A Novel Approach For Generating Customizable Light Field Datasets for Machine Learning

Julia Huang  
Northville High School  
SHTEM Program  
Stanford University  
Northville, Michigan  
juliah6169@gmail.com

Toure Smith  
Abraham Lincoln HS  
SHTEM Program  
Stanford University  
San Jose, California  
touresmith10@gmail.com

Aloukika Patro  
Douglas County HS  
SHTEM Program  
Stanford University  
Castle Rock, Colorado  
patro.aloukika@gmail.com

Vidhi Chhabra  
West Haven High School  
SHTEM Program  
Stanford University  
West Haven, Connecticut  
vidhichhabra06@gmail.com

Abstract—To train deep learning models, which often outperform traditional approaches, large datasets of a specified medium, e.g., images, are used in numerous areas. However, for light field-specific machine learning tasks, there is a lack of such available datasets. Therefore, we create our own light field datasets, which have great potential for a variety of applications due to the abundance of information in light fields compared to singular images. Using the Unity and C# frameworks, we develop a novel approach for generating large, scalable, and reproducible light field datasets based on customizable hardware configurations to accelerate light field deep learning research.

Keywords—Dataset, machine learning, light fields, 3D graphics pipeline, vertex processor, Unity Engine, C#

I. INTRODUCTION

A light field (represented by L(u, v, s, t)) consists of a set of 4D light rays through every point in space, describing the amount of light flowing in every direction in space. We can represent light fields using a two-plane representation in general position (see Fig. 1) to model the analytic geometry of perspective imaging. This two-plane representation could be seen as a collection of perspective images of the ST plane each taken from an observer position on the UV plane. In another way, this can be interpreted as many cameras taking photos of the same scene from different perspectives; thus, light fields are technically a collection of photos taken at different angles. Because light fields contain more information than singular images, they have a lot of critical applications in the computer vision field, such as light field depth estimation [1], synthetic aperture photography [2], and 3-Dimensional models of objects.

Fig. 1. A diagrammatic representation of an individual light field, which is composed of light rays from points on the ST plane to a point on the UV plane. In another interpretation, the coordinates (u, v) describe a location of a camera on the UV plane, and the coordinates (s, t) describe the location on the ST plane through which the ray from the camera passes. The parameterization and depiction of light fields with this 2-plane representation are adopted from Levoy and Hanrahan [3, 4].

In the application of depth estimation, the objective is to calculate the depths of all objects in an image, or, in our case, a light field. To compute the depth of light field scenes, several manual optimization-based techniques have been used: using epipolar plane images that contain lines of different slopes, combining defocus and correspondence cues [5], etc. However, these methods either take too much time or result in low accuracies due to noise or occlusions.

Alternatively, when using synthetic aperture imaging, the goal is to measure the sharpness of a synthetic aperture image. Calculation and optimization-based approaches for this type of application include using derivatives or local statistics of image pixel value, variance [6], discrete cosine transform [7], Tian and Chen’s Laplacian mixture model of wavelets method [8], etc. However, similar to the methods in the application of light field...
depth estimation, these optimization approaches are either computationally expensive, inaccurate, and/or time-consuming.

Besides the depth estimation approaches detailed above, plenoptic cameras - commonly known as light field cameras - can also be used to augment photo editing. Plenoptic cameras capture light fields in a scene where many images are taken at different angles. In addition, they allow photographers to alternate the focus and perspective of an image after it is captured because of the angular information of the light captured by the light field camera. However, there are limitations to its use, especially due to noise prevalence in most real-world light field data. For example, depth maps, which are derived from a light field captured through a plenoptic camera, appear unclear.

II. RECENT RELATED WORK

Fortunately, recent machine learning approaches for light field applications have shown the ability to be more precise and faster than previous conventional methods, even in the presence of noise. However, there is a lack of data surrounding light fields, both in terms of the number of datasets publicly available and the amount of data in current sparse light field datasets. This motivates the creation of more robust light field datasets to fully unlock the ability of deep learning techniques, as higher quality, more varied, and larger amounts of data lead to more accurate models.

