

Nerf-Pytorch

Task 1: Setting up the env and reconstructing lego

The setup was completed with some conflict in dependencies (torch and numpy), the lego was reconstructed

The video output is clear as shown in the repo and paper.

This folder consists of the videos generated at different iterations: **Task 1**

Implementation:

```
requirements.txt
1 torch==1.11.0
2 torchvision==0.9.1
3 imageio
4 imageio-ffmpeg
5 matplotlib
6 configparser
7 tensorboard==2.0
8 tqdm
9 opencv-python
10
```

```
[TRAIN] Iter: 146200 Loss: 0.002417289651938332 PSNR: 31.92352294921875
[TRAIN] Iter: 146300 Loss: 0.0026620037388056517 PSNR: 33.85533005028297
[TRAIN] Iter: 146400 Loss: 0.00280450886048926 PSNR: 34.26043701171875
[TRAIN] Iter: 146500 Loss: 0.00252707302570343 PSNR: 30.848291397094727
[TRAIN] Iter: 146600 Loss: 0.002521931193768978 PSNR: 32.68977737426758
[TRAIN] Iter: 146700 Loss: 0.0022846395149827003 PSNR: 33.66714096069336
[TRAIN] Iter: 146800 Loss: 0.003820823272690177 PSNR: 31.61016082763672
[TRAIN] Iter: 146900 Loss: 0.002971435897052288 PSNR: 32.70148849487305
[TRAIN] Iter: 147000 Loss: 0.0036777732893824577 PSNR: 29.89671273803711
[TRAIN] Iter: 147100 Loss: 0.0027538915164768696 PSNR: 31.976770401000977
74%
```

After completion:

```
requirements.txt
1 torch==1.11.0
```

```
1 2.7276272773742676 | 11/25 [00:30<00:38, 2.73s/it] 1
2 2.7300713062286377 | 12/25 [00:32<00:35, 2.73s/it] 1
3 2.729926109313965 | 13/25 [00:35<00:32, 2.73s/it] 1
4 2.728933095932007 | 14/25 [00:38<00:30, 2.73s/it] 1
5 2.7312941551208496 | 15/25 [00:40<00:27, 2.73s/it] 1
6 2.7326953411102295 | 16/25 [00:43<00:24, 2.73s/it] 1
7 2.7322850227355957 | 17/25 [00:46<00:21, 2.73s/it] 1
8 2.734842538033618 | 18/25 [00:49<00:19, 2.73s/it] 1
9 2.7356157302856445 | 19/25 [00:51<00:16, 2.73s/it] 1
0 2.7318859100341797 | 20/25 [00:54<00:13, 2.73s/it] 2
1 2.735360860824585 | 21/25 [00:57<00:10, 2.73s/it] 2
2 2.73778557774048 | 22/25 [01:00<00:08, 2.73s/it] 2
3 2.7382123470306396 | 23/25 [01:02<00:05, 2.74s/it] 2
4 2.7364463806152344 | 24/25 [01:05<00:02, 2.74s/it] 2
100% | 25/25 [01:08<00:00, 2.73s/it] 2
Saved test set
100% | 25/25 [01:08<00:00, 2.74s/it] 2
[TRAIN] Iter: 200000 Loss: 0.002358278026804328 PSNR: 33.50570297241211
200000/200000 [3:14:36<00:00, 17.13it/s]
```

