3D Reconstruction Framework for Development of Robots and IoT for Precision Agriculture

Brian Park, Andrew Choi, Shivam Kumar Panda, Khalid M. Jawed, Ph.D., Jungseock Joo, PhD., Sriram Narasimhan, PhD.
Structures-Computer Interaction Laboratory, Mechanical and Aerospace Engineering Department, UC Los Angeles, CA 90095

Introduction

- In agricultural robotics, there is no complete testing framework for robots and sensors used in precision agriculture
- It is expensive to test our physical robot in the outside world, especially in its earlier stages of development
- Therefore, we require a virtual testing environment that includes a 3D, high-resolution rendering of crops, weeds, and soil to simulate our robot's sensors and analyze its behavior
- In order to optimize the quality of these 3D renderings, we will develop an active learning model that is trained to discover the view angles of the object that require the most additional information.

Objectives

1. Develop an image sampling Unity program to obtain various snapshots of a 3D Object and the transform structure of the camera poses, such that we are able to input these images/transforms into Instant-NGP
2. Write a script to automate the sampling pipeline, which will ultimately output a mesh object file of the resulting NeRF.

Methodology

Objective 1: Unity Sampling Pipeline

Our goal with this objective is to establish a proper 3D environment, such that our simulated camera is able to capture images of a 3D object. With these images, we will use them to train and build a NeRF using Instant NGP.

Initially, our environment consisted solely of the 3D object, floating in empty space. This led to problems in constructing the NeRF. Since there was no concrete platform/boundaries that defined the object's location, the resulting NeRF had lots of noise and incorrect structure.

The biggest challenge to this part of the project was aligning the Unity transform structure to the expected structure of Instant NGP. To do this, we performed a series of matrix operations. After obtaining the raw positions and quaternions data from the Unity script, we used NumPy to change the quaternion format from [x y z w] to [-x –z –y w]. We then converted this quaternion into a rotation matrix using Scipy, and appended it to the corresponding position array to obtain our final transformation matrix.

Objective 2: NeRF Automation Script

The goal for this object is to streamline our Unity sampling process. Originally, the user had to open up the Unity engine and start the script themselves. Once the script finished running, the user had to manually copy over all of the files and build the NeRF using the Instant NGP GUI.

With so many steps to this process, it is inconvenient to the user. Our solution was to create a script that automates every step of this process. Using bash scripting, we can string together a list of commands that starts the Unity sampling phase, as well as automatically copies over all the images and camera information to the correct Instant NGP directory.

Once the Instant NGP part was set up, we can use Instant NGP's Python API to train and build a NeRF without using their GUI. By passing in the correct arguments to the given python script, we can create a mesh object file that represents our resulting NeRF.

