Ray Tracing Learned Deformed SDFs

Project Proposal and Literature Review

Ryan Zesch

October 2021

1 Summary of Work Done

After considering various options, I decided to write a new ray tracer in Python instead of using my existing C++ ray tracer for this project. This option was chosen as the performance bottleneck for ray tracing an SDF stored in a neural net is the neural net itself, making any performance loss from using Python negligible. Additionally, this choice is convenient, as I am using Pytorch Lightning to learn SDFs.

In working with Dr. Sueda, I have successfully learned an SDF in a fully connected network which is able to interpolate between simple deformations, given linear modes of deformation from a training mesh. I have incorporated this simple neural net into my ray tracer, and am able to render the learned SDF using a conservative marching spheres approach.

In addition to SDFs, my ray tracer supports spheres, planes, and affine transformations. I currently support emissive and diffuse materials. I have also added the ability to render simple Blinn-Phong style lighting and normal maps for debugging purposes.

Currently, I am working to profile the code I have in place in order to optimize speeds before moving forward, as run times are longer than expected.

Figure 1: Rendered normals of a learned SDF. With one net, multiple poses are rendered by varying the input linear modes.

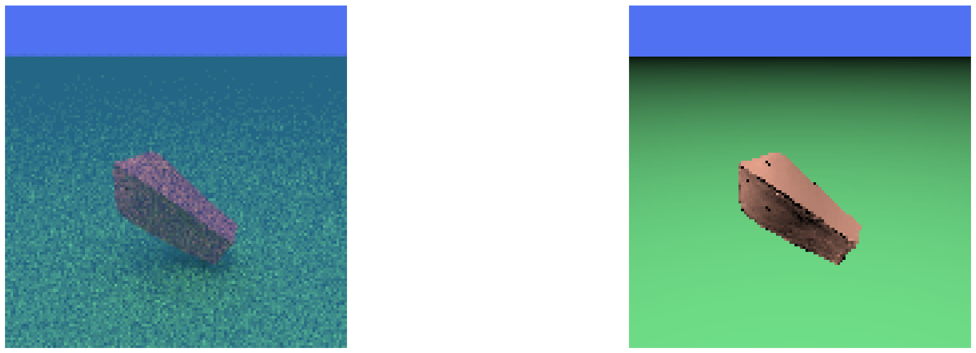
Figure 2: A 128×128 ray traced image with 40 rays per pixel and max recursion depth 5 took 8 hours. With Blinn-Phong style lighting and 1 ray per pixel, it took minutes.

2 Analysis of Work Done

I have taken a slightly different direction than expected compared to my stated goals for update 1.

- I have verified that linear modes of deformation interpolate well in learned SDFs in simple cases, though it appears that this approach may not work in more complicated cases. Training data is taken from my research, however, and is not the primary direction of this project.
- As mentioned above, I decided to write a new ray tracer instead of using my existing implementation. My implementation supports learned SDFs through simple conservative sphere tracing.
- I have not yet implemented other neural net architectures. This will be a goal of mine before the second update.

3 Plan for Completion

- By Update 2 (Nov. 18), I intend to
  - My primary goal before the next update is to implement another neural net architecture to compare against my simple MLP for the purposes of ray tracing.
  - I intend to profile my code in order to make optimizations, as currently rendering times are long when neural nets are involved.
  - In changing the network and optimizing, I hope to fix issues with computing surface normals - the black pixels in Figure 2 are an artefact of this.
  - Additionally, I hope to implement recently published ray tracing algorithms for fixed-scene neural SDFs in order to speed up ray tracing.
  - I will push rendering other meshes / scenes to the next goal set.
- By the Final Presentation (Dec. 15), I intend to
  - I hope to have optimized my ray tracer and implemented known algorithms leading to performance improvement.
  - I hope to have trained my network on multiple scenes to render.
  - I will compare runtimes for different approaches I have implemented.