Reconstructing 3D Geometries for Scientific Applications: An Image to Simulation Pipeline

by

Marissa Isabella Ramirez de Chanlatte

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy
in
Applied Science & Technology
and the Designated Emphasis
in
Computational and Data Science and Engineering
in the
Graduate Division
of the
University of California, Berkeley

Committee in charge:
Professor Phillip Colella, Co-Chair
Professor Trevor Darrell, Co-Chair
Associate Professor Angjoo Kanazawa

Spring 2024

Reconstructing 3D Geometries for Scientific Applications: An Image to Simulation Pipeline

Abstract

Reconstructing 3D Geometries for Scientific Applications: An Image to Simulation Pipeline

by

Marissa Isabella Ramirez de Chanlatte

Doctor of Philosophy in Applied Science & Technology
and the
Designated Emphasis in Computational and Data Science and Engineering

University of California, Berkeley

Professor Phil Colella, Co-Chair
Professor Trevor Darrell, Co-Chair

Numerical simulation is a powerful tool that aids scientists in the understanding of the physics of the world around them, but to achieve an understanding of the physics, we must start with a representation of the geometry. Embedded Boundary (EB) simulation provides robust, automatic handling of complex geometries, enabling large scale physics simulations, but it still depends on an initial 3D implicit function representation of said geometry. These geometries are typically user-supplied through 3D design specifications, but for many domains of interest (i.e. outdoor scenes such as cities or forests) no such design specifications exist. This means that simulations of this nature rely on expensive, time consuming, and often noisy measurements from LiDAR or other sources, and because of the cost and lack of quality in the results, there have been relatively few 3D, large scale EB simulations of outdoor scenes.

Recent advances in computer vision, such as Neural Radiance Fields (NeRFs) and their corresponding Neural Signed Distance Functions (NeuS) have made producing 3D implicit functions from images much easier, but they have not yet been studied from the view point of numerical simulation. In this thesis, we present for the first time a 3D simulation using the Embedded Boundary method on a neural-SDF learned from images. In the process, we identify several challenges in bridging the two methods, including differences in how the computer vision community measures error/uncertainty and what types of error/uncertainty actually matter in an EB simulation, and present appropriate alternatives. In the following chapters, we describe several methods for learning 3D geometries from RGB images, discuss their suitability for scientific tasks, and do an in-depth analysis of their error and uncertainty and how those affect a physics simulation. We finally discuss this pipeline through the lens of a high-impact, real-world application: forest management. We reconstruct a forest scene using NeRF that is visibly much more appropriate for simulation than previous examples. We also discuss what challenges this real-world data presents and how the technology presented in this thesis can be used to answer real scientific questions.

Acknowledgments

This thesis marks the end of a thirteen year career as a student at UC Berkeley. Accordingly, there are dozens, if not hundreds, of people who have improved my time here in ways both big and small. I will attempt to name many of them here, but know that this list is not comprehensive.

I have to start with my day one, my freshman roommate, Bridget Vaughan. Thank you for joining me in our quest for eternal student-hood and all of the study sessions, application reviews, and commiseration that came with that. Looking forward to the next degree!

To my various undergraduate mentors: Dr. Daniel Greengard, who very patiently helped me through my first numerical analysis class, I know I said I just wanted to pass and never think about numerics again, but here we are. Thank you for giving me the base on which I built the rest of my career. Juan Francisco Esteva, Dr. Joyce Onyenedum, and the rest of the McNair Scholars Program. When I applied to the program, I was ready to give up on my dream of a PhD. You two were the first to tell me I could do it and have continued to support me every step of the way. Thank you to the program more broadly for providing mentorship, community, and financial support. I'm looking forward to the day that we can bring it back to campus to support future generations of Berkeley students.

To the community that formed in my undergraduate years and has supported me to this day: My roommates, Cecile Basnage, Emma Latham Jones, and Erika Cruz, my dear friends Alexis Flores-Betancourt, Areidy Beltrán-Peña, Amada Beltrán de Cornet, and Dr. Alexandra McCleary, and my homes away from home, the Berkeley Institute (Dr. Dena Fehrenbacher, Prof. Jonathan Shelley, Dr. Monica Mikhail, Prof. Karl van Bibber, and many more) and the Oakland Catholic Worker (Prof. Bob Lassalle-Klein, Luly Aranda McEtchin, Barbara Zavala, Chepe Zavala, Tulio Serrano, the late and greatly missed Tom Webb, and many more), and Chestnut Center (Marissa Ebora, Mia Ricafort, Mia Antonia, Dr. Helen Kwan, Carole DeCosse, Fr. Leo and everyone else) - thank you for giving meaning and joy to my life, for the laughter, the tears, and the long conversations and philosophical debates. You made Berkeley into a place that could feed my soul, not just my mind.

To my first research advisors: Prof. Per-Olaf Persson who introduced me to discontinuous-Galerkin methods, Dr. Rachel Slaybaugh who gave me my first software development opportunity on PyNE (and Ryan Feng who allowed me to drag him to that first hackathon), set me up with my first formal research experience at Oak Ridge, advised my master's thesis and introduced me to the world of supercomputing, and Dr. Bob Grove, Dr. Tom Evans, Dr. Tara Pandya, Dr. Steven Hamilton, and the rest of the ORNL Radiation Transport Group. While my current research interests have deviated a bit from those days, you all set me up the with fundamental skills in scientific computing that made this work possible.

