Ray Tracing Learned Deformed SDFs

Project Proposal and Literature Review

Ryan Zesch

October 2021

1 Project Summary

For my class project, I intend to synthesize and extend recent work on ray tracing learned implicit signed distance functions with the work on deformations of learned representations of objects.

Neural networks can represent signed distance functions to arbitrary precision, and have been shown to offer interpolation based on latent codes in networks like DeepSDF [17]. Additionally, work has been done to speed up render times by leveraging the design of networks themselves, as in Neural Geometric Level of Detail [22]. In combining these works, I hope to train a network which is able to represent diverse deformations of an SDF while still leveraging rendering speedups due to the design of the network and implementation of ray casting algorithms.

More specifically, for a given mesh, I intend to develop a neural network which can represent deformations of that mesh which is well suited for the ray tracing pipeline. The exact network structure will be determined through testing various setups, but independent of structure, I intend for the network to take in inputs of a position and a deformation descriptor and output a signed distance to that point for the input deformation.

Per my research and literature review below, I believe that representing modes of deformation for a fixed mesh with a neural network is novel - in ray tracing, a traditional approach to render

deformed SDFs is to transform a cast ray into the deformed space. Additionally, I intend to test various network designs based on recent work in order to determine what performs best both for learning a given shape's deformations and for rending that shape.

2 Intended Goals

- By Update 1 (Oct. 28), I intend to
  - Verify that linear modes of deformation interpolate well in simple neural SDFs
  - Test various neural network architectures synthesized from recent publications, including different activation functions, loss functions, and layer structures
  - Modify my existing CPU based ray tracer to support neural networks SDFs with naive sphere tracing
- By Update 2 (Nov. 18), I intend to
  - Finalize the architecture of my neural network
  - Verify that the neural network is able to learn more complex scenes
  - Implement recently published ray tracing algorithms for fixed-scene neural SDFs
- By the Final Presentation (Dec. 15), I intend to
  - Modify ray tracing algorithms to take mesh deformation into consideration, depending on my chosen network architecture
  - Train the network on multiple complex meshes to verify robustness
  - Compare ray traced results of my chosen network architecture against other similar network architectures and against ground truth
  - Stretch goal: move ray tracing to the GPU

3 Literature Review

Ray tracing is a way of rendering scenes based on the realistic transfer of light as light scatters through an environment. In a simple ray tracing pipeline, rays $r(t) = r_0 + r_d t$ are projected from a virtual camera at r_0. These rays are then traced to their first collision with world geometry, often using implicit equations to determine points of intersection. Then, rays recursively scatter according to a bidirectional reflectance distribution function (BRDF) in order to composite lighting across multiple bounces, mimicking the way light scatters in reality.

Implicit equations are quick to evaluate for many primitives, such as planes, spheres, and triangles. An exact implicit equation is not always known for all geometry, however. In these cases, geometry may be able to be rendered by making use of a signed distance function (SDF). A signed distance function for a model X is a function $f : \mathbb{R}^3 \to \mathbb{R}$ such $f(x)$ is the distance from x to the closest point on X, with the sign of $f(x)$ determining which side of the surface x is on. SDFs are a special case of signed distance bounds, which only require $f(x)$ is less than or equal to the distance to the closest point on X.

One way to render signed distance functions is by using marching cubes [11]. In the marching cubes algorithm, an SDF's level set is rendered by filling a 3D grid with precomputed cube configurations based on how the level set passes through that cube. A second way to render SDFs, which closer aligns with the ray tracing pipeline, is sphere tracing [6]. In this procedure, a ray is marched along by taking a step of length $f(r(t_i))$ at time t_i. This ensures that each step will not cross the boundary of the signed distance bound. If $f(r(t_i)) < \varepsilon$, then the point of intersection with the signed distance bound is $r(t_i)$. This process allows SDFs to be rendered as smooth surfaces without first having to be cast to a triangulated mesh as in marching cubes.

One thing which is not immediately clear when considering using an SDF to represent geometry is how the representation can be deformed. In tetrahedral meshes, one can describe mesh deformation using energies based on the deformation gradient [13, 9]. Deformations can also be analyzed by projecting to lower dimensional spaces, determined by the mass and stiffness of a mesh, which encode the principal modes of deformation [19]. When considering deformation be-

tween specified start and end positions over time, linear, quadratic, and plastic deformations can be quickly computed if sacrificing physical accuracy is acceptable [15].

