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 reality 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. Research various 3D Object datasets and decide which one to use for training our active RL agent.
2. Develop a sampling method using Unity Engine to obtain various snapshots of a 3D Object and the transform structure of the camera poses, such that we are able to input this into Instant-NGP

Methodology

<u>Objective 1: 3D Object Datasets</u>

Five datasets were considered in choosing our final dataset:

1. ShapeNet
2. ModelNet40
3. OmniObject3D
4. Toys4K
5. Scanned Objects by Google Research (GSO)

For each dataset, we were able to obtain 3 samples and view example renderings in MeshLab. We also gathered various qualities of the dataset, including size and number of different categories.

The rendering quality and size of the datasets were key metrics in determining which dataset to use.

After determining which dataset to use, we also had to test different simulation frameworks to see which one we wanted to use to implement our sampling code. To determine this, we rendered various meshes into Unity Engine and Gazebo, and compared their rendering qualities.

<u>Objective 2: Unity Sampling Method</u>

As shown in the results, Unity proved to be the better option for the simulation framework. In Unity, we take the CameraObject and a GameObject, which is a 3D rendering of a plant from the ShapeNet dataset. We then rotate the camera around the object, in order to ensure that we sample images from various viewpoints.

After obtaining these images, we must also convert the Unity's transform structure into the convention that Instant-NGP expects.

Since Unity uses a left-handed Y-up coordinate system and Instant NGP expects a left-handed Y-down coordinate system, we have to rotate our transform matrix 180 degrees about the x-axis. After obtaining the 4x4 transform matrix for each frame, we can compile them all into a list data structure.

Results: 3D Object Datasets

Figure 1: Renderings of ShapeNet's 3D objects using Unity (left) and Gazebo (right)

Name	# Objects	# Classes	Terms/Conditions
ShapeNet	51,000	55	Need to cite paper
ModelNet40	12,311	40	Need to cite paper
OmniObject3D	6,000	190	Currently not publicly available
Toys4K	4,179	105	Need to request access to data
Google Scanned Objects	1,000	17	Publicly available

Table 1: Summarizes the characteristics of the different 3D Object Datasets

Results: Unity Sampling Method

Figure 2: Screenshots from various viewpoints obtained from the Unity Sampling Method

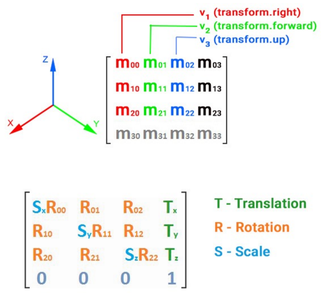

Figure 3: Transform matrices are represented by 4x4 matrices that contain information about the translation and rotation

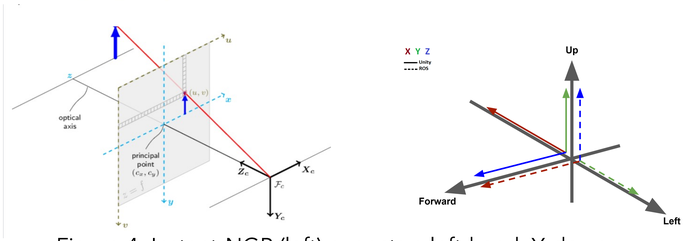

Figure 4: Instant-NGP (left) expects a left-hand, Y-down coordinate system. Unity (right) expects a left-handed, Y-up coordinate system.

Conclusion

<u>3D Object Datasets</u>

- Given its size, diversity of categories, and the rendering quality, we chose ShapeNet as the dataset to train our Active RL Agent
- ShapeNet contains 51,000 different objects, has 55 classes, and is available for public use
- Based on Figure 1, it is clear that the renderings that Gazebo gives produce sharp, jagged edges, which would produce inaccurate renderings

<u>Unity Sampling Code</u>

- So far this quarter, we were able to write a Unity C# script to control the Camera, such that it rotates around the 3D object along multiple axes
- As shown in Figure 4, the conversion from Unity's transform structure requires a 180 degree rotation about the x-axis
- We were able to use Unity API's cameraToWorldMatrix() call to obtain the camera's transform structure.
- We then multiply Unity's 4x4 matrix by the matrix shown in Figure 5, in order to obtain the transform matrix in Instant NGP's convention

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 5: The constant rotation matrix that represents a 180 degree rotation about the x-axis

Future Works

1. Research into Unity's Scripting API to obtain the Unity Camera's physical properties, including focal length, image sharpness, image width/height, etc.
2. Automate the sampling process to apply it to every other 3D Object in the ShapeNet dataset.

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).

NSF UCLA STRUCTURES-COMPUTER INTERACTION LABORATORY