To my mentors in AI & computer vision: Dr. Ethan van Andel, Dr. Mary Cameron, and

Kachi Okoronkwo from Hivemapper Inc. - thank you for giving me my first opportunity in computer vision. Not many were willing to take a chance on a computational physics student who barely understood what a neural net was. Thank you for giving me that chance. Dr. Radomir Mech, Dr. Matheus Gadelha, Dr. Thibault Groueix, and Shanayvia Lattimore from Adobe Research - thank you for your mentorship and support, and to the researchers more specifically for their contributions to Chapter 2 of this thesis. The LXAI mentorship program and my mentor Dr. Georgios Pavlakos - thank you for the coffee chats and support. Finally to my co-advisor Prof. Trevor Darrell - thank you for seeing something in this little research project bringing 3D vision to science that wasn't quite sure where it fit and shepherding it to this point. I'm looking forward to seeing how far we can go.

To the mathematicians: My co-advisor Prof. Phil Colella and the rest of the Applied Numerical Methods Group at Lawrence Berkeley National Lab (especially Dr. Dan Martin, Dr. Hans Johansen, Dr. Dan Graves, and Dr. Terry Ligocki). Thank you for your endless patience, understanding and support. Most of all thank you for keeping me honest, pushing me to critically think through my ideas, and never letting me get away with any hand-waving, AI magic.

To the many other professors and researchers I have had the privilege of crossing paths with: Prof. Jon Wilkening and Prof. Jonathan Shewchuk who sat on my qualifying exam and were very understanding as we had to last minute make it virtual in April 2020; Prof. Angjoo Kanazawa who is serving on the committee of this thesis and deserves credit for the initial push to combine BayesRays and NeuS which turned into Chapter 4; Dr. Derek Young from UC Davis who is my collaborator on forest reconstruction and contributed much to Chapter 5; Prof. Lee Bernstein who taught me everything I know about physics and believed in me at my lowest point; Prof. Oscar Dubón, Dr. Armando Fox, Prof. Ken Ribet, and Prof. Karl van Bibber (who deserves a place in quite a few of these paragraphs, but will get another shout-out here) who were always ready with a word of advice; and Dr. Natarajan Shankar who was generous enough to give me a chance and admit me to his Formal Methods Summer School despite my utter lack of experience, while I never pursued much formal methods after that, you arguably had the biggest impact of them all by introducing me to my now husband – Thank you all for your contributions big and small. Each of these relationships, while some of them shorter than others, have each had an outsized impact on my research path.

To my many peers and colleagues: Dr. Ben Caulfield and the residents of 1833 Cedar; Dr. Samantha Lewis, Dr. Emily Vu, Jessica Chow, and the Computational Neutronics Group (Dr. Mario Ortega, Dr. Vanessa Goss, Dr. Jessica Rehak, Dr. Sam Olivier, and the rest!); My ANAG office mates (Dr. Chris Bozhart, Dr. Colin Wahl, and Will Thatcher); The Vision & Language Group, my BAIR colleagues (especially Sarah Barrington, Neerja Thakkar, and Carlo Bosio), Dr. Frederick Warburg who collaborated with me on many of the ideas and software development in Chapter 4; Dr. Jane Wu and our undergraduate research assistants (in particular Arjun Rewari for his contributions to the paper that served as the basis for Chapter 5, and Rushil Desai for his work on NeuS uncertainty) – Thank you for your ideas, challenges, questions, support, and friendship.

To my Berkeley heroes: I would be remiss to forget the staff members who make this university run and most importantly (to me) helped make sure I didn't get caught in the cogs. Sara Harmon, Roxana Infante, Meltem Errol, Catalina Estrada, and most of all Ariana Castro - thank you for your infinite patience with every deadline reminder, email exchange, and special exception form. I was a non-traditional student who seemed to bump up against Berkeley's bureaucracy at every turn, without your creativity and persistence I know we would have never gotten here. And to my union UAW 2865 - thank you for picking up the cause when we ran out of options. I may not have been able to get university sanctioned maternity leave, but thanks to our union, future students shouldn't face that particular struggle.

To my village: The other big project of these past few years has been the birth and care of my daughter Zoraida (Zoey). Thank you to my strong village of people who aid me in her care: Rosa Top, Monica Vega, the lovely teachers at Imagina and Bowman. A very extra big thank you goes to those who cared for me as a child that now give that same love and attention to Zoey: Mary Beth Ricks, Erin Knight, Dr. Aimee Blaustein, Dr. Christina Kong, Dianna Zupp, A-ling Santiago, the late and dearly missed Dr. Fidelia Butt, and my childhood best friend Krista Abela. Being able to surround my daughter with the same exceptional love and support I had growing up has been such a privilege. I would not have had the time or peace of mind to focus on this research if not for all you.

To my mom, Andrea Ramirez – thank you for patience and willingness to pick up the phone to be with me in celebrating every win and overcoming every challenge. Thank you for picking up Zoey every time I have to work late, showing up at every school event I've had to miss, cleaning my apartment while I was in labor, and cooking for me while I healed. Thank you for all of your sacrifices to invest in my education and your encouragement and unwaivering support.

