Neural Radiance Fields (NeRF) for 3D Visualization Rendering Based on 2D Images

Joann Cyriac
Westlake High School
joann.cyriac@gmail.com

Ivan Rodriguez
Brown University
ivan_felipe_rodriguez@brown.edu

INTRODUCTION

We are living in a 3D world, but oftentimes we visualize things in 2D scopes like in pictures and videos of different objects. The world of 3D visualization has been constantly evolving over the past few years, still growing with the introduction of cutting-edge computer vision and deep learning methods. Reconstructing 3D scenes and being able to interact with them in virtual environments used to be something beyond our capabilities but now with the use of Neural Radiance Fields (NeRF) and Instant Neural Graphic Primitives (InstantNGP) we are able to create a world in a virtual space, opening the gateways to applications in object detection and generation, augmented reality, medical imaging, and more. Access to these methods will allow for more immersive development by creators, hands-on educational resources, and more experimentation and research, stimulating future applications and techniques.

PROCEDURES AND RESEARCH

NeRFs are neural network models with radiance fields that essentially use 2D images taken from multiple angles and viewpoints and create highly detailed 3D scenes. Learning from neural networks, NeRF models have the ability to map 3D spatial coordinates as a continuous

function by combining 2D images of a particular scene. They use volumetric rendering, simulations of interactions of light with the object, to output the color and opacity of each of the 3D volume pixels. After taking in the data consisting of the set of 2D images, the NeRF models trace rays from the camera and compute the color and detail of each pixel, allowing for viewing of the scene from any desired viewpoint.

Figure 1: Various camera angles

My interest was first in object detection, and we started doing research on computer vision and how it is being used right now. We found out about the Amazon Picking Challenge, and how robots utilized different methods of object detection, reconstructing the object virtually to understand its interactions, which is what eventually led us to NeRFs. The first experiment we ran was to fit a volume with raymarching, using sample data provided by the authors of the code. (https://github.com/facebookresearch/pytorch3d/blob/main/docs/tutorials/fit_textured_volume.ipynb). Importing it into a Google Colab Notebook, we ran through the detailed steps to render a

silhouette and an image of their sample data, a small cow. (Figure 2). In the notebook we were able to establish a differentiable volumetric renderer, create a volume to encompass the object with little space left based on the images, and visualize the volume that was predicted. Raymarching is a graphics technique that traces rays in a scene and "marches" through them to find intersections of objects and surfaces. It allows for more realistic renders and for more detail.

