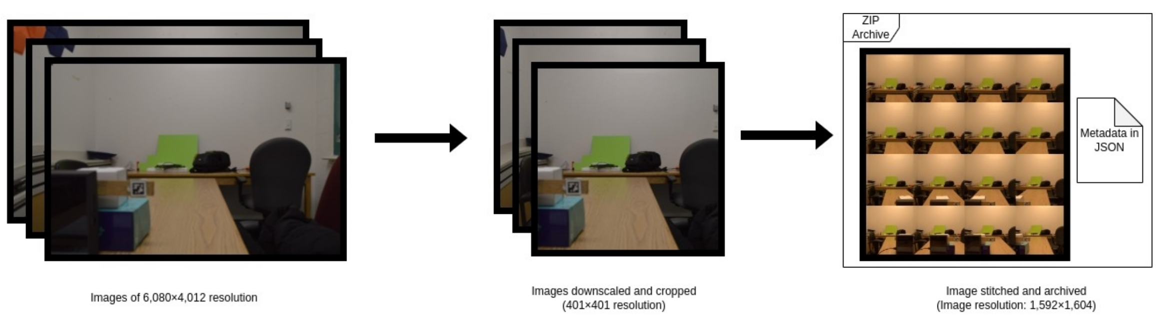


Figure 3.5: Steps of image processor.

Chapter 4

Results

4.1 Demo on the Light Field Renderer

Using the light field renderer made by N. Wells et al. [2], we can view a light field render of the image. Figure 4.1 shows a render of projections in several angles for a 150×150 image, downscaled to use 5% of each dimension, and cropped to 45 degrees in each dimension. The result is what you would expect for a light field render of the image. This validates the output images of our project. Before this project, the project was only tested on images created by virtual 3-dimensional models, therefore this also validates the renderer working on real light field images.

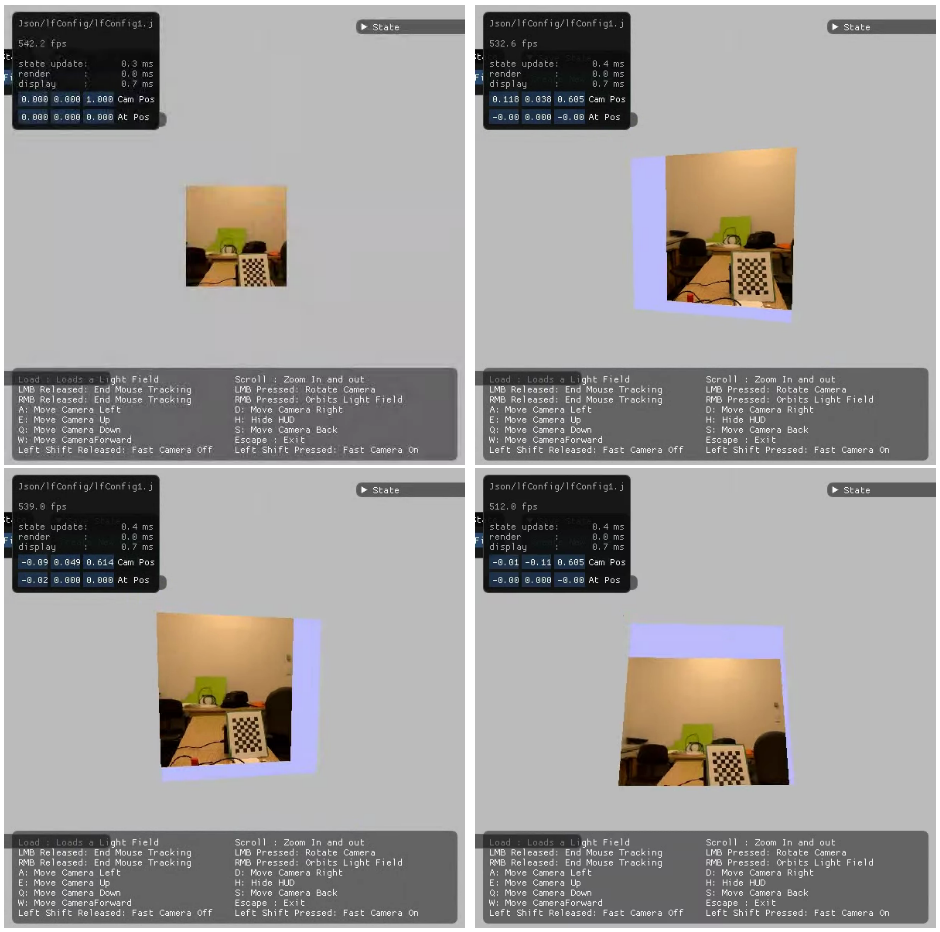


Figure 4.1: Light field render of a 150×150 image.

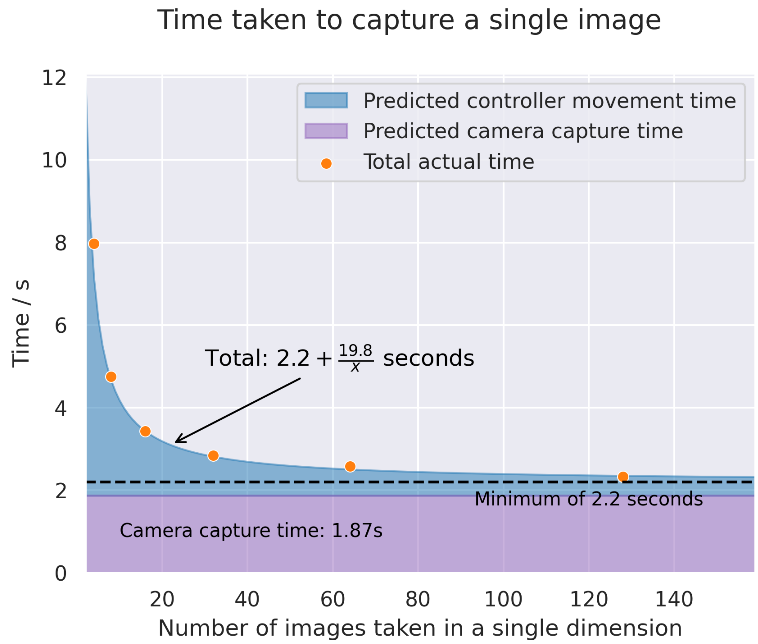


Figure 4.2: Time taken per image.