To my husband, Dr. Marcell Vazquez-Chanlatte, my partner in all senses of the word – Thank you for being my lab partner in countless classes. While I hope we never pair-program again, I will admit that you have made me a better engineer. Thank you for being my partner in research - the first person I bounce an idea off of, my harshest critic, and my biggest champion, and for agreeing to put a whiteboard on our wedding registry and be sys admin for our home compute resources. I am grateful for our constant flow of ideas, and the knowledge that whether I stay in academia or not, you have ensured that I will never be want for research support or stimulation. Thank you for being my partner in life, for setting an example as you wrote your dissertation and began your post-grad research career, and for holding down the fort at home while it's been my turn. Thank you for being my partner in

parenting, my favorite collaboration by far. Thank you for healing my heart and making the journey everything I dreamed it could be. Most of all thank you for being my best friend. I'm so lucky we get to do this together.

To my daughter, Zoraida – You have brought endless joy to our lives. Your morning cuddles and good night kisses are the best parts of my day, and your curiosity and kindness inspire me. I hope you pick this up many years from now and remember not the times that I couldn't play because I was too busy writing, but that your ideas have power, and when you put them to paper they can change the world. Thank you for letting me be your mom.

Contents

1 Introduction **1**
1.1 Implicit Functions & Level Sets 4
1.2 The Embedded Boundary Method 5
1.3 Neural Networks 10
1.4 3D Reconstruction from Images 13
1.4.1 Structure from Motion 13
1.4.2 Neural Methods 14

2 Learning SDFs from Single Images **17**
2.1 Single View Reconstruction 18
2.2 Monocular Depth Estimation 19
2.3 Projecting Disparity to 3D 20
2.4 Recovering Detail in 3D Shapes Using Disparity Maps 22
2.5 Experimental Results 23
2.5.1 Disparity Projection Sensitivity Studies 23
2.5.2 Full-Method Validation 24
2.6 Discussion & Limitations 30
2.6.1 Limitations of The Models Used 30
2.6.2 Limitations of SVR for Scientific Use Cases 31
2.7 Related Work 32
2.7.1 Monocular Depth Estimation 32
2.7.2 Single View Reconstruction 33
2.7.3 3D Diffusion and Shape Completion 34

3 Simulating Physics with NeuralSDFs **35**
3.1 Learning SDFs for Scientific Simulation 35
3.2 Methodology for Evaluating NeuralSDFs in EB Simulation 37
3.2.1 Geometry Quality Metrics 38

3.2.2 NeuralSDF Convergence 41
3.3 Image to Simulation on Complex Geometries 46
3.4 Conclusions 46

4 Uncertainty in NeuralSDFs **50**
4.1 Uncertainty in NeRF 50
4.1.1 Intuition 51
4.1.2 Spatial Deformation 51
4.1.3 Laplace's Approximation 53
4.1.4 Computing the Hessian 54
4.2 BayesRays with NeuralSDFs 56
4.2.1 Rendering Uncertainty 57
4.2.2 Uncertainty and Geometric Error 59
4.3 Future Work: Propagating Uncertainty in Embedded Boundary Simulation . 61

5 Applications to Forestry **62**
5.1 Forest Mapping 63
5.1.1 Structure-from-Motion (SfM) 63
5.2 Introducing NeRFs to the OFO 66
5.3 Climate and Environmental Impacts 68
5.4 Uncertainty in Forest Renderings 69
5.5 Future Work 70

6 Final Words **72**

Chapter 1

Introduction

Numerical simulation is an incredibly powerful tool that enables much of modern life. Constructing a new aircraft, planning a new city, preparing for a natural disaster, deciding where to plant crops, and so much more, often begin with a physical simulation of the scene or object of interest. Foundational to all of these simulations is some representation of the geometry. Much of the numerical methods literature takes for granted that the geometry is represented accurately. Given there are no perfect 3D computer models of most of the world around us, an accurate geometry seems like a big assumption, but up until recently, it was difficult to get even imperfect 3D representations of outdoor scenes. Also, even if high-quality 3D geometries were available, performing high-fidelity, 3D physics simulations is computationally intensive. For those reasons, much of the historical 3D numerical methods literature focused on simulations of smaller man-made objects where access to an accurate geometry can be reasonably assumed. For simulation of large-scale outdoor scenes simplifying assumptions were often made to reduce the geometry to 2D or otherwise simplify the problem.

However, significant progress in computer vision, high-performance computing, and numerical PDEs is finally coming together to make large scale, 3D, high-fidelity physics simulation of the natural world practical. Imagine driving a car with a dash-cam through a city, uploading that footage and then performing an earthquake simulation, or flying a drone over a forest, capturing video, and then simulating a wildfire through it. While this exact dream is not yet a reality, all of the building blocks are there. High quality 3D reconstruction models, semantic segmentation and material classification models, high-performance computing architectures, and 3D numerical PDE solvers are all improving rapidly, and with time I believe a robust version of this pipeline will be possible. But while each of these areas of research are being thoroughly studied and improved individually, what has not yet been fully explored, is how all of these methods will interact together.