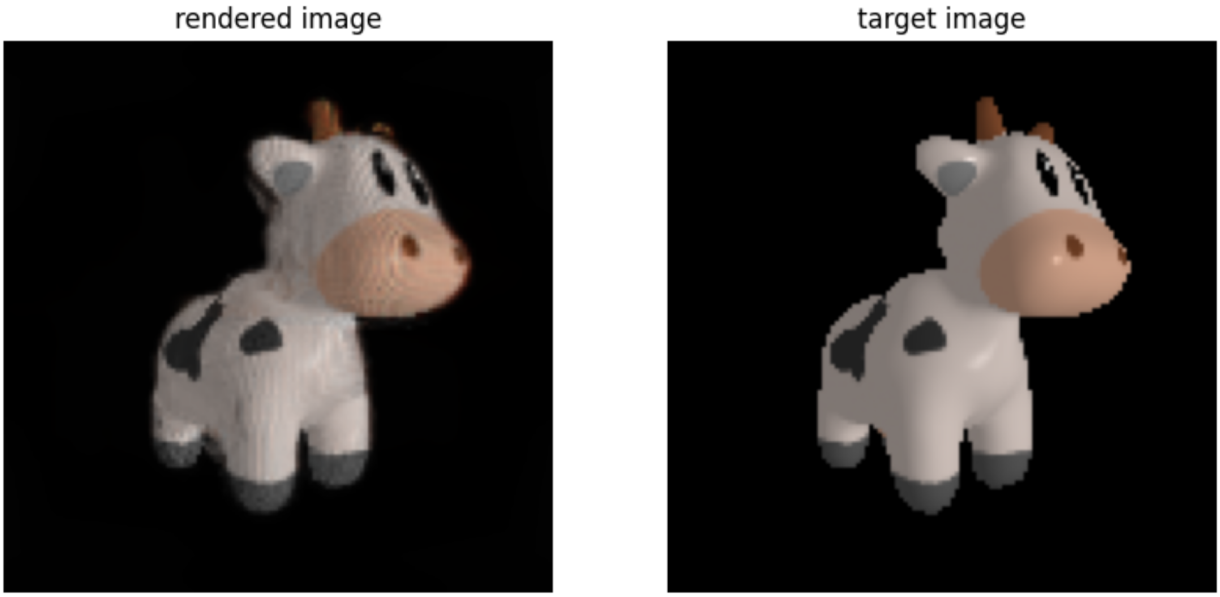

Figure 2: Sample data used to fit a volume via raymarching

The next experiment we tried was a 3D volumetric rendering as well but with NeRFs. (https://keras.io/examples/vision/nerf/) Utilizing pose matrices to capture the camera angles and positions, the authors then use volumetric rendering with ray casting and tracing to map it from a 2D image to a 3D one. This all serves as the input for the NeRF which then tries to predict the color and density, similar to the model previously mentioned. The NeRF uses a multi-layer perceptron (MLP) to identify patterns and relationships. MLPs consist of multiple layers of artificial neurons that each input and compute data which is then computed throughout the following layers. The first layer is the input, the middle layer(s) serve for computation, and the

final layer for the final output. This model also uses Rectified Linear Units (ReLU) to introduce non-linearity, essentially allowing the model to learn more complex patterns or relationships. It is an activation function defined as f(x) = max(0,x), which means if the input is greater than or equal to zero it returns itself, otherwise it returns a 0. This function is applied to these artificial neurons in the MLP to aid in computation. Using image data collected by UC Berkeley for the Amazon Picking Challenge, a challenge designed to test abilities of a robot picking and placing objects in a warehouse using object detection and computer vision, we were able to train the model. (Figure 2 and Figure 3). We resized the images and changed formats to allow it to be compatible with the model inputs. (Figure 4) In the end, the results portrayed the general silhouette of the object in the pictures, but it was not very well detailed or high quality.