Results: Unity Sampling Pipeline

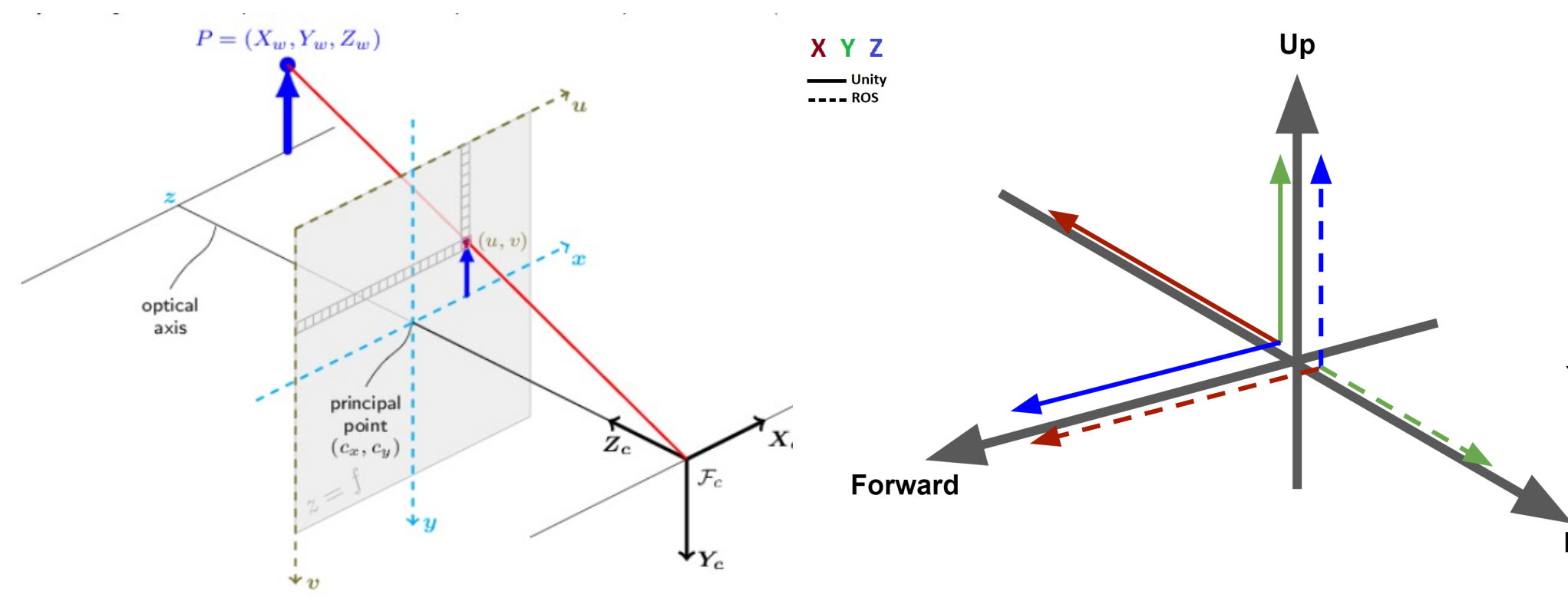

Figure 1: Instant-NGP (left) expects a left-hand, Y-down coordinate system. Unity (right) expects a left-handed, Y-up coordinate system.

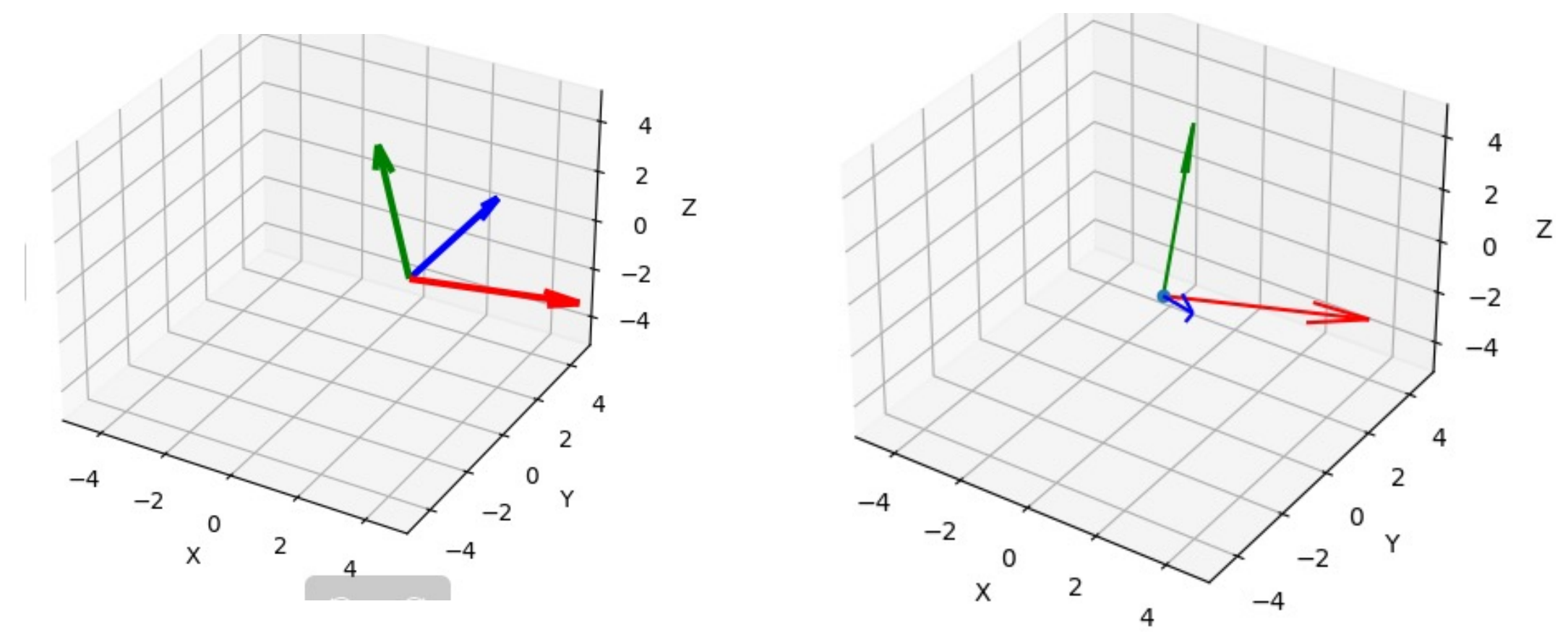
Figure 2: Transform structures of the original Unity camera. (left), compared to the transform structure that was obtained using the conversion method listed previously (right).