To compute solutions to a PDE in complex geometries, there are three essential components:

(1) A method for constructing a mathematical representation of the geometry from data; (2) A method for constructing discretizations given the representation of the geometry in (1); (3) A method for constructing robust and efficient solvers for the discretizations in (2). The Embedded Boundary method (EB) provides robust, automatic handling of steps (2) and (3) as demonstrated in the Cart3D code [1] which is used to simulate aerodynamics for realistic geometries in a completely automated fashion and in the work in [14, 18, 83, 84] which provide examples of constructing robust solvers given an implicit function representation of the geometry. A significant open problem in the EB methodology is step (1), mainly because the data that is the starting point for generating the required mathematical representation of discretizations is so diverse. In [1], this problem is solved for the case of surface triangulations coming from CAD systems. In the other methods cited here, the generation of discretizations and robust solvers is solved by generating in step (1) implicit function representations of geometry from whatever data is available. Notably, in [83] 2D boundaries were determined from images via level set methods, but there has not yet been work performing an Embedded Boundary simulation on a 3D geometry learned from images.

This thesis represents the first exploration of a pipeline from RGB images to 3D Embedded Boundary PDE simulation using neural networks. In the following chapters, we explore several tools from the AI and computer vision community that learn 3D implicit functions from RGB images. While most of these tools were developed with visual reconstruction quality in mind, we instead probe their suitability for scientific tasks, and do an in-depth analysis of their error and uncertainty and how those affect a physics simulation. Through experiments, we demonstrate the failings of Chamfer Distance, a commonly used 3D reconstruction metric, to capture simulation error, and instead present our own metric which combines the two qualities most relevant to EB simulation: SDF fidelity and surface smoothness. This opens up a new class of application domains where such image data is the best, or even the only data available from which to derive geometric information. We more deeply explore one of these high-impact domains, forest management, use neural methods to reconstruct a forest scene, improving on the current forest 3D models available, discuss what challenges this real-world data presents and how the technology presented in this thesis can be used to answer real scientific questions.

As this is a highly interdisciplinary work, in the rest of chapter 1, we provide a quick background (as well as references for additional reading) on several of the methods that will be discussed. In chapter 2, we start the original content with how to obtain 3D computer geometries of the world around us. Methods such as LiDAR and Structure-from-Motion [81] have long been used for this purpose, but they are often noisy and can at times be impractical. More recently, work has emerged using neural networks to predict 3D geometry from images [59, 100]. Images are much easier to obtain through existing satellite imagery, lightweight video drones, or even an individual with a smart phone, so reliably being able to derive 3D geometries from these images greatly opens up the world of possibilities. The most successful neural image-to-3D techniques use implicit functions to represent the 3D geometry. In chapter 2, we describe one such method for learning 3D SDFs from single images.

But despite the rich literature and progress made by the computer vision community, we have not yet seen widespread adoption of these neural geometries in numerical PDEs. There are many possible reasons for this, but one appears to be the lack of rigorous study around the accuracy of these methods, particularly how that accuracy affects the accuracy and stability of numerical solvers. Neural-implicit representations have inherent uncertainty. In graphics and other visual applications one can easily inspect with their eyes if a reconstruction is suitable enough for use, but in scientific applications it is not so easy.

In chapter 3, we explore how neural SDFs interact with a numerical PDE solver. What kinds of errors do these methods make in the representation of the geometry? How do numerical solvers react to these errors? How should an applied mathematician understand these tools in the context of an image-to-simulation pipeline? What would such a pipeline even look like? There are many methods to create 3D geometries of the real world, and even more methods to solve PDEs on 3D geometries, so it is difficult to make claims about the interaction of the two classes of methods more broadly, but the boom in neural implicit methods for 3D reconstruction provides a suitably interesting and narrow lens to begin to study the issue. Implicit functions have nice geometrical properties that can result in higher quality 3D surface representations. They also have properties that can facilitate discretization for numerical methods. The Embedded Boundary method [39] is an approach for finite-volume discretization that uses implicit surface representations as geometric inputs. Using the implicit surface function intersected with a Cartesian grid, we can calculate to high accuracy the volume fractions and centroids of the cells cut by the boundary of the geometry and use that information to perform a finite-volume fluid simulation. Chapter 3 of this thesis addresses the questions posed at the beginning of this paragraph by providing a systematic analysis of error measurements in 3D reconstruction, proposing a new metric that is more correlated with physical simulation error; describing for the numerical analyst the various parameters that can be tuned in NeuS reconstruction and analyzing the convergence properties of these parameters in both geometry and in PDE solution; and providing a full workflow of the image to simulation pipeline on a complex geometry.

In Chapter 4, we explore and define uncertainty in neuralSDFs. Exploring the effect of geometric error in the numerics pipeline leads us to wonder if there is a way to better understand and control those errors before we begin simulation. Of course, without a ground truth to compare to, it is difficult to have a concept of “error,” so we instead try to quantify some estimation of uncertainty. An uncertainty estimate of a geometry is a powerful tool that combined with our earlier understanding of the effect of geometric error on simulation can provide insight into a simulation’s quality and alert scientists if a geometry is going to be suitable for their needs or if additional data or other remediation is necessary. Chapter 4 discusses in detail uncertainty quantification for neural-implicit methods and presents a novel method for predicting uncertainty in NeuralSDFs.

In Chapter 5 we describe how these methods can be applied in real-world application. We chronicle the collaboration with the Open Forest Observatory in their effort to create a freely available 3D library of forests in the Western United States. This project is a perfect example of why applications research, translating state-of-the-art computational methods to scientific problems, is vital. Real world data presents difficulties often overlooked by initial methods research. This chapter serves both as a proof of concept of the power and feasibility of our image-to-simulation pipeline as well as a how-to manual for applying neural reconstruction methods to large-scale outdoor datasets.