Researchers have turned to machine learning algorithms to replace past manual calculation and human-reliant approaches to solve various problems. For example, scientists used to manually sort through large amounts of data, but with a machine learning model, scientists can solve tasks such as image classification or object detection faster and more accurately. Therefore, in light field applications, machine learning algorithms have recently been more widely used. For example, in 2019, Pei et al. developed a deep neural network to estimate whether a single synthetic aperture image is in focus for the task of synthetic aperture imaging, a technique where multiple viewpoints of light fields are used to simulate a large aperture camera with a large virtual convex lens [2]. Traditionally, Convolutional Neural Networks (CNNs), due to their robust ability to learn visual features from pictures, are trained on image sets, most notably ImageNet [9], to perform image identification and classification tasks; however, more recently, CNNs have also been used to estimate depths of light fields from light field datasets, as these machine-dependent algorithms can achieve faster and more accurate light field depth estimation results than traditional methods, such as correspondence matching between views and depth from defocus with synthetic aperture photography. For example, Shin et al. [1] developed a depth estimation CNN called EPINET that achieved top rank in the HCI 4D Light Field Benchmark [10] on assessment metrics such as bad pixel ratio, mean square error, etc. The EPINET design takes in four processing streams from four directions (vertical, horizontal, right diagonal, left diagonal) of sub-aperture pictures as input and outputs four independent light field representations. Then, Shin et al. combined these feature maps to produce singular and higher-level representations to estimate depth. However, the HCI synthetic light field dataset only has 28 scenes in total and the EPINET’s network architecture has a small receptive field, meaning it can only handle a limited spacing between cameras, making its state-of-the-art (SOTA) performance limited and unrepresentative of performance on real-world light field data [11].

To address the lack of data surrounding light fields, Shin et al. [1] also used data augmentations, or modifications, on light field image data. Then, the modified data are added to an existing dataset of the same medium to expand the dataset. Because there is not enough variety of publicly available light field data and machine learning algorithms are usually highly dependent on the amount and variety of data to be trained accurately, Shin et al. applied augmentations such as scaling, rotations, transpositions, etc. to increase the amount of data in existing sparse light field datasets, such as the HCI dataset. In the end, using augmentations helped increase the depth estimation accuracy of the EPINET algorithm. However, augmentations result in a limited increase of data, as only crops, rotations, and other modifications of the same scenes are added to an existing dataset. Hence, this process leads to a limited increase in accuracy.

Others attempted dataset creation from scratch to increase the amount and diversity of light field data. Xiong et al. [12] captured three discrete 5D hyperspectral light field scenes (represented by f(x,y,u,v,λ)) with a special hyperspectral camera. Unfortunately, this dataset is limited in size and does not have disparity labels; therefore, it cannot be used for evaluating the performance of deep learning algorithms. In addition, it takes 16 hours for their MATLAB code to reconstruct a single hyperspectral light field. Further, although their hybrid imager hardware system, including a Lytro camera and a coded aperture snapshot spectral imager, recovers 5D light fields with both high spectral and angular resolutions, their expensive and time-consuming process deems it impractical for others to reproduce. On the other hand, Schambake et al. [13] created a 507-scene multispectral light field dataset, where each light field is represented by L(u, v, s, t, λ), using a designed scene generator to randomly output images as well as adding seven handcrafted scenes. Though their dataset is larger than Xiong et al.’s hyperspectral light field dataset, their multispectral dataset creation approach is complex, especially the manual handcrafting of the seven scenes. Besides, most cameras, especially most light field cameras, are not multispectral, narrowing the practical applications of using a multispectral dataset.

III. MATERIALS AND METHODOLOGY

In this paper, we present a novel approach to create more robust light field datasets to avoid the shortcomings present in previous augmentation and creation methods and to assist learning-based methods, such as the ones mentioned above, as they are shown to outperform manual optimization-based methods for calculating light field tasks. We plan to generate different RGB (red, blue, and green) proportions of synthetic light field datasets from scratch. Our approach is easy to create, set up, and modify using a Unity engine and covers a wide range of different hardware and camera parameters that can easily be changed for any light field tasks and applications, such as the three mentioned above. In addition, the datasets created by our approach are in the RGB color format, which is significantly
easier to generate and covers a wider range of applications than hyperspectral and multispectral versions, as the images captured by most cameras, including mobile, handheld, and most significantly, plenoptic or light field cameras, are in RGB format. Furthermore, light field images in our datasets can be generated into any file type, e.g., png, jpg, etc. after capture by just changing one or more parameters in the C# Script, which means they are scalable for many different tasks and data requirements. Our goal is to offer a practical, convenient, and robust solution to deep learning and light field research by creating large datasets that are reliable as well as easily scalable for numerous light field applications and a wide range of deep learning algorithms. Therefore, in this paper, we present a Unity-based approach toward increasing the diversity of light field datasets to enable the creation of more machine learning algorithms that are both speed- and accuracy-efficient for a variety of applications.

