NEDF: CREATING NEURAL DISTANCE FIELDS WITH DEPTH MAPS

Alex Foulon, Alexandre Miralles, Antoine Marie

1. INTRODUCTION

Real-time rendering is one of the most important subjects in computer graphics, and 3D reconstruction is paramount in computer vision. Our project has the potential to make progress in both fields by leveraging an entirely new neural network training process.

Our goal is to train a deep learning model to simulate a distance field using only a few depth maps and camera transforms as training material. The estimated distance field could then be used to render synthetic scenes or to reconstruct entire objects, both using a process called sphere tracing.

It is highly similar to Neural Radiance Fields (NeRFs) [1] regarding the inputs and the architecture, but differs significantly on the training process, the type of outputs it produces and it's use cases.

It differs from all of the literature in the way that it supervises the model outputs, potentially leading to considerable improvements in simplicity and speed.

2. LITERATURE REVIEW

There are several notable approaches for training models to predict distance fields, the key articles being the following:

- NeuralUDF: Learning Unsigned Distance Fields for Multi-view Reconstruction of Surfaces with Arbitrary Topologies [2]
- DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation [3]
- Learning Signed Distance Fields for Multi-View Surface Reconstruction [4]

NeuralUDF takes advantage of regular images as input instead of depth maps. This is beneficial as it allows the reconstruction of objects from real photographs. However, our approach focuses on speeding up both inference and training by using only essential data: depth maps. If needed, we could still integrate existing image-to-depth conversion models as a preprocessing step, enabling the reconstruction of real-world objects in a similar way.

DeepSDF differs by requiring an existing signed distance field (SDF) representation and then learning to predict it in a continuous manner. While both methods aim for the same result, our model directly reconstructs the SDF purely from depth maps, eliminating the need for a pre-existing SDF reference, which would be very limiting.

Lastly, Learning Signed Distance Fields for Multi-View Surface Reconstruction adopts a strategy close to ours, but relies on supervising the distance values on each sampled point along the rays. It works great but in this paper we will try to supervise only the depth accumulated during sphere tracing [5] instead of all the individual points that were queried along the ray. We hope this will lead to faster training times, while keeping a satisfying quality and reducing the computations needed for training.

3. METHODOLOGY

3.1. Model architecture

The architecture of the NeDF model is inspired by Neural Radiance Fields as their architecture is well-suited for problems involving continuous spatial fields like SDFs.

It takes 3D Cartesian coordinates $\mathbf{x} \in R^3$ as input. To accurately represent small detailed variations in geometry, the input is expanded using positional encoding:

$$\gamma(\mathbf{x}) = \left[\mathbf{x}, \sin(2^n \pi \mathbf{x}), \cos(2^n \pi \mathbf{x}) \,\middle|\, n = 0, \ldots, L-1\right].$$

where L is the number of frequencies used for encoding. The resulting encoded vector $\gamma(\mathbf{x}) \in R^{6L+3}$ is passed as input to the network.

The input layer maps the encoded vector $\gamma(\mathbf{x})$ of size $6L+3$ to a high-dimensional latent space using a fully connected layer.

The network employs a total of 7 hidden layers, each consisting of 256 neurons. To improve convergence and mitigate the vanishing gradient problem, skip connections are used in layer 4. At this layer, the raw input $\mathbf{x}$ is concatenated with the intermediate representation. Skip connections allows the model to retain geometric context at all layers, which is critical for accurate predictions.

The final fully connected layer predicts a single scalar distance value.

3.2. Sphere tracing

Ray-marching, combined with distance fields, is a modern technique used to render implicit shapes and volumes effi-

ciently [5]. The idea is to cast a ray and make it march using variable step sizes that match the closest distance to the surface. When this distance approaches zero, we consider that the ray has hit the surface. This kind of ray marching is commonly referred to as sphere tracing, because the distance value at a given step actually corresponds to the radius of a sphere that touches the nearest surface, as seen in Figure 1.

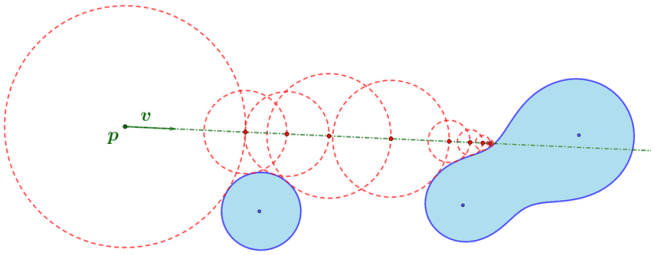

Fig. 1. Visualization of the sphere tracing process.