This thesis presents and walks the reader through the image-to-simulation pipeline; first presenting a novel method in using neural networks to learn 3D signed-distance functions from single images, next exploring how neural signed distance functions interact with physical simulation through the Embedded Boundary method, then exploring uncertainty in neural signed distance functions, and finally exploring how all three classes of methods (NeuralSDFs, Embedded Boundary simulation, and uncertainty prediction) can be used together in a real-world setting, the Open Forest Observatory, to answer pressing scientific questions. We synthesize methods from the computer vision, scientific computing, and applied science communities, identifying unexplored gaps at the intersections. We also present new methods to fill some of these gaps and give a blueprint for a new interdisciplinary tool chain that could make the simulation of real-world scenes much easier.

1.1 Implicit Functions & Level Sets

In both 3D reconstruction and simulation we are interested in representing a *surface* in 3D Euclidean space. Before we delve into deeper discussion of how to generate these surfaces, let us briefly define some useful geometric concepts. Fundamentally, all surfaces are subsets of $\mathbb{E}^3$, however not all subsets are surfaces - they must also be smooth and two-dimensional. We can also define a surface with an implicit function. For example the equation $x^2 + y^2 + z^2 = 1$ implicitly defines the spherical surface of radius 1.

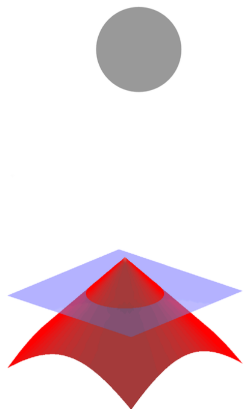


Figure 1.1: An illustration of the signed distance function of a circle.

Some types of implicit surface functions have other special properties. The work contained in this thesis primarily involves *implicit signed distance functions* (SDFs) 1.1. A signed distance function returns the shortest distance to a surface given a point, along with a sign indicating whether the point is inside or outside the surface. In the example of a sphere of radius 1, the equation $f(x, y, z) = x^2 + y^2 + z^2 - 1$ would be the implicit signed distance function, with the *zero level set* of that function, $x^2 + y^2 + z^2 - 1 = 0$ representing the surface. A useful property of SDFs is that where differentiable, their gradients always satisfy the Eikonal equation: $|\nabla f| = 1$. While not all implicitly defined functions are Signed Distance Functions (for example, one could implicitly define a geometry with sharp corners where the gradient is undefined), there exist algorithms to smooth such geometry into a signed distance function [18].

Implicit functions have been well explored as a tool for numerical analysis of surfaces and shapes. First introduced in [67] for computing the dynamics of propagating fronts, level-set methods provide a framework for performing numerical computations involving curved surfaces on a Cartesian grid. This method has been used for a variety of applications, including numerical PDEs, computational geometry, and shape/surface reconstruction from images. We will discuss in greater detail a specific numerical method of this flavor called the Embedded Boundary method in the following section, but make note of the level set work here, to emphasize that the current popular methods in numerical PDEs and 3D reconstruction that we are exploring in this thesis are rooted in the same mathematical theory.

1.2 The Embedded Boundary Method

Signed distance functions have many uses, one being in numerical simulation of partial differential equations (PDEs) via the Embedded Boundary Method [39]. The Embedded Boundary is a finite volume method that involves intersecting a complex geometry (represented by an implicit signed distance function) with a Cartesian grid. The two are laid on top of each other resulting in “cut cells" at the surface (shown in Fig. 1.2), where the boundary function “cuts" the Cartesian grid cells. For this reason, it is also referred to as a cut-cell method.

From there, the volume information necessary for a finite volume computation can be recursively constructed (as seen in Fig. 1.4), starting by calculating where the geometry intersects the grid on the 1D cell edges, then calculating the centroid or moment of the covered part of the geometry, and then using that information to build up to the 2D cell faces and then the 3D volume.

The input geometry always takes the form of an implicit function (and often an implicit signed distance function), but that function can be derived from a variety of sources: Sometimes we know the function analytically (i.e. shapes such as a sphere or cube), sometimes we can also use those shapes to construct more complicated geometries, other approaches include ray casting towards a surface mesh and calculating the SDF that way. In this thesis we focus on learning the SDF from input images. Earlier work [83] has been done inferring a 2D surface from a single image for use with EB, but this is the first time we have explored

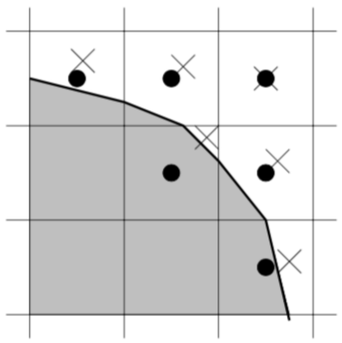


Figure 1.2: Control volume formation on a 2D example. The gray geometry represents some object, and we are interested in simulating the mechanics of fluid flowing around it in the white region. The EB method intersects the more complicated grey geometry, with a simpler Cartesian grid, so that we may use the finite volume method to calculate the physics solution. But instead of performing a finite volume calculation on the entire volume of the cell centered at the black circles, we perform the calculation on just white region of the cell whose centers of mass are represented by the x's. In cells that are completely white, those two centers of mass line up, so no additional computation is needed, but in the boundary cells we must use the machinery outlined below to calculate the correct center of mass.