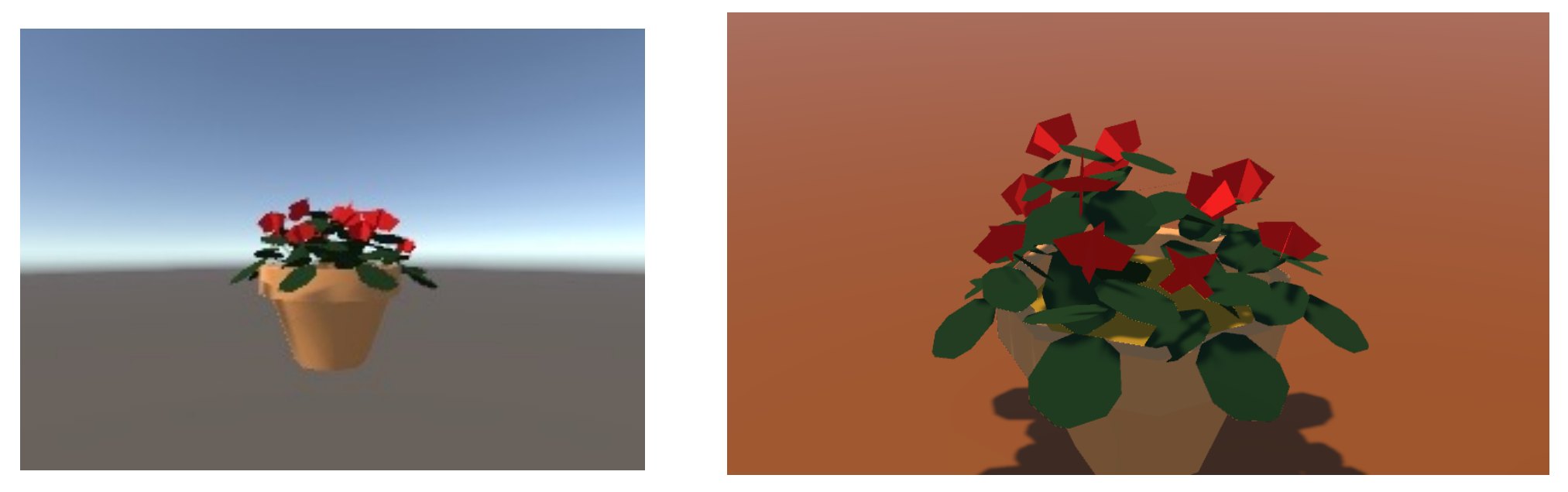
Figure 3: Images showing the old Unity environment (left), compared to the new Unity environment (right) with a platform.