When considering the deformation of an SDF, these techniques do not transfer over one-to-one. Affine transformations of SDFs are easy to account for - when ray casting, one can simply put the cast ray into the transformed space and back. Early work on deformed SDF ray tracing by Alan Barr utilized the Jacobian of transformations to allow for some more general deformations with twisting, as well as ray tracing deformations as planes [2, 1]. More complex deformations require special care, however.

More recent work has incorporated both marching cubes and non-linear sphere tracing as a way to render deformed SDFs [18]. Given a fixed SDF, marching cubes can be used to create a bounding mesh surrounding the signed distance function. Deformations can easily be applied to this triangulation. When a ray then intersects this mesh, the ray is marched to its point of intersection with the SDF in a non-linear fashion, using the inverse Jacobian of the deformation to determine the direction in which to march the ray. This method has the advantage that the inverse deformation only must be computed or approximated once, allowing for deformations where computing the inverse is slow or impossible.

Outside of their use in rendering, SDFs can be used to enhance collision detection in physics simulations. By precomputing a grid based signed distance field for a given mesh in a fixed position, collisions in a simulation can be detected by deforming the SDF as the mesh deforms with finite element discretization [4]. This formulation only keeps track of grid cells near the mesh in order to improve runtime, whereas SDFs for ray tracing benefit from more globally defined SDFs in order to take larger steps along a given ray. More recent work on SDF collision demonstrated the ability to detect smoothly varying collision between rigid SDFs and deformed meshes [12].

Early representations of signed distance functions often relied on 3D occupancy or distance grid. While this approach works, it is often far too expensive in terms of memory footprint, with space requirements growing cubically. More recently, various attempts have been made to learn a signed distance function in a neural network. Representing signed distance functions to arbitrary

precision using neural networks is entirely possible, due to universal approximation properties of neural nets [8].

In 2015, a neural network called 3D ShapeNets [23] was developed with the goal of addressing the computer vision problem of object classification and completion from 2.5D data. The neural net behind this project was a voxel based convolutional neural net, based on Deep Belief networks - networks which train one layer at a time by pretending higher layers do not exist [7]. After training, this model is evaluated by feeding in a set of voxels which are known to be occupied or empty, and a set of voxel which are unknown if they are present, allowing for queries with incomplete data. Finally, a label and shape completion for the observation are determined by Gibbs sampling. At the time, this network outperformed competitors at the tasks of shape completion and labeling.

More recently, a similar network called DeepSDF [17] has been developed. Unlike 3D ShapeNets, DeepSDF does not rely on a voxel backing for its signed distance function representation, opting for a continuous approach where any point can be evaluated for a signed distance. Rather than learning to classify objects of different types like 3D ShapeNets, DeepSDF aims to learn different representations of a single class of objects, such as learning a space of chairs or of airplanes.

In order to learn such a class, training SDFs from the class are paired with randomly initialized latent codes. By employing a multi-layer perceptron (MLP) auto-decoder network structure with RELU nonlinearities, DeepSDF is able to optimize not only the network itself, but also the latent vectors of the training SDFs. At evaluation, the network returns a signed distance given a point in space and a latent code. By smoothly varying the latent code, the SDF smoothly deforms between different representations in the learned class - if a class of chairs was learned, interpolating between the latent codes for an armchair and a computer chair would smoothly deform the armchair into a computer chair. Similar to 3D ShapeNets, DeepSDF also provides shape completion capabilities. At the time of publication, DeepSDF outperformed similar networks. Since publication, various papers have improved upon the results of DeepSDF in multiple directions.

One improvement upon DeepSDF comes from MetaSDF [20]. The primary difference between the approaches of these models is how the latent code is applied. In DeepSDF, at evaluation, the

latent code and position query are appended into a single query vector passed to the network. In MetaSDF, on the other hand, two neural nets are involved - first a hypernetwork takes the latent code as input and outputs the layer weights for a second fully connected network into which a query point is passed, yielding a signed distance. As a result, MetaSDF is faster at inference while producing results on par with auto-decoder like networks. Of note, this paper demonstrates that concatenation of a latent code is a special case of their more general hypernetwork approach of determining layer weights.