3D surfaces from images.

In this section we will describe the process of calculating an EB geometry as outlined in [84]. The Embedded Boundary method starts by intersecting an irregular geometry of interest, Ω, with a Cartesian mesh with cells i. We define that mesh as:

$$\Upsilon_i = [(\boldsymbol{i} - \frac{1}{2}\boldsymbol{u})h, (\boldsymbol{i} + \frac{1}{2}\boldsymbol{u}h)], \boldsymbol{i} \in \mathbb{Z}^D \tag{1.1}$$

where D represents the dimensional of the problem, h is the grid spacing, and $\boldsymbol{u}$ is a vector whose entries are all one. We then obtain control volumes $V_i = \Upsilon_i \cap \Omega$ and faces $A_{\boldsymbol{i},d\pm} = A_{\boldsymbol{i}\pm\boldsymbol{e_d}/2}$. We can use the divergence theorem to model the divergence of the field as the surface flux which is approximately equal to the balance of the various cell-edge fluxes and the flux through the boundary.

$$\int_V \nabla \cdot \mathbf{F} dV = \sum_{d=1}^{D} \left(\int_{A_{d+}} F_d dA - \int_{A_{d-}} F_d dA + \int_{A_B} F_d n_d(\mathbf{x}) dA \right). \tag{1.2}$$

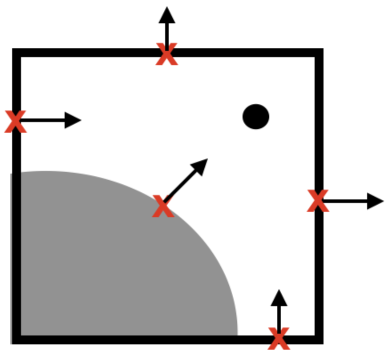


Figure 1.3: An example of a single boundary cell showing flux in and flux out in addition to the various face and boundary aperatures and normals. Here we have all the geometric quantities necessary for Eq. 1.4. The red x's represent the midpoints (or centroids) of the one-dimensional boundaries, the arrows are the normals, and the black dot is the centroid of the volume.

We can discretize this equation via the midpoint rule to obtain:

$$\int_V \nabla \cdot \mathbf{F} dV \approx \frac{1}{\kappa_i h}\left(\sum_{\pm=+,-}\sum_{s=1}^{d} \pm\alpha_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_s} F^s(\boldsymbol{x}_{\boldsymbol{i}\pm\frac{1}{2}\boldsymbol{e}_s}) + \alpha_{\boldsymbol{i}}^B \boldsymbol{n}_{\boldsymbol{i}}^B \cdot \vec{F}(\boldsymbol{x}_{\boldsymbol{i}}^B)\right) \tag{1.3}$$

where $\kappa_{\boldsymbol{i}} = |V_{\boldsymbol{i}}|h^{-d}$ are the volume fractions, $\alpha_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_s} = |A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_s}|h^{-(d-1)}$ are the face apertures, $\alpha_{\boldsymbol{i}}^B = |A_{\boldsymbol{i}}^B|h^{-(d-1)}$ are the boundary aperatures, $\boldsymbol{n}_{\boldsymbol{i}}^B = \frac{1}{|A_{\boldsymbol{i}}^B|}\int_{A_{\boldsymbol{i}}^B} \boldsymbol{n}^B dA$ is the average outward normal to the boundary, and $\boldsymbol{n}^B$ is the outward normal to $\partial\Omega$ defined for each point on $\partial\Omega$. Each of these quantities are illustrated in Fig. 1.3.

The midpoint rule provides a second-order accurate approximation. We can extend this to high orders of accuracy. Given a sufficiently smooth function ψ,

$$\psi(\mathbf{x}) = \sum_{|\mathbf{q}|<Q} \frac{1}{\mathbf{q}!}\psi^{(\mathbf{q})}(\overline{\mathbf{x}})(\mathbf{x}-\overline{\mathbf{x}})^{\mathbf{q}} + O(h^Q) \tag{1.4}$$

we can use a Taylor expansion to approximate $\boldsymbol{F}$ in the neighborhood of $\overline{\mathbf{x}}$ to arbitrary order

Q:

$$\int_v (\nabla \cdot \mathbf{F}) dV = \sum_{|\mathbf{q}|<Q} \sum_{d=1}^{D} \frac{1}{\mathbf{q}!} \Bigg(F_d^{(\mathbf{q})}(\overline{\mathbf{x}}_{d+}) \int_{A_{d+}} (\mathbf{x} - \overline{\mathbf{x}}_{d+})^{\mathbf{q}} dA$$
$$- F_d^{(\mathbf{q})}(\overline{\mathbf{x}}_{d-}) \int_{A_{d-}} (\mathbf{x} - \overline{\mathbf{x}}_{d-})^{\mathbf{q}} dA + F_d^{(\mathbf{q})}(\overline{\mathbf{x}}_B) \int_{A_B} (\mathbf{x} - \overline{\mathbf{x}}_B)^{\mathbf{q}} n_d(\mathbf{x}) dA \Bigg) + O(h^{Q+D-1}) \tag{1.5}$$