Task 2: Implementing on Custom dataset

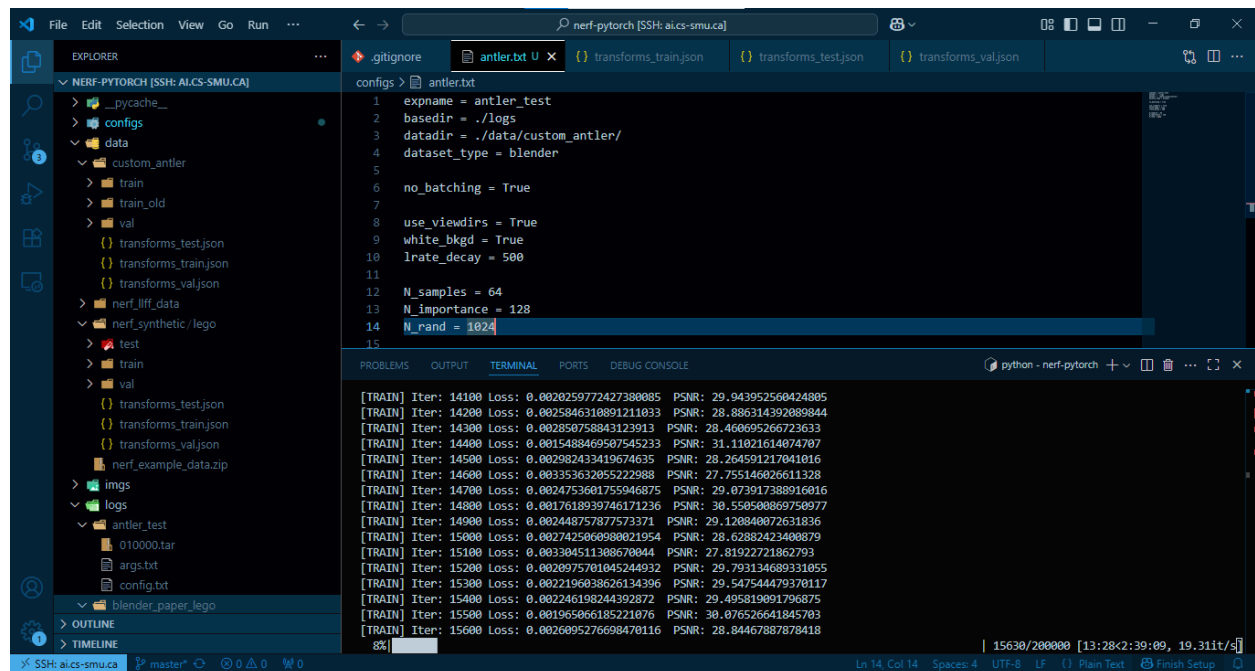
First I extracted the frames and made transform.json from the .stl file.

I wanted to render it via blender but my system did not support it so I used an alternative approach, which I believe did not generate the transform.json. Hence the output of video is not optimal.

The script (and config files) can be found here: https://github.com/Talha-Here/render_frames

The data can be found here: [antler_data2](#)

Implementation:



```
configs > antler.txt
1 expname = antler_test
2 basedir = ./logs
3 datadir = ./data/custom_antler/
4 dataset_type = blender
5
6 no_batching = True
7
8 use_viewdirs = True
9 white_bkgd = True
10 lrate_decay = 500
11
12 N_samples = 64
13 N_importance = 128
14 N_rand = 1024
15
```

```
[TRAIN] Iter: 14100 Loss: 0.0020259772427380085 PSNR: 29.943952560424805
[TRAIN] Iter: 14200 Loss: 0.0025846310891211033 PSNR: 28.886314392080804
[TRAIN] Iter: 14300 Loss: 0.002850758843123913 PSNR: 28.460695266723633
[TRAIN] Iter: 14400 Loss: 0.0015489469597545233 PSNR: 31.11021614874707
[TRAIN] Iter: 14500 Loss: 0.002982433419674635 PSNR: 28.264591217041816
[TRAIN] Iter: 14600 Loss: 0.003353632055222988 PSNR: 27.755146026611328
[TRAIN] Iter: 14700 Loss: 0.0024753601755946875 PSNR: 29.073917388916016
[TRAIN] Iter: 14800 Loss: 0.0017618939746171236 PSNR: 30.559500869750977
[TRAIN] Iter: 14900 Loss: 0.002448757877573371 PSNR: 29.120040072631836
[TRAIN] Iter: 15000 Loss: 0.0027425060980021954 PSNR: 28.62882423400879
[TRAIN] Iter: 15100 Loss: 0.003304511308670044 PSNR: 27.81922721862793
[TRAIN] Iter: 15200 Loss: 0.0020975701045244932 PSNR: 29.793134689331055
[TRAIN] Iter: 15300 Loss: 0.0022196038626134396 PSNR: 29.547544479370117
[TRAIN] Iter: 15400 Loss: 0.002246198244392872 PSNR: 29.495819091796875
[TRAIN] Iter: 15500 Loss: 0.001965966185221076 PSNR: 30.076526641845703
[TRAIN] Iter: 15600 Loss: 0.0026095276698470116 PSNR: 28.84467887878418
```

However the output is not as required (I changed white_bkgd to False, but could not get the correct results). I believe using blender would have given the correct results.

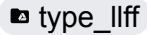
Results: [type_blender](#)

Then I tried to use the lllf type, which needs colmap. Here I used Kaggle (GPU T4 x2) because I could not run it on my system.

This implementation failed multiple times, as colmap kept on crashing but after changing configuration it worked.

Implementation:

Notebook: [nerf-pytorch_2.ipynb](#)

The videos could not be compiled as I reached the weekly limit of GPU provided by Kaggle. Here are the png that were generated in between the process: 

References:

1. [A PyTorch implementation of NeRF \(Neural Radiance Fields\) that reproduces the results.](#)
2. <https://github.com/bryceschultz/nerf-pytorch>
3. [The Annotated NeRF – Training on Custom Dataset from Scratch in Pytorch](#)