Similar to DeepSDF, the SIREN [21] network architecture replaces the RELU nonlinearities of DeepSDF with sinusoidal activation functions, as well as incorporating a loss function which more strongly penalizes incorrectly predicted surface points. In doing so, SDF representations of complex scenes are smoothed out and scene gradients are more accurate to training data. The loss function used by SIREN is similar to that of the contemporaneous network created by Gropp et al. [5], which uses an Eikonal loss term to make the gradient of the network a unit vector in order to increase SDF accuracy away from the surface. The network produced by Gropp et al. additionally included loss terms to force the network to vanish on point cloud training data, and worked to prevent their model from failing during training.

Another improvement to DeepSDF is found in DIF-Net, which improves on texture mapping between class deformations and uses field based deformation [3]. Other models published contemporaneously to DeepSDF include Occupancy Networks [14] which are a special case of SDFs where only sign is considered, and Occupancy Flow Networks [16], which consider an Occupancy Network over time as it is deformed by a field.

In addition to training neural networks in order to learn classes of signed distance functions, it is possible to tailor neural network design in order to be highly suitable with the ray tracing pipeline. One goal of modern ray tracing is enabling a fully differentiable rendering pipeline - for example, a differentiable rendering pipeline allows for advancements in inverse graphics, such as learning based automatic scene and material tuning [24]. Recent work has provided a differentiable sphere tracing based rendering algorithm for neural implicit signed distance functions, based on the SDF

representation found in DeepSDF [10]. The most expensive step of neural net based differentiable rendering is the multiple neural net back propagations required while sphere tracing. Ways in which the number of back propagations needed can be mitigated include coloring multiple pixels per ray by splitting rays during sphere tracing, tuning sphere step sizes, and adjusting convergence criteria to suit use cases.

Another recent paper, Neural Geometric Level of Detail [22], focuses on the design of neural networks for representing signed distance functions in the context of ray tracing. Instead of an auto-decoder or MLP network design, an octree based collection of many smaller MLP networks is leveraged. This design allows for different levels of detail to be rendered based on blending MLPs from different depths in the octrees, and yields a render time approximately 100 times faster than rendering using DeepSDF style networks.

In summary, signed distance functions have a long history in computer graphics, and have been studied extensively for their usefulness in the ray tracing pipeline. Deformation of signed distance functions is a non-trival problem, but the use of algorithms like marching cubes and sphere tracing show promising results. Recently, SDFs have been encoded in neural networks of various designs with much success at representing object classes, as well as object labeling and reconstruction with incomplete data. Neural representations of SDFs have been designed to enhance the rendering pipeline by decreasing run time and enabling differentiable rendering.

References

[1] Alan H Barr. Ray tracing deformed surfaces. *ACM SIGGRAPH Computer Graphics*, 20(4):287–296, 1986.

[2] Alan H Barr. Global and local deformations of solid primitives. In *Readings in Computer Vision*, pages 661–670. Elsevier, 1987.

[3] Yu Deng, Jiaolong Yang, and Xin Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10286–10296, June 2021.

[4] Susan Fisher and Ming C Lin. Deformed distance fields for simulation of non-penetrating flexible bodies. In *Computer Animation and Simulation 2001*, pages 99–111. Springer, 2001.

[5] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

[6] John C Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996.

[7] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[8] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[9] Theodore Kim and David Eberle. Dynamic deformables: implementation and production practicalities. In *ACM SIGGRAPH 2020 Courses*, pages 1–182. 2020.

[10] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[11] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[12] Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Zach Corse. Local optimization for robust signed distance field collision. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(1):1–17, 2020.

[13] Jerrold E Marsden and Thomas JR Hughes. *Mathematical foundations of elasticity*. Courier Corporation, 1994.

[14] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[15] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless deformations based on shape matching. *ACM transactions on graphics (TOG)*, 24(3):471–478, 2005.

[16] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[17] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[18] Dario Seyb, Alec Jacobson, Derek Nowrouzezahrai, and Wojciech Jarosz. Non-linear sphere tracing for rendering deformed signed distance fields. *ACM Transactions on Graphics*, 38(6), 2019.

[19] Eftychios Sifakis and Jernej Barbic. Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction. In *Acm siggraph 2012 courses*, pages 1–50. 2012.

[20] Vincent Sitzmann, Eric R Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. *arXiv preprint arXiv:2006.09662*, 2020.

[21] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[22] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. 2021.

[23] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[24] Károly Zsolnai-Fehér, Peter Wonka, and Michael Wimmer. Gaussian material synthesis. *arXiv preprint arXiv:1804.08369*, 2018.