SRMCV Project: Extending the Diffeomorphic Neural Reconstruction Approach - Week 2 Report

1 Overview

Our main goal for the second week was to adjust the original codebase to be able to train a good mesh on our dataset. We also improved our training setup to simplify debugging and training different configurations in batch. Additionally we investigated whether we can train using purely the image loss (without the silhouette). After fixing various issues, we end up achieving good training results and should be able to proceed with including an image background next.

2 Verification of the existing implementation

In the first week, we found that after setting up our dataset and adapting the training code the initial training did not yield good results when training only on the image loss. The original implementation includes a loss based on the silhouette of the mesh, so our inital thought was to add this mask loss that we had left out previously. We implemented a renderer using a soft silhouette shader which returns the alpha channel in order to generate the masks of the mesh. We then extended our dataset with masks from the 30 viewpoints. During training, we render the silhouette of the object during each iteration to compute the mask loss.

In the following sections, we go over what problems we found with the existing implementation and our solutions.

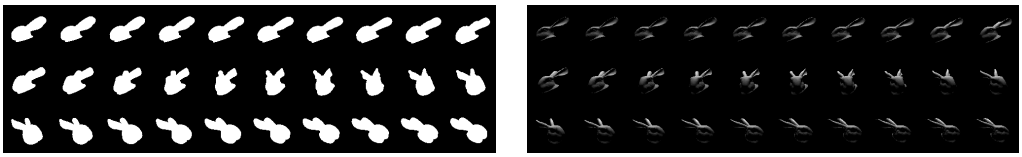
2.1 Improvements to training configuration

For all our following experiments we needed a good setup to easily execute different training configurations and look at their results quickly. For this we extracted the existing configuration into separate files. This allows us to create a set of configs ahead of time which can then be trained in sequence.

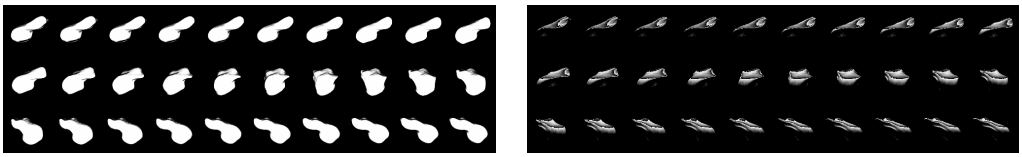
Alongside this we also added the ability to specify checkpoint intervals at which a checkpoint of the model as well as a render of all viewpoints is generated. This allows us to easily track training progress during/after training. We also save the ground truth views from the dataset as well as the used configuration file. Each training session's data is saved in a folder using a unique ID.

2.2 Loss weights for mask and image losses

First of all, we had to verify if the losses work as intended. There are two main losses that concern us: image loss and mask loss. Trying different weights for the losses may help us understand why the training fails. To test the training code on different loss weight setups, the training was run multiple times. During this we found that the mask loss seemed to lead the network towards a decent approximation of the mesh, while the image loss mostly caused the mesh to either deform wrongly or move out of the frame entirely. This can be seen in Figure 1 as the image loss weight increases, the training fails to reconstruct the bunny mesh and the image becomes darker. The implication is that the image loss is not working correctly. We can also see here that by itself the silhouette loss is not capable of generating a good quality mesh, as the surface contains unexpected deformations.



(a) Ground truth



(b) Mask loss only

(c) Image and mask loss

Figure 1: Silhouette (left) and image (right) renders of mesh after 100 training iterations each. The silhouette loss works towards a reasonable approximation of the mesh. The image loss is counterproductive, moving the object out of frame, indicating a problem.

2.3 Lighting

During our investigation we noticed that the point light source was not actually positioned at the camera point but instead above the object. This error occured, because in pytorch3d the light position has to be given in world space coordinates, while the camera transformation has to be given in the camera space. With this, we generated our dataset again, the new ground truth can be seen in 2.

The bunny is now much more well-lit, though this change by itself did not fix the image loss.