First, we discuss the software and materials used to generate our datasets. To create custom light field datasets in 3D, we utilize the capabilities of Unity (see Fig. 2), a cross-platform game engine that provides a variety of virtual scenes, backgrounds, prefabs, assets, and objects. With this engine, we conveniently set up a full scene with directional lighting and terrain as the background for our light field images. After environment setup, we develop the code to tune specific hardware parameters, spawn random objects into the empty terrain space, create and position multiple cameras to view this space, and take automatic snapshots of the entire scene. The code is written in the form of C# scripts that are embedded within the Unity project in Unity Hub 3.2.0. and can be found in our public GitHub repository here [14]. We use the Unity Editor Version of 2020.3.36f1 to leverage the objects, functions, etc. that exist in and are compatible with older Unity versions. Based on our Unity approach, we have generated and stored six of our datasets in .png format of a large variety of sizes (18, 18, 40, 100, 200, and 500 images) into a public Kaggle dataset for others to use [15]. We selected Kaggle because it is one of the top platforms with Jupyter notebook environments for deep learning researchers and engineers alike to utilize datasets to develop novel machine learning architectures. Our Kaggle datasets can be easily used either in the same platform or downloaded to a computer and loaded into another IDE (code development tool) with a Jupyter notebook for model training.

Next, we discuss the process to create our custom Unity dataset. A 3D graphics pipeline (see the top sequence diagram in Fig. 3), which includes view transforms, is the starting point we use to create our light field dataset [16] [17] (Nanyang Technological University, 2012; Wetzstein). We specifically modify the Vertex Processor, one part of the graphics pipeline described above, to create light fields (see the bottom sequence in Fig. 3). We adjust both the model and view transforms of this step. We use random model transforms to create randomized scenes for an infinitely large dataset of scenes, setting up fields of view and the spawning of random objects. We create functions in our C# script to automatically generate random objects of different shapes, sizes, and textures at different positions. Additionally, we use multiple view transforms to generate a light field given a particular scene by modifying the positions of and adding multiple cameras in our Unity engine setup so they can take snapshots of different scenes from multiple angles. We create functions in our C# script that set up a specified number of cameras at different positions and take automatic snapshots of the scene, which can be easily modified by adding various textures, objects, and backgrounds. Combining the model and view transforms, we can develop an automated Unity process that generates full light field datasets within minutes.

With the end goal of generating more light field data to train robust machine learning models, we design an end-to-end algorithm using Unity graphics and virtual scenes to generate a dataset of random images and depths of random scenes relative to any number of cameras. As mentioned above, our algorithm generates the random positioning of objects in the Unity scene and allows for adding multiple virtual Unity cameras for multiple view transforms to capture a full light field. By taking quick snapshots from slightly different angles, we can create a large dataset containing a high diversity of light field snapshots to train a deep learning architecture. We create a program using C# scripts to automatically generate objects at random positions within set ranges and boundaries that fit within the rectangular scene boundaries and automate snapshots of these scenes. Next, the cameras, built within Unity, capture a snapshot with a specified Width x Height resolution (which can be modified in our Unity program) from each sampled location on the UV plane. Then, our program deletes all the objects from the scene.
regenerates new random objects at new random positions, takes snapshots of this new scene from different angles, and the entire process repeats. At the end of the multi-step process, all the snapshots are automatically stored in a Snapshots folder.

