Rheinische Friedrich-Wilhelms-Universität Bonn
Prof. Dr. Maren Bennewitz

Institut für Informatik
Abteilung VI
Humanoid Robots Lab
Adresse:
Friedrich-Hirzebruch-Allee 8
53115 Bonn

Humanoid Robotics

Assignment 3

Due Thursday, May 8th, before class.

Neural Fields:

Overview

In this assignment you will:

- **Implement a basic NeRF** in PyTorch3D, handling large numbers of rays via a batched forward pass for rendering and Monte Carlo sampling for training.
- **Experiment with silhouette supervision** by incorporating a mask-based loss term.
- **Extend NeRF to render depth**, deriving and implementing the volume rendering formula that computes per-ray depth.
- **Enhance NeRF with depth supervision**, adding a depth loss to the training loop and visualizing the results.

Two different ipython notebooks are provided, one for each task: **task_1_fit_nerf_rgb.ipynb** and **task_2_fit_nerf_rgbd.ipynb**. Wherever **# TODO** comment is marked in the ipython notebook, fill in the required code or explanatory text. When you're done, submit your completed notebooks along with any supporting figures or logs as a zip file.

Environment

Codes are tested on conda environment with Python 3.9 + PyTorch 2.0.0 + CUDA 11.8 + PyTorch3D 0.7.3 + Jupyter Notebook. You can create a Conda environment as follows:

```
conda create -n pytorch3d python=3.9
conda activate pytorch3d
# Install PyTorch + CUDA 11.8
conda install pytorch==2.0.0 torchvision==0.15.0 torchaudio==2.0.0 pytorch-cuda=11.8 -c pytorch
-c nvidia
# Install PyTorch3D
conda install -c iopath iopath
pip install pytorch3d -f
https://dl.fbaipublicfiles.com/pytorch3d/packaging/wheels/py39_cu118_pyt200/download.html
# Install other tools
conda install jupyter
pip install scikit-image matplotlib imageio plotly opencv-python
# Make sure the numpy version is <2
pip install numpy==1.24.4
```

Please note: NeRF is computationally intensive. Running the notebook on a CPU will be very slow. We strongly recommend using a **GPU** with at least **6 GB** of memory. If you don't have access to a suitable GPU, you can run the notebooks on Google Colab. In that case, just install the required packages directly in the notebook.

1. Basic NeRF with Silhouette Experiment (task_1_fit_nerf_rgb.ipynb)

In the lecture on Neural Fields, an intuitive and mathematical understanding of volume rendering was provided. In this first task students are expected to build a minimal NeRF pipeline in PyTorch3D, from ray sampling through volume rendering to training on RGB and silhouette images. Although the note-book provides skeleton code, the key functionalities must be implemented by the students as detailed below:

a. Implement batched_forward function

Implement block-wise rendering to avoid OOM (out of memory) in batched_forward function of the class NeuralRadianceField. Please fill the blanks of batch_outputs (Use forward pass to obtain the result) and rays_densities, rays_colors (Concatenate the per-batch result).

(3 points)

b. Influence of Batch Splitting on Input Rays

Explore how the number of chunks (n_batches) affects GPU memory. Record the required GPU memory for n_batches = [16,1024] in the below table. Explain in plain text why increasing n_batches reduce memory at the cost of additional loop overhead and discuss how this affects overall speed.

n_batches	GPU Memory (GB)
16	
1024	

(3 points)

c. Influence of Samples Per Ray

Explore how the number of samples per ray (N_pts_per_ray) affects GPU memory. Record the required GPU memory for N_pts_per_ray = [16, 64] in the below table. Explain in plain text why decreasing N_pts_per_ray reduce memory usage, and discuss how this impacts overall speed and final rendering quality.

N_pts_per_ray	GPU Memory (GB)
16	
64	

(3 points)

d. Training Iterations and Convergence

Evaluate how quickly NeRF converges by varying the number of training iterations. Record the shape quality and training time for n_iter = [100, 500, 3000] in the below table. Explain in plain text how increasing the number of iterations impacts the clarity of the reconstructed shape and the overall training time.

n_iter	Shape Quality (No / Almost / Clear)	Time (Sec)
100		
500		
3000		

(3 points)

e. Silhouette Loss Ablation Experiment

Evaluate the effect of removing the silhouette loss term with the default iterations of 3000: Remove sil_err so that loss = color_err. Tabulate your results in the below table. Explain in plain text whether and how silhouette supervision improved RGB reconstruction.

loss	Shape Quality (Bad / Good)
color_err	
color_err + sil_err	

(3 points)

UNIVERSITÄT BONN

Rheinische Friedrich-Wilhelms-Universität Bonn

Institut für Informatik
Abteilung VI
Humanoid Robots Lab

2. Depth Rendering and Supervision (task_2_fit_nerf_rgbd.ipynb)

In this task you'll extend your Task 1 NeRF to output per-ray depth maps and use provided ground-truth depth for supervision, implementing the depth-rendering formula, adding a depth loss term, and visualizing the results.

a. Volume Rendering Depth Formula

Write down and annotate for each symbol.
Compositing weights:

$$w_i = \alpha_i * T_i =$$

Expected depth:

$$\text{depth} =$$

Explain in plain text what happens to the computed depth when a ray does not accumulate any opacity (i.e., it effectively travels to "infinity"), and describe a practical strategy for handling such rays.

(3 points)

b. Implement functions to get depth from NeRF

Complete the helper methods in the modified NeuralRadianceField class. Please fill the blanks in _get_weights (Compute the array of weights), _get_depth (Compute the expected depth), and batched_forward (Obtain the third output rays_depths).

(3 points)

c. Implement Depth Visualization

Compute percentiles [5, 95] for each depth map. Normalize to [0,1] by clipping to that percentile range. Explain in plain text why we need to convert these values for visualizing the depth.

(3 points)

d. Implement Depth Interface and Loss

Extend the forward pass to produce rendered depths and supervise them with ground truth depth images in the training loop. Please complete the implementation of rendered_depths (Call the forward function of neural_radiance_field) and depth_err (Compute the depth loss using sample_images_at_mc_locs function and Huber loss), by filling in the blanks.

(3 points)

e. Depth Loss Weighting Experiment

Investigate how the weight on the depth-loss term influences training convergence. Record the results for depth_weight = [0.1, 1.0] in the below table. Explain in plain text why using a larger depth_weight can hinder or improve convergence.

depth_weight	Convergence (Yes / No)
0.1	
1.0	

(3 points)