6.8300 Pset 4 Problem 3 writeup

Zhi Ren

March 22, 2025

1 Positional encoding

Code written for positional encoding below:

```
freq_bands = torch.tensor([torch.pi * 2 ** (i+1)
    for i in range(self.num_octaves)])
```

```
# Fill the tensor with sine and cosine embedded values
for i, freq in enumerate(freq_bands):
    embedded_samples[..., i :: 2 * self.num_octaves]
    = torch.sin(samples * freq)
    # TODO: materize the PE for cos.
    # Copy the below code (a few lines is enough)
    # to the final submission report.
    # Better to add a small title
    # "Positional Encoding of NeRF" in the report

    embedded_samples[..., i + 1 :: 2 * self.num_octaves]
    = torch.cos(samples * freq)
```

2 NeRF rendering

There is one line that needs to be filled in in src/nerf.py:

```
# 4. Composite the alpha values and colors
# TODO: call the final composite function
# replace the following line with the final composite function
radiance = self.alpha_composite(alphas, colors)
```

2.1 Short answer question

Question 1. *The rendering model used in our code is slightly different from the one used in the original NeRF paper. Please identify at least one difference.*

Answer: The rendering model used in our code differs from the one used in the original NeRF paper in that the original paper uses a stratified sampling method to generate samples along the rays, so that they divide the ray into a number of intervals and drawn a random sample from each sinterval, whereas in our code, we take the mid points of all the intervals and there is no random sampling there. □

3   Training

Attached below are several visualizations from the last few iterations of training.

Figure 1: Visualization of the scene at iteration 7808
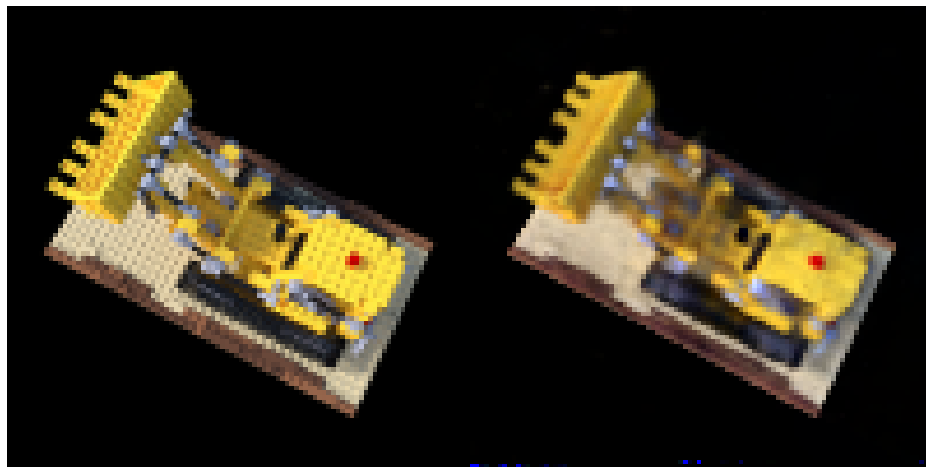
Figure 2: Visualization of the scene at iteration 7936

Figure 3: Visualization of the scene at iteration 8064

4 Final rendering

From the final rendering:

Figure 4: 000000 in spin

Figure 5: 000001 in spin