Results: NeRF Automation Script

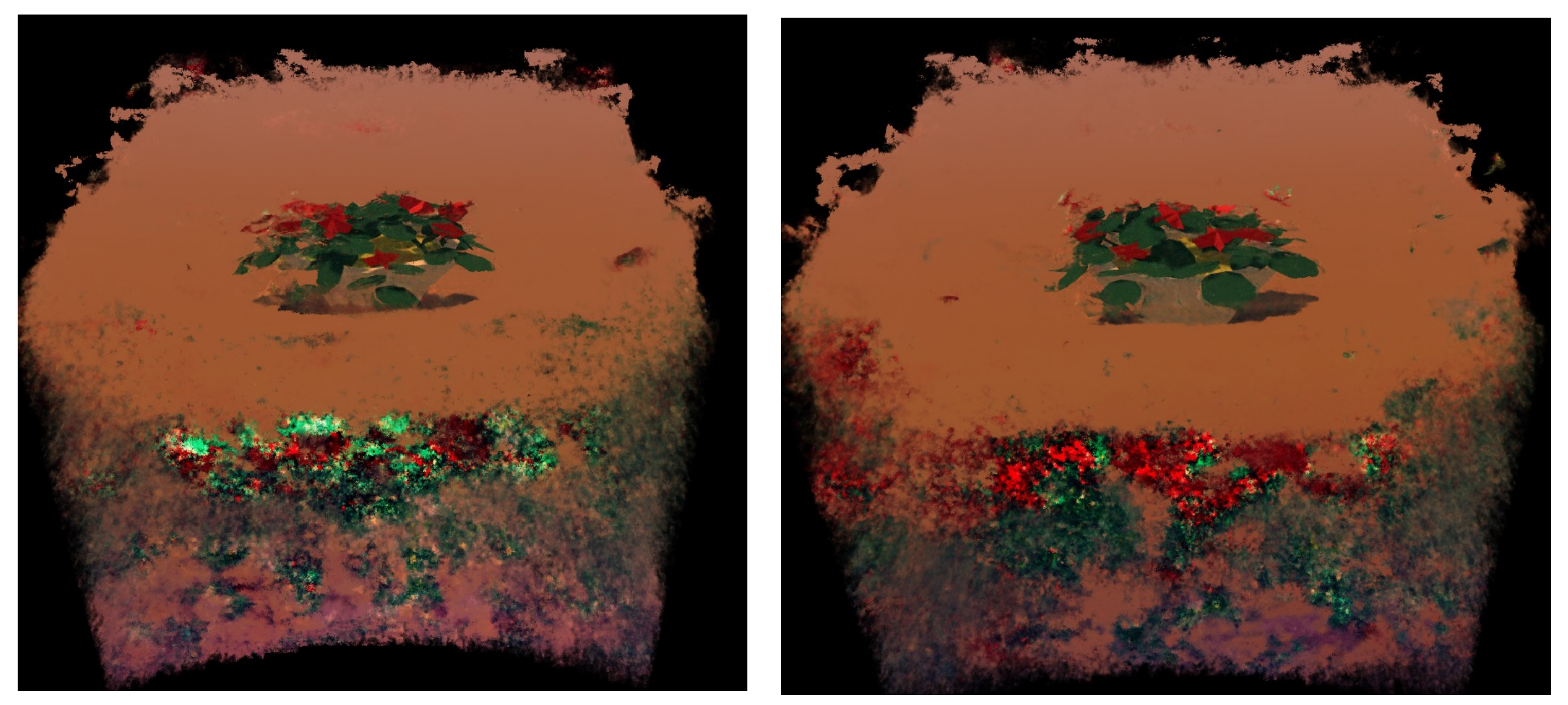
Figure 4: Resulting NeRF that was obtained by training in the Instant NGP GUI.