Our Unity-based approach is much faster, more convenient, scalable, reproducible, and robust than all previous methods for the reasons below. The speed of data generation is only limited by the processing speed of the computer running our Unity engine. Using a Windows 10 desktop with a 3.60GHz CPU and 1080 Ti GPU (more details of our computer hardware specifications are listed in Fig. 5), spawning up to one thousand objects takes only about four seconds or less, and creating a 2,000 light field image dataset with 40 objects in each snapshot takes less than five and a half minutes due to our automated random object spawner script that can generate new data synthetically from any angle in the scene and take a snapshot of each light field image or scene. We also have the flexibility of creating a dataset of any size for any situation by modifying the number of images variable in Unity. We can easily generate very large light field datasets (up to 2,000 snapshots within our RAM limits) by just adjusting several parameter variables, such as producing more than a thousand different scenes for a dataset, outperforming all three datasets mentioned above in terms of both size and convenience - the 28-scene HCI dataset [10], 5D hyperspectral dataset [12], and 507-scene multispectral dataset [13]. In addition, we can easily increase the number of viewpoints by simply updating parameters in the same Unity scene to match any new hardware, such as another plenoptic camera. Our Unity setup also allows for the tuning and modifications of numerous parameters, including the set number of cameras, number of objects, types of objects, textures of objects, etc. to increase the diversity of produced data. Also, because we have different random scenes for each data point, we ensure no point of similarity between any data points, therefore outperforming augmentation-based methods and Schamback et al.’s two-camera system for their dataset in terms of data variety. As mentioned before, data augmentations only utilize different modifications on clones of the same scene on a dataset and are therefore not as diverse as our approach. In addition, we choose to generate RGB light fields, where their three-channel colors replace light fields’ coordinates and spectral dependencies [13], because RGB light fields are computationally more efficient to generate and are more widely applicable for light field tasks than multispectral light fields, as most cameras use RGB formats. Therefore, our approach also outperforms Schamback et al.’s multispectral dataset by a second assessment metric [13]. Furthermore, our Unity-based approach is flexible, as its data generation parameters can easily be changed for different hardware setups and match specific hardware parameter requirements, such as cameras’ resolutions and positions. For Xiong et al.’s hyperspectral dataset, they included different light field images, but their datasets were only made for a specific hardware setup. To make their datasets suitable for different hardware setups, their datasets must be changed or regenerated to match that specific hardware setup, which is cumbersome as they need to use both a Lytro camera and an imager to recover light fields. For our method, the parameters of our snapshots and locations on the UV plane can be updated easily in Unity to match specified hardware parameters, such as the number and position of cameras, etc. Our Unity scenes simulate the occlusions, reflections, and diffractions of light present in real-world scenes, which allows us to quickly generate accurate light field datasets without noise, expensive hardware setups such as plenoptic cameras or spectral imagers, or human intervention. The only materials one would need to generate a light field dataset are a Unity3D engine and Unity Hub/Editor installed on a desktop or laptop and our publicly available code, which takes just minutes to create a light field image dataset of any size. Most notably, we developed our program in C# scripts to automate the generation of objects and capturing of scenes, which requires no manual work other than changing variables and parameters. Therefore, our approach is affordable, customizable, convenient, and reproducible.

IV. RESULTS

Our Unity program outputs a set number of images with multiple cameras at different x and y coordinate positions. As discussed in the previous section, our C# program allows parameters on the cameras to be customized, such as the number of images to capture, the number of translations the cameras make, and the distance the cameras move. Similarly, in the viewpoint of the scene, every parameter within the scene can be customized to fit specific simulation specifications. Fig. 4 displays one of our datasets.

![A Light Field Dataset Generated from Our Unity Approach](image)

Fig. 4. Different x and y coordinate views of randomly generated objects. From left to right the cameras were translated right and from top to bottom they were translated down.

The amount, size, and quantity of objects spawned can be adjusted to simulate real-life measurements and quantities and are only limited by hardware (our specific computer hardware parameters are listed in the caption of Fig. 5). Our program is lightweight, computationally efficient, and CPU-efficient where a substantial number of objects can be spawned at once. Up to 1,000 generated objects in a single scene are tested and proven to be fast and stable with a variety of Unity hardware and variable parameter specifications, and up to 2,000 light field images for a dataset can be generated within five and a half minutes (see Fig. 5 for a comparison of data creation times).
In this paper, we presented a novel approach for generating customizable light field image datasets that is quick, easy, customizable, and robust for machine learning. Deep learning-based architectures, such as EPINET, have been shown to be more robust, accurate, and faster than light field cameras and optimization-based techniques for many light field tasks.

Overall, our custom datasets can be used for machine learning models in an expansive variety of light field applications, notably synthetic aperture photography, depth estimation, 3D representations, and more.

VI. LIMITATIONS AND FUTURE DIRECTIONS

Because Unity provides only virtual scenes, all our datasets are synthetic and without noise, which is unrepresentative of the real world, because most real-life cameras and images may include noise. Hence, in the future, we plan to simulate real-world noise by adding Unity3D’s built-in Cinemachine Noise Properties, such as Perlin noise, which helps simulate random movements in our virtual cameras [18]. Thanks to the promise of machine learning as a robust and accurate mechanism for numerous applications, specifically for light field tasks, our novel Unity-based approach can easily be used as a reproducible mechanism to automatically create large, variegated datasets to train any machine learning model to maximize its efficiency and accuracy. For future research, we plan to develop a novel machine learning model ourselves, such as a convolutional neural network or autoencoder, to test the robustness and usability of our Unity-generated light field datasets.

ACKNOWLEDGMENT

We would like to thank the SHTEM Summer Research Internship Program managed by the Stanford University Compression Forum and its organizers Cindy Nguyen, Sylvia Chin, Carrie Lei, Eric Guo, and Kaiser Williams for providing us with amazing opportunities to participate in scientific research. Specifically, we would like to thank our mentor Manu Gopakumar, an Electrical Engineering Ph.D. student from Stanford University, for his guidance throughout our research.