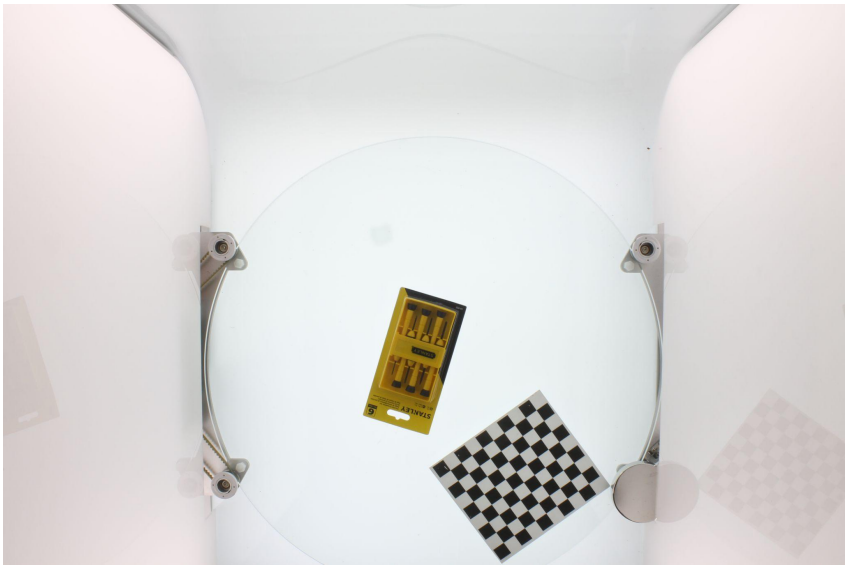

Figure 3: Object used to train model

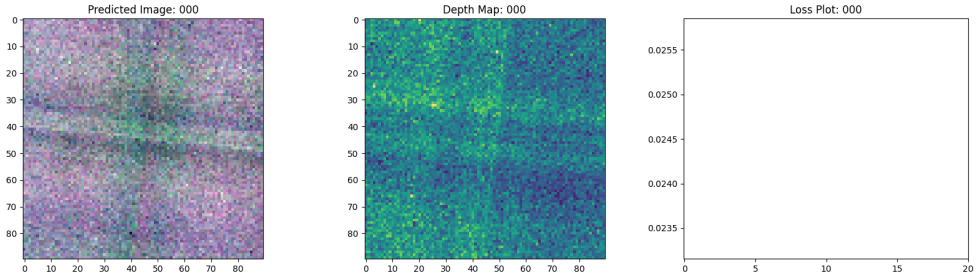

Figure 3: Predicted image for volumetric rendering

```
import cv2
import glob
import h5py
import os
def load_image(f):
  image = cv2.imread(f)
  image  = cv2.resize(image,(90,90))
  image -= image.min()
  image = image/image.max()
  return image

folder = 'stanley_66_052/rgb_highres'
images_files = glob.glob(os.path.join(folder,'*.jpg'))
images_stanly = np.array([load_image(f) for f in images_files ])

pose_folder = 'stanley_66_052/rgb_highres/poses'
poses_stanly =[]
for fn in images_files:
  if '(1)' in fn: continue
  pf = os.path.join(pose_folder, fn.split('/')[-1].replace('N','NP').replace('.jpg','_pose.h5')).
  replace('NP1','NP5').replace('NP2', 'NP5').replace('NP3','NP5').replace('NP4','NP5')
  hf = h5py.File(pf, 'r')
  poses_stanly.append(hf.get('H_table_from_reference_camera')[:,:])
poses_stanly= np.array(poses_stanly)
```

Figure 4: Additional code for resizing and formatting

The last experiment we ran was the Instant Neural Graphic Primitives program (InstantNGP). A very recently published model by NVIDIA employees in July 2022, this program encodes maps for “neural networks inputs into a higher dimensional space”, according to the published paper. Without features such as COLMAP, we followed the google colab to generate a .json file to be inputed directly into the downloaded program, starting the training of the data. Within seconds, the image became clearly defined and highly detailed, allowing user interactions with the scene as well from any angle. The model utilizes multiresolution hash

encoding, which efficiently indexes images with many layers of detail and resolution, making the image sharper and clearer. This model also has downloadable IOS apps to aid in creation of your own datasets. The apps aim to capture and train real time pictures or videos to render a 3D scene within seconds. The model produced very good results, sharper and more detailed than any of the other models we experimented with.