Given a point in space $\overline{\mathbf{x}}$ and a D-dimensional integer vector $\mathbf{p}$, we define the $\mathbf{p}$-th moment of the faces $A_{d\pm}$ and the boundary A_B relative to the point $\overline{\mathbf{x}}$

$$m_{d\pm}^{\mathbf{p}} = \int_{A_{d\pm}} (\mathbf{x} - \overline{\mathbf{x}}_{d\pm})^{\mathbf{p}} dA \tag{1.6}$$

$$m_{B,d}^{\mathbf{p}} = \int_{A_B} (\mathbf{x} - \overline{\mathbf{x}}_B)^{\mathbf{p}} n_d(\mathbf{x}) dA \tag{1.7}$$

Using the moment definitions,

$$\int_v (\nabla \cdot \mathbf{F}) dV = \sum_{|\mathbf{q}|<Q} \sum_{d=1}^{D} \frac{1}{\mathbf{q}!} (F_{d+}^{(q)} m_{d_+}^{\mathbf{q}} - F_{d-}^{(q)} m_{d_-}^{\mathbf{q}} + F_B^{(q)} m_{B,d}^{\mathbf{q}}) + O(h^{Q+D-1}). \tag{1.8}$$

This equation provides a relationship between the divergence and the moments, but it also gives us a path towards calculating the moment values. We do this by considering a volume V at cell i and letting $\overline{\mathbf{x}}$ be some point in the cell (we will set $\overline{\mathbf{x}} = 0$ without loss of generality). If we choose $\mathbf{F} = \mathbf{x}^{\mathbf{q}} \mathbf{e}_d$, we get D equations of the form

$$q_d \int_V \mathbf{x}^{\mathbf{q}-\mathbf{e}_d} dV = \int_{A_{d+}} \mathbf{x}^{\mathbf{q}} dA - \int_{A_{d-}} \mathbf{x}^{\mathbf{q}} dA + \int_{A_B} \mathbf{x}^{\mathbf{q}} n_d dA. \tag{1.9}$$

These face integrals are the moments shown in the picture in Fig. 1.3. We approximate the last integral using a Taylor series to order of accuracy S:

$$\int_{A_B} \mathbf{x}^{\mathbf{q}} n_d(\mathbf{x}) dA = \sum_{|\mathbf{s}|<S} \frac{1}{\mathbf{s}!} \partial^s n_d(0) m_B^{\mathbf{q}+\mathbf{s}} + O(h^{|\mathbf{q}|+D+S-1}). \tag{1.10}$$

Combining, we use the following equation to calculate the moments m_v, m_B, and $m_{d\pm}$.

$$q_d m_v^{\mathbf{q}-\mathbf{e}_d} - n_d(0) m_B^{\mathbf{q}} = m_{d+}^{\mathbf{q}} - m_{d-}^{\mathbf{q}} + \sum_{|s|<S} \frac{1}{\mathbf{s}!} \partial^s n_d(0) m_B^{\mathbf{q}+\mathbf{s}} + O(h^{|\mathbf{q}|+D+S-1}). \tag{1.11}$$

We can calculate these values recursively as shown in Fig. 1.4. Each moment can be written in terms of the lower dimensional moments, going down to 1D which are easily calculated via numerical root finding.

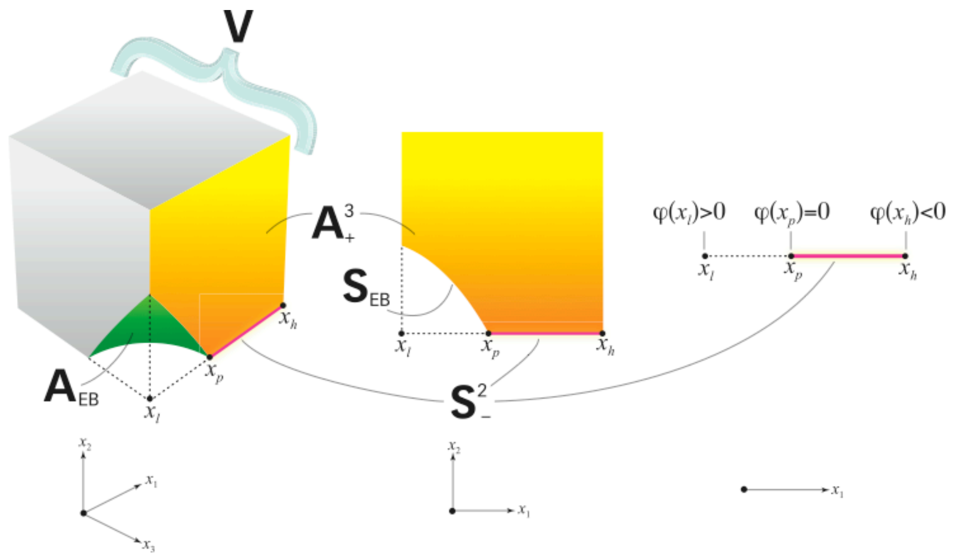


Figure 1.4: Recursive moment building from 1D [51]. For a 3D finite-volume simulation, we need to compute the 3D control volume on the left. We do this recursively, using Eq. 1.11 to build up a system of equations based on the 2D moments that can be solved via least squares to determine the 3D moments. The 2D moments are in turn dependent on the 1D moments, which can be easily found via a simple root-finding algorithm. It is important to note that accuracy is lost as we move to higher dimensions, so we must calculate lower dimensions to higher order of accuracy than we will need for the higher dimensional moments.

