CS6353 course project

Installation

```
conda create -n cs6353-project python=3.9
pip install -r requirements.txt
conda install pytorch==1.11.0 torchvision==0.12.0 torchaudio==0.11.0
cudatoolkit=11.3 -c pytorch
```

How To Run?

Quick Start

Download data for two example datasets: `lego` and `fern`

```
bash download_example_data.sh
```

To train a low-res `lego` NeRF:

```
python run_nerf.py --config configs/lego.txt
```

After training for 100k iterations (~4 hours on a single 2080 Ti), you can find the following video at `logs/lego_test/lego_test_spiral_100000_rgb.mp4`.

To train a low-res `fern` NeRF:

```
python run_nerf.py --config configs/fern.txt
```

After training for 200k iterations (~8 hours on a single 2080 Ti), you can find the following video at `logs/fern_test/fern_test_spiral_200000_rgb.mp4` and `logs/fern_test/fern_test_spiral_200000_disp.mp4`

More Datasets

To play with other scenes presented in the paper, download the data here. Place the downloaded dataset according to the following directory structure:

```
├── configs
│   ├── ...
│
├── data
│   ├── nerf_llff_data
│   │   └── fern
│   │   └── flower  # downloaded llff dataset
│   │   └── horns   # downloaded llff dataset
│   │   └── ...
│   ├── nerf_synthetic
│   │   └── lego
│   │   └── ship    # downloaded synthetic dataset
│   │   └── ...
```

To train NeRF on different datasets:

```
python run_nerf.py --config configs/{DATASET}.txt
```

replace `{DATASET}` with `trex` | `horns` | `flower` | `fortress` | `lego` | etc.

To test NeRF trained on different datasets:

```
python run_nerf.py --config configs/{DATASET}.txt --render_only
```

replace `{DATASET}` with `trex` | `horns` | `flower` | `fortress` | `lego` | etc.

Pre-trained Models

You can download the pre-trained models here. Place the downloaded directory in `./logs` in order to test it later. See the following directory structure for an example:

```
├── logs
│   ├── fern_test
│   ├── flower_test  # downloaded logs
│   ├── trex_test    # downloaded logs
```

Reproducibility

Tests that ensure the results of all functions and training loop match the official implentation are contained in a different branch `reproduce`. One can check it out and run the tests:

```
git checkout reproduce
py.test
```

Method

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Ben Mildenhall*[1],

Pratul P. Srinivasan*[1],

Matthew Tancik*[1],

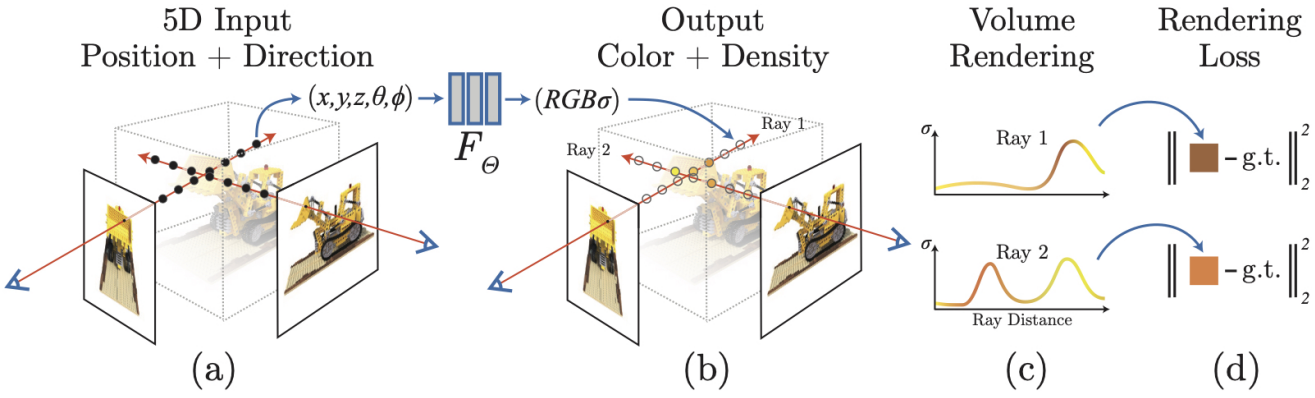
Jonathan T. Barron[2],

Ravi Ramamoorthi[3],

Ren Ng[1]

[1]UC Berkeley, [2]Google Research, [3]UC San Diego

*denotes equal contribution



> A neural radiance field is a simple fully connected network (weights are ~5MB) trained to reproduce input views of a single scene using a rendering loss. The network directly maps from spatial location and viewing direction (5D input) to color and opacity (4D output), acting as the "volume" so we can use volume rendering to differentiably render new views

Citation

Kudos to the authors for their amazing results:

```
@misc{mildenhall2020nerf,
    title={NeRF: Representing Scenes as Neural Radiance Fields for View
Synthesis},
    author={Ben Mildenhall and Pratul P. Srinivasan and Matthew Tancik and
Jonathan T. Barron and Ravi Ramamoorthi and Ren Ng},
    year={2020},
    eprint={2003.08934},
    archivePrefix={arXiv},
    primaryClass={cs.CV}
}
```

However, if you find this implementation or pre-trained models helpful, please consider to cite:

```
@misc{lin2020nerfpytorch,
  title={NeRF-pytorch},
  author={Yen-Chen, Lin},
  publisher = {GitHub},
  journal = {GitHub repository},
  howpublished={\url{https://github.com/yenchenlin/nerf-pytorch/}},
  year={2020}
}
```