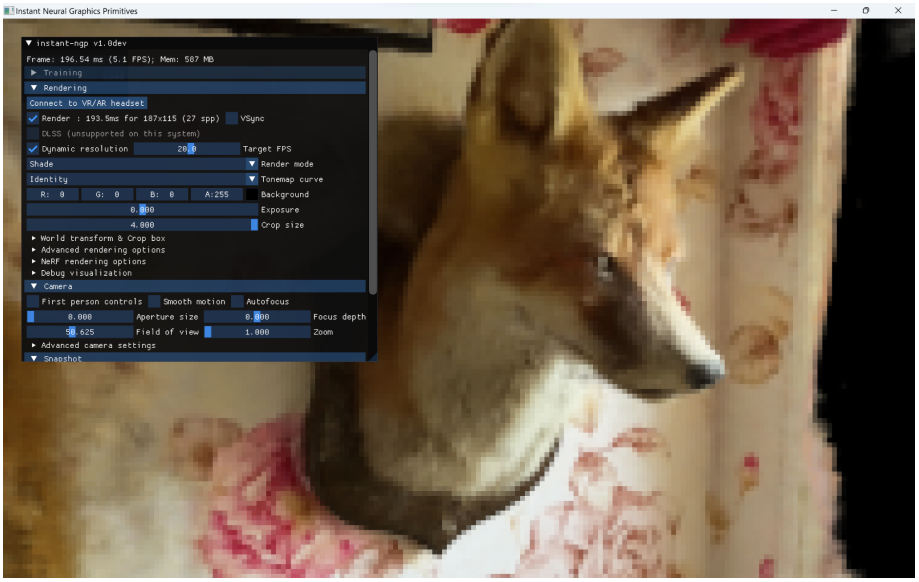

Figure 5: Data sample of fox head

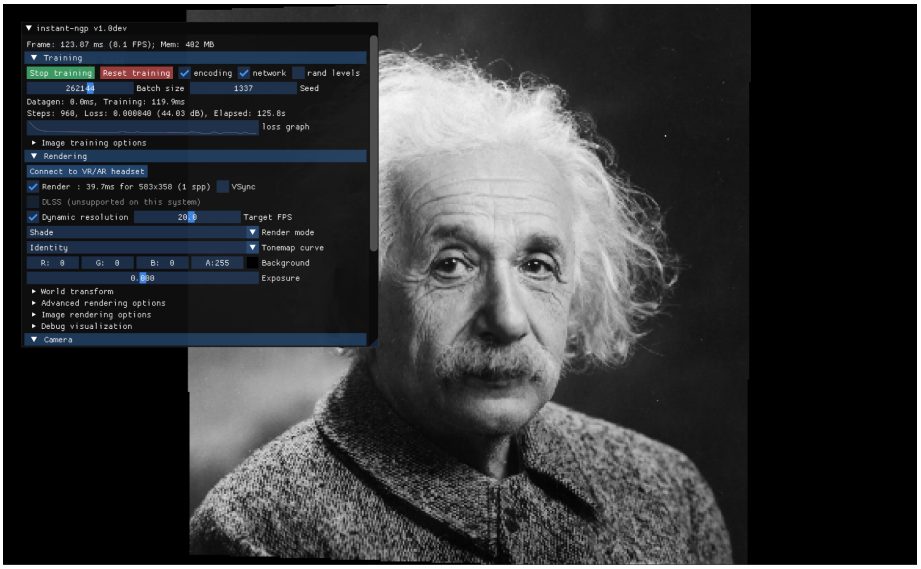

Figure 6: Data sample of Albert Einstein's picture

LIMITATIONS AND FAILURES

With the first model, we were able to render the 3D visualized video of the sample data provided, but it was not easy to implement our own data into the model to be shown. The predictions were successful, but we were unable to train it on new data.

For the second model, we were able to train it on our own dataset from the Amazon Picking Challenge objects, but the figure was very unclear and pixelated.

We concluded that the image dataset was not large enough for the model to be properly trained on without other applications, but was overall successful in its predictions.

Lastly, for the InstantNGP model, it was simpler to follow and generate the files needed from the sample data and it produced high quality results in seconds. We were unable to use COLMAP, the application the authors suggest to be utilized to help improve results, which hindered us from being able to apply our own datasets. We downloaded the IOS apps, but they lacked a user-friendly interface and were hard to understand. There are issues posted with these apps, such as not being able to switch front lens versus back lens, etc. that we also encountered.

CONCLUSIONS

Our world is rapidly expanding and so is our technology. With the rise of 3D visualization and the increasing precision and detail made possible by many computer vision techniques and neural networks such as NeRF, we are able to produce results like never before. We saw results from every experiment we ran, but they varied in quality of their results. Overall, with InstantNGP being the newest application, it proved to yield some of the most hyper-realistic results compared to the rest. But even with our restricted dataset sizes and tools, we were able to get this far, which leaves room for further developments and work. Our hope is for an increase in

accessibility for education and creative outputs, and eventually stimulate future research developments in the field.

References

Gosthipaty, A. R., & Raha, R. (2021, August 9). *3D Volumetric Rendering with NeRF*. Keras. https://keras.io/examples/vision/nerf/

Müller, T., Evans, A., Schied, C., & Keller, A. (n.d.). *Instant Neural Graphics Primitives*. NV Labs. https://nvlabs.github.io/instant-ngp/

Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W. Y., Johnson, J., & Gkioxari, G. (2020). Accelerating 3D Deep Learning with PyTorch3D. *ArXiv:2007.08501*.

Singh, A., Narayan, K., Kehoe, B., Patil, S., Goldberg, K., & Abbeel, P. (n.d.). *Amazon Picking Challenge Object Scans*. Robot Learning Lab, UC Berkeley. https://rll.berkeley.edu/amazon_picking_challenge/