SRMCV Project: Extending the Diffeomorphic Neural Reconstruction Approach - Week 4 Report

1 Overview

Our main goal for the fourth week was to start integrating NeRF into our project. Previously, we found that our network was unable to reconstruct input scenes with non-trivial backgrounds. Before using a NeRF to obtain the background color, we tested if this idea could work by using the ground truth rendered background as input to our rendering function as described in subsection 1.2. We also added the NeRF network to our implementation.

Finally, to facilitate hyperparameter tuning, we included a validation process with early stopping and added Tensorboard support to track validation losses and rendered images at each validation step.

1.1 Memory Leak Issue

At the start of the week we experienced some issues with our implementation crashing after just a few iterations due to a CUDA memory error. We later found that this was due to the placement of the image save command for the Tensorboard writer. Even though we moved the tensor to cpu prior to handing it off, it seems the placement of the command within the inner training loop somehow lead to the memory not properly being freed.

Moving this into the outer training loop fixed the issue.

1.2 Preliminary Tests for NeRF

To see if replacing the current black background with a NeRF would work at all, we first had to see if rendering with a fixed (correct) background would work.

So far, the background was assumed to be all black and therefore some modifications were required in our rendering and training codes. To insert the NeRF background option to our code, the steps provided in the Road Map were followed carefully as described in subsection 1.3. For our tests, we replace this background with the rendered cube background in the training pipeline.

The main idea here is to use the scene consisting of cubes without the bunny mesh to create a background for rendering the bunny with a background image. The cube background was previously created using Blender. It is then rendered using the 'create_dataset.py' file where we set L0 is to the same value as the full scene of bunny mesh to render the background cubes. Also, here, the resulting image is not normalized because the values of the pixels in the background of cubes are already between 0 and 1. This way, it is possible to set the NeRF_bgc as this background image as a parameter for the renderer. After the check is done, the loss value computed by comparing with the ground truth full scene is 10^{-8} per image, implying our implementation is correct (see Figure 1).

After these check steps, the training is done using the velocity and image losses for 1000 iterations without cropping. As can be seen in Figure 2, the bunny mesh is constructed successfully except some parts around the ear.

Unfortunately, the results for the bunny, especially around the ears, are very similar to our previous test on a non-trivial background which was performed without a NeRF/ground truth background as

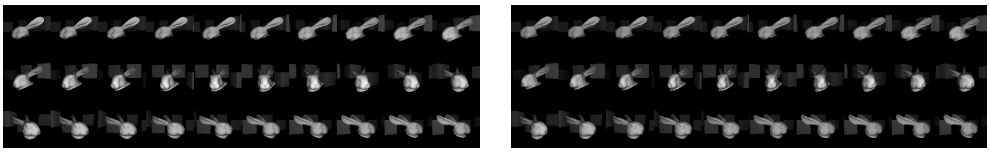

Figure 1: On the left, the ground truth images of the full scene consisting of meshes of cubes as the background and the bunny mesh. On the right, the scene consisting of only the bunny mesh rendered with the background images of cubes.

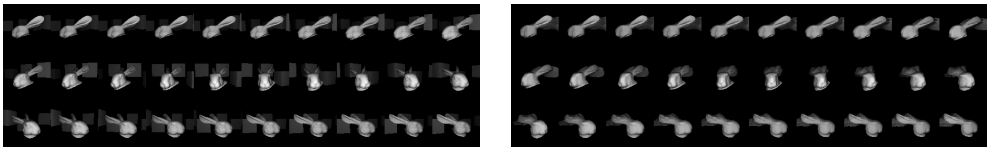

Figure 2: On the left, the ground truth images of the full scene consisting of meshes of cubes as the background and the bunny mesh. On the right, the reconstructed mesh.

part of the render step. The image can be found in our previous report. This implies that just adding a background will not fix the network's results on this scene.

There is a few possible causes for this problem:

1. The background is too similar in color and changing the scene might improve our results
2. We are missing view-points from the back of the bunny
3. Because the ears are never very close to the camera, they can never have a very high contrast to the background due to exposure to the point light
4. We need different hyperparameters (specifically learning rate and velocity loss weight)
5. It simply can't work because our network cannot differentiate between foreground and background

To find out what the reason is, we will have to do some experiments with different background setups. For now these results are a bit unfortunate.

1.3 Adding NeRF to the codebase

For the purposes of our project, we would like to model the background of our mesh using a NeRF, while rendering the mesh itself using the existing velocity field implementation.

While the results of the tests performed in subsection 1.2 are not great, it might still be plausible to replace the background with a NeRF after some adjustments.

So we decided to add the NeRF network to our codebase in preparation. We are using the existing implementation from https://github.com/lioryariv/volsdf/ with changes to the configuration only.

While we have added the network to our code, we have not yet run any tests on it, as we want to focus first on mitigating the problems that our tests have highlighted.

1.4 Implement validation for tracking during Training

In our previous work, we created validation datasets using viewpoints from many angles. Now we have incorporated a completed validation process with early stopping for better tuning while training. In tensorboard, the losses of the validation sets are depicted as well as the rendered images for every validation step.

1.5 Finding Parameters for the Bounding Sphere

For the implementation of the NeRF we need to separate the background from the foreground. Thus we added a sphere S(c,r) (of center c = (x= -0.67079, y = -1.78058 ,z = 0.92213 , and radius r=3.1)

where everything inside this sphere is called the foreground, and everything out of the sphere is called the background as it can be seen in Figure 3.

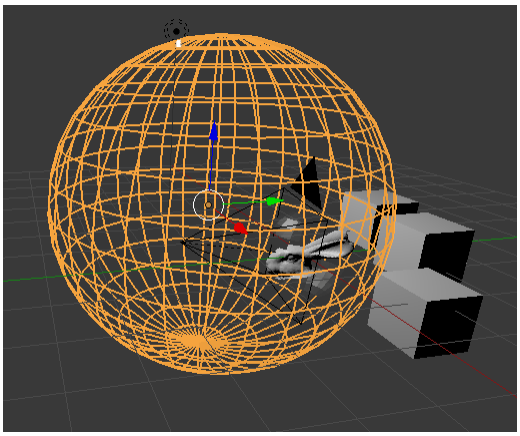

Figure 3: Separating foreground from background

2 Next Steps

Since we have found that our current setup does not work even with a ground truth background, we will have to tweak it a bit.

On the bright side, we now have a very good setup for testing different configurations, so we should be able to do some experimentation to hopefully find the problem and see if it is fixable.

This means that we will focus on experimenting with different configurations of the scene as well as hyperparameters to hopefully make this approach work.

3 Task Assignment

Vasiliki Papadouli:

- Incorporate a completed validation process for better tracking of training and depict the corresponding losses and images in tensorboard.
- Creation of the new dataset containing the background only by separating the joint mesh using Blender Software.
- Separate foreground from background by adding a sphere S(c,r) containing the foreground.

Alexander Fuchs:

- Integration of NeRF code into our codebase
- Tensorboard implementation
- Debugging (memory leak) and experiments

Zehranaz Canfes:

- Test the cropping method using the ground truth mesh as input
- Insertion of NeRF_bgc in the code
- Check if it works to use the real "background images" as background in the code and train with velocity loss