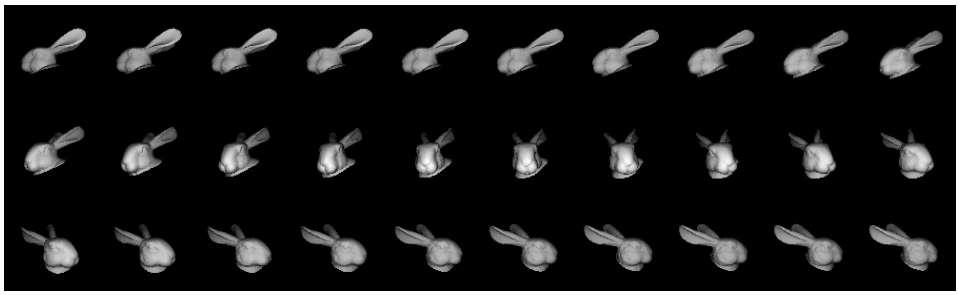


Figure 2: Render of the dataset after moving the point light source to the camera position. The bunny's features are much more visible.

2.4 Training adaptations

While training with only the image loss we still found that the optimization would generally just slowly move our mesh out of frame within the first 100 iterations. After going over the loss calculation and other possible error sources, we found that the problem was actually with our learning rate. Reducing it by a factor of 10 instantly lead to vastly improved results, preventing the network from moving the mesh out of frame. Results for training 1000 iterations using only the image loss can be seen in Figure 3. There are however still visible problems with the mesh where the network seemingly tries to move a part of it out of frame.

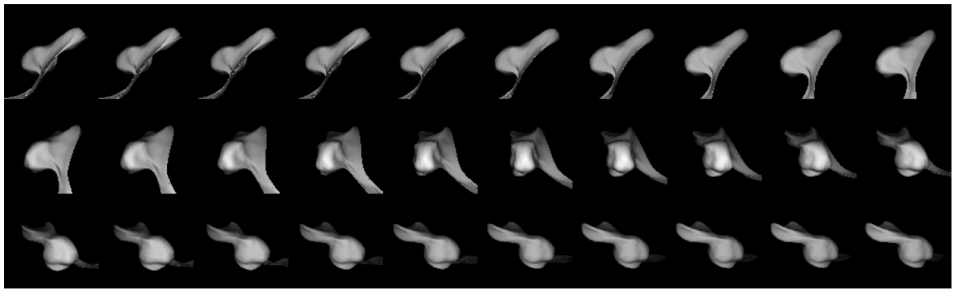


Figure 3: Render of mesh trained using image loss only

We finally looked at the image loss itself. The original paper uses the mean of the L1 loss on the image pixels. We found that using the sum instead of the mean fixes the issues with our model leading to our final result, as seen in Figure 4.

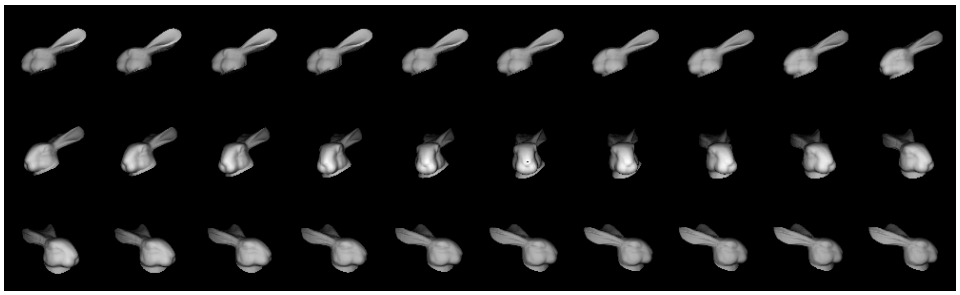


Figure 4: Renderings with SoftCookTorranceShader using only the image loss

3 Next Steps

As the training process now works correctly, even when using only the image loss, we can move on to the main part of the project. For this we will begin by looking at how a background affects training results and then how to include a NeRF for modeling the background.

4 Task assignment

Vasiliki Papadouli:

- Implementing Silhouette Renderer and creating the dataset of masks.
- Run experiments for the debugging process, observe the results and check the training process without silhouette loss.

Alexander Fuchs:

- Improved training configuration, allow batch training
- Lots of debugging (e.g. fixed lighting, image loss)
- Added training script outputs for checkpointing and debugging (save rendered images, weights, configuration)

Zehranaz Canfes:

- Verify the rendering settings using the ground truth mesh
- Work on bugs in the training code
- Run multiple experiments with different loss weights and analyse the results
- Try overfitting to only one render view