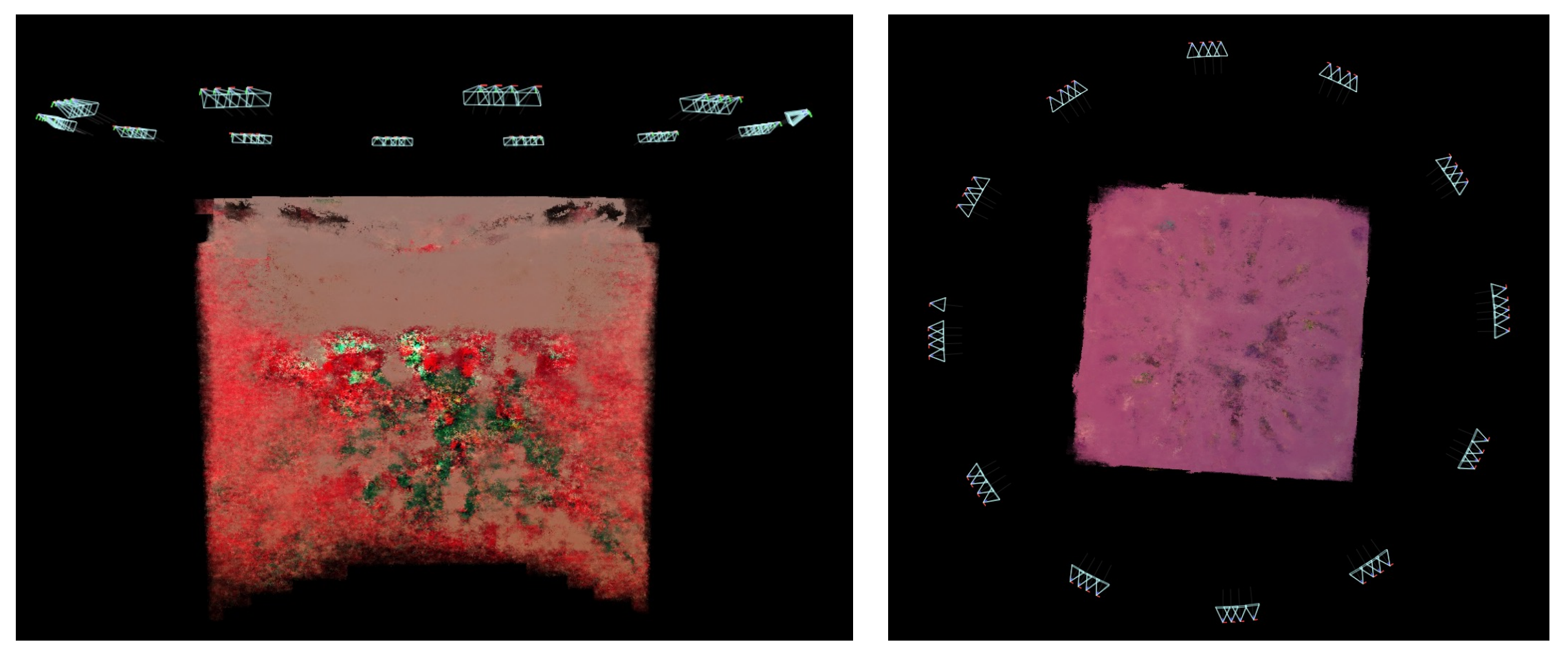
Figure 5: Examples of viewing the NeRF from "bad" view angles (i.e. angles that have insufficient data). Camera angles are shown with the white icons.

