Project: Neural Radiance Fields

1 Project Goal

Multiview 3D reconstruction is divided into 2 stages:

1. Finding the pose of each image. This is the goal of Structure from Motion.
2. Recovering dense 3D model given posed images. Multiview stereo / NeRF apply here.

This project focuses on the 2nd stage, and we will use NeRF [1]. We have another project on SfM. Check it out and decide which one you want to work on. The SfM project might be easier since it's using the existing code we have from homeworks. This project involves more uncertainties.

You must have noticed that our hw3 SfM reconstruction is very sparse i.e. the body and the weapon are there, but very little surface details are discernible even to a trained technician. Compared to the provided ground truth mesh, it just feels empty. To get a numerical sense of this sparsity, note that the images are of size 1600×1200, but for each image fewer than 2000 points are used during triangulation. In other words only about 0.1% of the pixels participate in SfM. This seems wasteful but it's intentional.

The goal of SfM itself is to find camera poses, and the reconstructed 3D points could be thought of as a means to this end. We want to find good poses using the fewest and the most confident, accurate keypoints, hopefully at a small computational cost. Imagine solving eight-point, triangulation and then bundle adjustment using $1000\times$ more points. It will be redundant and slow. The task of extracting dense model is usually left to the 2nd stage.

Multiview stereo / NeRF assumes a collection of posed images as input, and produces dense 3D model. Note that a stereo algorithm usually outputs geometry i.e. dense point cloud or mesh, whereas NeRF is advertised as a novel view synthesis method. The output is a radiance field. This is codeword for saying it doesn't care that much about geometry. Indeed 3D is much richer than just geometry. Lighting effects, materials, textures are all critical aspects of human visual experience. Think about it: arguably 3D is whatever it is that enables view synthesis. Take a look at the NeRF's **gallery** to get a sense of its wonder. Refer to the paper and our classroom slides for the details behind its power.

Training a NeRF even on GPUs can be very slow. Instead we will get pre-trained NeRF models, and play around with it. We recommend these two repos **nerf-pytorch** and **DvGO**. DvGO [2] is a well-maintained fork on nerf acceleration. Both provide PyTorch checkpoints of fully trained models.

Projects are more open-ended and less guided than homeworks. Expect some rough edges. We will keep posting updated resources on canvas as issues arrive. The final report is evaluated more on efforts than a certain outcome. The spirit is to enjoy the creative possibilities of this technology.

2 Steps Outline

1. Check the world coordinate convention and camera coordinate convention used in NeRF. It is different from what we have used for homeworks. Write the code for conversion between these two sets of coordinate systems.
2. NeRF deals with two types of scenes. Inward-facing scenes: lego bulldozers, ships, chairs fall into this category. Forward-facing scenes: fern, dinasour horns, flowers and so on. Forward-facing scenes are parameterized in Normalized Device Coordinates (NDC). NDC is a **3D** homography that warps the space in such a way that on z-axis a lot more precision is put on nearby objects. NDC generalizes camera projection from 3D $\rightarrow$ 2D to 3D $\rightarrow$ 3D.
3. Notice that forward facing scenes are typically visualized within a very narrow range of viewpoints. Try to synthesize views at stranger viewpoints and let the model produce bad images. Comment on the image quality degradation and artifacts that emerge.
4. Geometry, lighting, texture are entangled in the radiance field representation, but we can attempt to recover the geometry from a radiance field. Pick a few models from both the inward facing scenes and forward-facing scenes. Evaluate the trained MLP at a voxel grid resolution of your choice, and threshold the predicted σ values to obtain a 3D voxel. Visualize this voxel using Open3D. Remember to convert the geometry to Open3D coordinate convention.
5. Draw the camera cones of the input images (usually around 100). Convert their poses into the Open3D pose convention that we are using.
6. For the forward-facing scenes, undo the NDC warp back to euclidean coordinates, and visualize the geometry after the unwarp. What kind of artifacts do you observe? What do you think might be the strategy that NeRF model adopts to deal with this kind of forward-facing scenes?
7. Explore some other interesting properties you find in this representation.
8. Optional: extract a 3D mesh from the occupancy values using **marching-cubes**.
9. Optional: train a NeRF on your SfM output for an object you like. Maybe a mouse or a cup.

3 Project submission

Please describe the pipeline you end up implementing in the PDF writeup. We recommend LaTeX, but other typesetting environments like Word or Google Docs are fine, as long as the submitted document is in PDF. Please submit this PDF (try to keep it shorter than 4 pages) along with the full source code files, and a .tar file containing any additional material we need to replicate your experiments. This may include your own images, etc.

Your writeup should include:

- Precise description of the implementation, written in a way that would allow a fellow student in the course to replicate your design.

- Discussion of any choices you had to make, and an explanation of how and why you made a particular choice there. (E.g., things that can be parameterized in different way; settings for hyperparameters; etc.)
- Qualitative evaluation of the performance of your method, and any thoughts on potential improvements one could make with more time/effort. This includes improvements to the method per se, as well as improvements to implementation (such as speeding it up).
- Discussion of the fundamental limitations of the method as designed and implemented, including any specific failure modes (types of input/reference combinations in which the method is likely to fail).

References

[1] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[2] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. *arXiv preprint arXiv:2111.11215*, 2021.