In our context, the goal is to feed the distance outputs of the model to the sphere tracing algorithm to render the scene. During the training process, the resulting accumulated depth is compared with the reference depth to calculate the error and optimize the weights. During inference, sphere tracing is used to render the learned scene using the estimated distance field.

3.3. Eikonal equation

The Eikonal equation is a mathematical principle often refered to as [6]:

$$\|\nabla f(\mathbf{x})\| = 1,$$

where $f(\mathbf{x})$ is the signed distance function (SDF). This equation enforces that the gradient of the SDF has a magnitude of 1 everywhere, ensuring that the predicted distances represent true geometric distances.

In the context of our model, the Eikonal equation is used for maintaining a coherent distance field. Without it, the model learns to predict very small distances which leads to precise depth estimation but poor ray-marching performance and an invalid distance field.

3.4. Dataset

To train our NN we needed a dataset consisting of depth maps of a single model. For the reference model of the dataset we used our robot seen in Figure 2.

We needed a way to create multiple cameras around the robot to have multiple points of view, because we want to be able to quickly generate a dataset to test our neural network on different models. Thus, Blender was the designated tool as the plugin (SED Creator [7] [1] permits the creation of a cluster

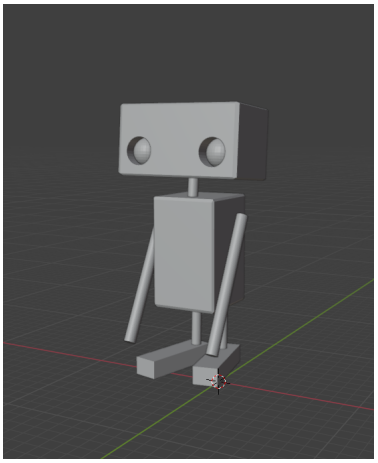

Fig. 2. Robot used for dataset

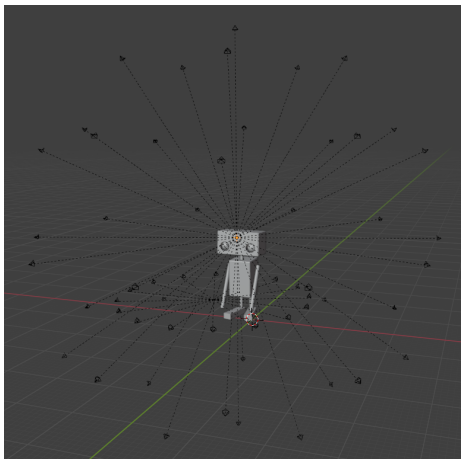

Fig. 3. Robot surrounded by two clusters of cameras

of cameras at once, as seen in Figure 3.

Then we render the depth maps, positions and angles of rotations of each cameras to create the dataset (all with the same plugin as it can be easily customized).

The final dataset includes a set of renders with cameras disposed as an icosahedron around the robot along with another set of closer cameras. It also includes two sets of renders using cameras orbiting around the robot, keeping the same height and always looking at the same point. This dataset is largely sufficient for our experiments.

3.5. Pre-processing

The 2D depth maps are flattened and filtered to remove the background pixels. This is done in order to speed up the training process by focusing the learning on meaningful pixels, as the model would not learn from estimating an infinite depth

[1] This add-on was developed by Alexandre Miralles and specifically modified for this project as we needed cameras positions.

value. Our pre-processing step reduces the amount of pixels to be treated by up to 83%.

3.6. Training process

To help the model generalize, we use batches of random pixels among the reference depth maps and cast one ray per pixel. We perform sphere tracing along each ray and compare the result with the reference depth to calculate the loss (as a reminder, the sphere tracing process stops when a distance close to zero is predicted, suggesting that the ray has hit the geometry). The Eikonal loss is computed as well after each batch using a portion of the model inputs to punish the model if it represents an invalid distance field. On top of these two losses, we experimented by adding a loss based on the amount of steps taken, in order to punish non-efficient sphere tracing. A last loss component based on the proportion of rays that end beyond the expected depth is added, to encourage the model to predict small values near the geometry.

4. EXPERIMENTS

All of the experiments were made using an NVIDIA RTX 3090 with 24Gb of video memory.