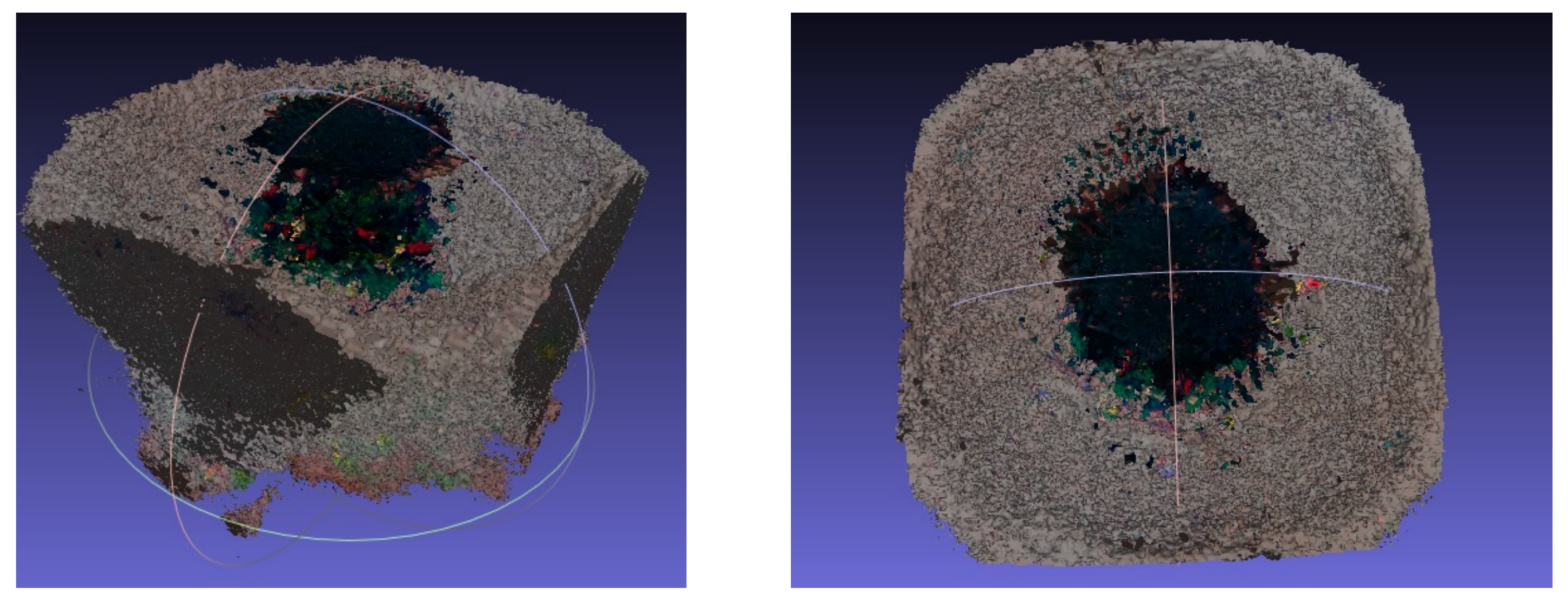
Figure 6: Resulting mesh object of calling Instant NGP's Python API in our NeRF Automation script, rendered using MeshLab

Conclusion

Objective 1: Unity Sampling Pipeline

From the results, we can conclude that we were able to successfully build an accurate, photorealistic NeRF of a simulated flower pot, as well as other 3D objects from the ShapeNet dataset. Additionally, Figure 2 shows that the matrix conversion method we applied to the Unity matrix is now aligned with the transform structure that Instant NGP expects.

We added some further optimizations to improve the efficiency of this Unity sampling pipeline. We were able to decrease the total time taken to sample images from 2-3 minutes down to 1-2 seconds. We achieved this change by teleporting the camera's position every frame, such that the camera moved along a circular path. In contrast, our previous implementation manually moved the camera around the object in a continuous motion, which slowed down the sampling process drastically.

Objective 2: Automation Script

From Figure 5, we are able to reinforce our belief that viewing a NeRF from an angle that doesn't have enough training data will result in a poor rendering. In this example, the camera icons show that we only sampled images from a diagonal angle. This means that when we view the NeRF from a side view or a top-down angle, we will see a lot of noise, and the actual 3D object will not be rendered. This observation helps reinforce our motivation for building the RL Active Learning model.

Future Works

1. Modify the Unity camera path, such that its path follows a half-spherical shape. This allows us to sample from a greater variety of view angles.
2. Further implement the automate.sh script to allow the user to build/start the Unity environment.

Literature Cited

Pan, Xuran., Lai, Zihang., Song, Shiji., Huang, Gao., "ActiveNERF: Learning Where To See With Uncertainty Estimation.", EECV 2022.
Mueller, Thomas., Evans, Alex., Scied Christopher., Keller, Alexander., "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding"., SISGRAPH 2022.

Acknowledgements

A special thanks to Professor Khalid Jawed, Andrew Choi, and the Structures-Computer Interaction Laboratory at the University of California, Los Angeles.

This project was sponsored by the NSF REU program (Award No. 1925360).