4.2 Source code

The source code for the camera operator and image processor are open source and available on GitHub [11][12].

4.3 Performance

One limitation of moving camera setups is that it takes a considerable amount of time to capture a single light field image. The amount of time the capture of the light field image takes depends on the number of images taken in total. For example, a 150×150 image will have to take 22,500 images.

Figure 4.2 illustrates the amount of time it takes to capture a single sub-image

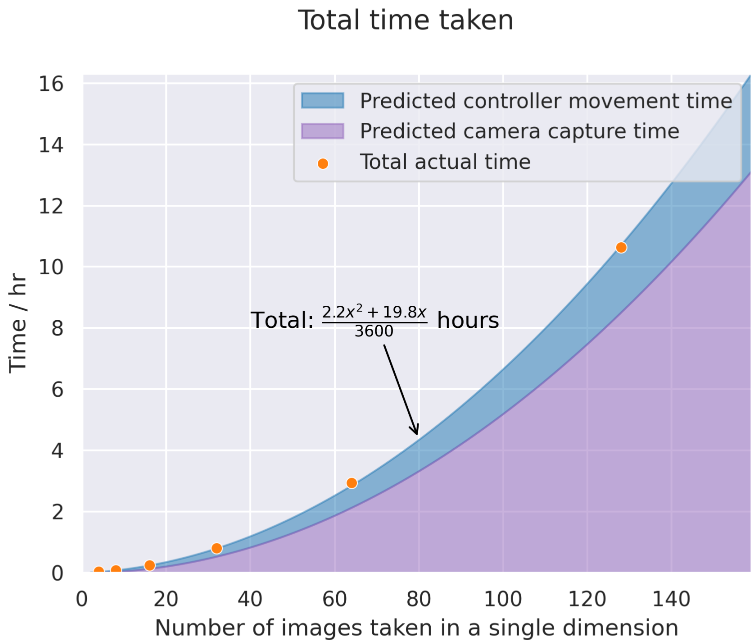


Figure 4.3: Total time taken.

within the capture session. The camera takes a constant amount of time to capture a single sub-image (1.87 seconds). The movement of the camera from one point to the next decreases as the amount of images increase, that is because the space to move is divided into smaller and smaller divisions as the amount increases. Therefore, the time it takes to move the camera decreases as well. However, there is still an overhead for issuing commands to the controller (about 0.33 seconds). Using linear regression, we can work out that the approximate time it takes to capture a single sub-image is $2.2 + \frac{19.8}{x}$ seconds, where x is the number of images to be taken in a single dimension. This means that the minimum time it takes to capture a single sub-image is 2.2 seconds

Figure 4.3 illustrates the amount of time it would take for the whole capture session to complete. This is deduced by multiplying the estimated time it takes to capture a single image by x^2. We can therefore estimate the total time taken for the whole session with the formula $\frac{2.2x^2+19.8x}{3600}$ hours. For example, to take the image shown in the demo (150×150), the figure and formula shows that it would take almost 15 hours.

4.4 Limitations

The main limitation of the project is the slow capture session. It usually takes hours to capture a good light field image as shown in Section 4.3. Using other methods discussed in chapter 2 would work around this issue. Using a camera with a lower capture time would also decrease the capture session time.

Bibliography

[1] Marc Levoy. Light field parameterization. https://commons.wikimedia.org/wiki/File:Light-field-parameterizations.png, August 2006.

[2] N. Wells, A. Rangrej, A. Soares, and M. Hamilton. Viewing of immersive video on extended reality platforms. In *The 31st Annual Newfoundland Electrical and Computer Engineering Conference*, 2022.

[3] MARC LEVOY Pat Hanrahan. Light field rendering. *SIGGRAPH96, Computer Graphics Proceeding*, 1996.

[4] M. Levoy. Light fields and computational imaging. *IEEE Computer Society*, 39(8):46–55, 2006.

[5] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. High performance imaging using large camera arrays. In *ACM siggraph 2005 papers*, pages 765–776. 2005.

[6] Todor Georgiev and Chintan Intwala. Light field camera design for integral view photography. *Adobe System, Inc., Technical Report*, page 1, 2006.

[7] Todor Georgiev, Zhan Yu, Andrew Lumsdaine, and Sergio Goma. Lytro camera technology: theory, algorithms, performance analysis. In *Multimedia content and mobile devices*, volume 8667, pages 458–467. SPIE, 2013.

[8] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3d scanning of large statues. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, page 131–144, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[9] OpenCV. Camera calibration. `https://docs.opencv.org/3.4/dc/dbb/tutorial_py_calibration.html`.

[10] K. Martinez and J. Cupitt. Vips - a highly tuned image processing software architecture. In *IEEE International Conference on Image Processing 2005*, volume 2, pages II–574, 2005.

[11] Abdelrahman Abdelrahman. camoperator. `https://github.com/Abogical/camoperator`.

[12] Abdelrahman Abdelrahman. lfgen. `https://github.com/Abogical/lfgen`.