4.1. Promising results

During the first part of the research, we trained the model by stopping the rays when they would exceed the reference depth. This was done with the intention of helping the model to optimize the distance values towards the geometry. At the time we did not realize that this was preventing it from learning when to stop properly.

Still, the resulting distance fields were promising, they allowed the rays to precisely reach the geometry surface within an optimal amount of steps. With less than one hour of training, the depth loss averaged 0.01 and eikonal loss 0.25, which was really good. Figures 4 and 5 show the reference vs predicted depth for the filtered pixels, and as you can see they are very close.

Fig. 4. Reference depth

Fig. 5. Predicted depth

Figures 6 and 7 show a similar comparison on an unfiltered dataset entry. The values looks different only because of gray scale normalization, but the predicted depth is highly accurate, it produces a very usable depth map with a loss close to 0 on the geometry (the predicted depth for the background is not relevant in our experiments).

Fig. 6. Reference depth map

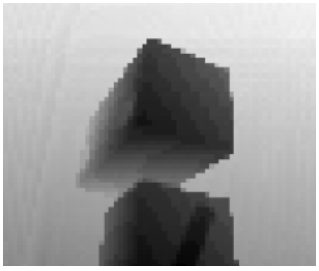

Fig. 7. Predicted depth map

As said before, the problem is that without the reference depth map to stop the rays, they would accurately step by the model's surface but never stop, making the model unable to produce depth maps from novel points of views, as seen on Figure 8

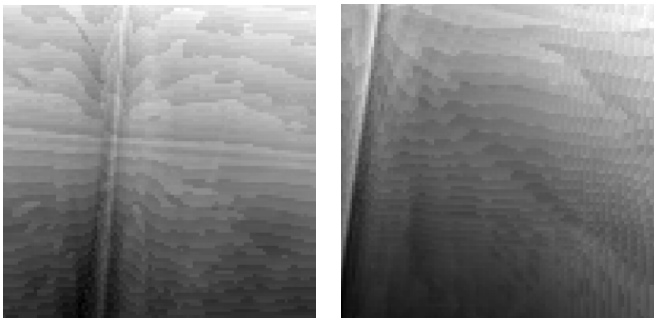

Fig. 8. Examples illustrating failed predictions (Rays failing to stop on geometry)

Still, the results were encouraging, now that the neural network is able to reach the geometry efficiently and precisely, all we have to do is to fine-tune the training process to help the model stop at the right moment (predict a value close to zero when the input is near the surface).

4.2. Learning when to stop

To try to overcome this flaw, we changed the sphere tracing process so that it does not stop when it exceeds the reference depth value. This forces the model to predict small values near the geometry, therefore stopping the ray marching process and predicting more accurate depth values without needing a reference depth map to stop. This happens because the loss calculated by comparing the reference depth with the predicted depth is successfully optimized. We trained our model

Fig. 9. Reference VS failed depth prediction on test entry

all night long in these conditions, and it resulted in a depth loss of 0.84, which is encouraging but certainly not enough to accurately predict depth on new viewing angles, as seen in Figure 9.

Training the model longer did not improve the loss, meaning the learning process or model has a limitation by design that cannot be overcome with raw compute power. While in it's current state the model does not produce useful renders, the promising results we saw makes us firmly believe that it can with a little more research. We could also rely on estimating the distance values at each point and supervising them, like in *Learning Signed Distance Field for Multi-view Surface Reconstruction* [4], but we still believe that we can find a different way, that is more computationally efficient.

5. CONCLUSION

The research we conduced lead to encouraging results. There are a lot of ways to estimate a distance field, but very few use only depth maps, and none supervise the accumulated depth to optimize the distance field. Perhaps there is a reason for that, but the results lead us to believe that there is a true potential.

6. REFERENCES

[1] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[2] Wei Jiang, Qiang Hu, et al., "Neuraludf: Learning unsigned distance fields for multi-view reconstruction of surfaces with arbitrary topologies," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[3] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[4] Sunghoon Ivan Wang et al., "Learning signed distance fields for multi-view surface reconstruction," in *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[5] John C. Hart, "Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces," *The Visual Computer*, 1996.

[6] "Eikonal equation — Wikipedia, the free encyclopedia," `https://en.wikipedia.org/wiki/Eikonal_equation`.

[7] Alexandre Miralles, "SEDcreator: A blender add-on for creating clusters of cameras in a scene," `https://github.com/Alex-665/SEDcreator_new`, 2024.

7. SOURCE CODE

The source code for this project is publicly available on GitHub: X13-A/NeDF.