REFERENCES

[1] C. Shin, H. G. Jeon, Y. Yoon, J. S. Kwon, and S. J. Kim, “Epinet: A fully-convolutional neural network using epipolar geometry for depth from light field images,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4748-4757), 2018.
[2] Z. Pei, L. Huang, Y. Zhang, M. Ma, Y. Peng and Y.-H. Yang, "Focus Measure for Synthetic Aperture Imaging Using a Deep Convolutional Network," in IEEE Access, vol. 7, pp. 19762-19774, 2019, doi: 10.1109/ACCESS.2019.2896655.
[3] M. Levoy and P. Hanrahan, “Light field rendering.” In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (pp. 31-42), August 1996.
[4] M. Levoy, “Light Fields and Computational Imaging,” in Computer, vol. 39, no. 8, pp. 46-55, Aug. 2006, doi: 10.1109/MC.2006.270.
[5] M. W. Tao, S. Hadap, J. Malik, and R. Ramamoorthi, “Depth from combining defocus and correspondence using light field cameras.” In Proceedings of International Conference on Computer Vision (ICCV), 2013.
[6] W. Huang and Z. Jing, “Evaluation of focus measures in multi-focus image fusion,” Pattern Recognit. Lett., vol. 28, no. 4, pp. 493–500, 2007.
[7] M. Kristan, J. Per, M. Pere, and S. Kovai, “A bayes-spectral-entropy based measure of camera focus using a discrete cosine transform,” Pattern Recognit., vol. 27, no. 13, pp. 1431–1439, 2006.
[8] J. Tian and L. Chen, “Adaptive multi-focus image fusion using a waveletbased statistical sharpness measure,” Signal Process., vol. 92, no. 9, pp. 2137–2146, 2012.
[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database.” In 2009 IEEE conference on computer vision and pattern recognition (pp. 248–255), 2009.
[10] K. Honauer, O. Johannsen, D. Kondermann, and B. Goldluecke, “A dataset and evaluation methodology for depth estimation on 4d light fields.” In Proceedings of Asian Conference on Computer Vision (ACCV), 2016.
[11] T. Leistner, R. Mackowiak, L. Ardizzone, U. Köthe, and C. Rother, “Towards Multimodal Depth Estimation from Light Fields.” In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 12953-12961), 2022.
[12] Z. Xiong, L. Wang, H. Li, D. Liu, and F. Wu, “Snapshot hyperspectral light field imaging,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jul. 2017, pp. 3270–3278.
[13] M. Schambach and M. Heizmann, October 15, 2020, “A Multispectral Light Field Dataset for Light Field Deep Learning”, IEEE Dataport, doi: https://dx.doi.org/10.21227/y90t-xk47.
[14] https://github.com/TheClassicTechno/LightFieldDatasetGenerator
[15] https://www.kaggle.com/datasets/julesh7/rgb-light-field-dataset
[16] “3D Graphics with OpenGL - The Basic Theory.” NTU.edu. https://www3.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html (accessed Jul. 22, 2022).
[17] G. Wetzstein. “The Graphics Pipeline and OpenGL I: Transformations. EE267.” Stanford.edu. https://stanford.edu/class/ee267/lectures/lecture2.pdf (accessed Jul. 22, 2022).
[18] “Noise properties: Cinemachine: 2.8.6. Cinemachine | 2.8.6.” Unity.com. https://docs.unity.cn/Packages/com.unity.cinemachine@2.8.6/manual/CinemachineVirtualCameraNoise.html (accessed Jul. 31, 2022).

| Dataset Size, in number of snapshots | Dataset Creation Time, in minutes and seconds |
|--------------------------------------|-----------------------------------------------|
| 100 | 16 seconds |
| 200 | 31 seconds |
| 500 | 79 seconds (1 minute 19 seconds) |
| 1,000 | 159 seconds (2 minutes 39 seconds) |
| 2,000 | 325 seconds (5 minutes 25 seconds) |

Fig. 5. Table showing the speed of our dataset generation approach based on several dataset sizes. Each snapshot of a scene has 40 objects. The quick speed ensures the reproducibility and convenience of our method in generating new light field datasets, which is much faster than Xiong et. al’s method, which took 16 hours. Our results in this table are generated by our Unity program running on a Windows 10 desktop, Version 10.0.19044 and Build 19044 with an Intel Core i9-9900K @ 3.60GHZ CPU, NVIDIA GeForce GTX 1080 Ti GPU, and 32.0 GB @ 3200 MHz RAM.