1.3 Neural Networks

In this work, the signed distance functions we consider are not only different because they are derived from images, but because they are represented not as a traditional implicit function, but as a neural network which predicts what the output of the signed distance function would be given a point as input. There are a variety of ways in which these networks can be trained (via supervised learning on matching 2D images and 3D ground truth SDFs, by rendering images and comparing to ground truth images, by looking at a base point-cloud, and more), but all the neural networks we will talk about in this thesis share the same basic structure. Because neural networks are not yet widely used in combination with numerical methods, we give a quick introduction to the basics following Hastie, Tibrshirani, and Friedman's exposition in *The Elements of Statistical Learning* [38].

A multilayer perceptron (MLP) is a function mapping $X \in \mathbb{R}^{n_{\text{input}}}$ to $\mathbb{R}^{n_{\text{output}}}$ parameterized by m linear maps, m-vectors, and $m+1$ non-linear maps. Each linear map is called a linear layer and is parameterized by a matrix of real numbers. Specifically, for the i-th linear layer, weight matrix $A_i \in \mathbb{R}^{d_i \times d_{i+1}}$ maps $\mathbb{R}^{d_i} \to \mathbb{R}^{d_{i+1}}$. The i-th linear layer is associated with a bias vector, $b_i \in \mathbb{R}^{d_{i+1}}$ and non-linear map (also known as an activation function), $\sigma_i : \mathbb{R}^{d_{i+1}} \to \mathbb{R}^{d_{i+1}}$, that applies a non-linear function, such as a sigmoid, element-wise. The final non-linear map, g, maps $\mathbb{R}^{d_{m+1}} \to \mathbb{R}^{n_{\text{output}}}$. Examples for g include the identity function, the softmax function, and the max function. A MLP, f, composes these maps as follows:

$$\begin{aligned} f(X) &= g(z_{m+1}) \\ z_{i+1} &= \sigma_i(A_i z_i + b_i) \\ z_1 &= X \end{aligned} \tag{1.12}$$

The linear layer plus its corresponding non-linear layer (i.e. the function producing z_i) are together referred to as a single perceptron layer. Layers whose output is not directly consumed by g are referred to as hidden layers. The output of a hidden layer, z_i, is referred to as a latent variable.

MLPs are often visualized as directed acyclic graphs (see Fig. 1.5) where each incoming edge is weighted by entries in the corresponding weight matrix, A_i, and each node represents the variable produced by summing the inputs weighted by the edges, applying the bias and non-linearity. Linear layers correspond to collections of edges feeding into nodes of the same depth. A hidden layer corresponds to a collection of interior nodes at the same depth along with their edges. For example, in Fig. 1.5 the red nodes correspond to the latent variables of a single hidden layer. Eq. 1.13 expands Eq. 1.12 for a single hidden layer to make the node-level semantics explicit:

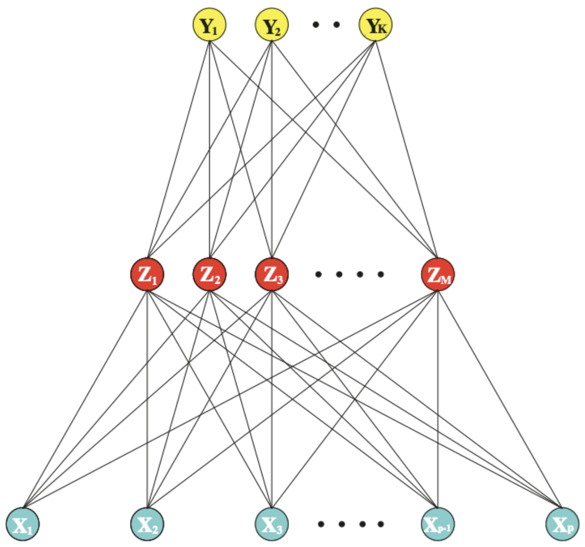


Figure 1.5: A schematic of the multilayer perceptron [38].

$$
\begin{aligned}
f(X) &= g(T) \\
T &= \beta_0 + \beta^T Z_0 \\
Z_{0m} &= \sigma_0(\alpha^0_{0m} + {\alpha^0_m}^T Z_1), m = 1, \ldots, M \\
Z_{1n} &= \sigma_1(\alpha^1_{0n} + {\alpha^1_n}^T X), n = 1, \ldots, N,
\end{aligned} \tag{1.13}
$$

where $f(X)$ is the function mapping inputs to outputs (for example a signed distance function), $(Z_0 = (Z_{00}, Z_{01}, \ldots, Z_{0m})$ and $Z_1 = (Z_{00}, Z_{01}, \ldots, Z_{0n})$, σ_0, σ_1 are activation functions, and α_i and β_i are biases of the neural network (e.g. elements of b_i from above). T corresponds to z_{m+1} (e.g. the last latent variable to which g is applied).

Neural networks such a MLPs are typically created by a process called “training”. When “training" the neural network, we search for weights that best fit the training data. The complete set of weights is denoted by θ which in the single layer case consists of,

$$\alpha_{0m}, \alpha_m; m = 1, 2, \ldots, M \quad M(p+1) \quad \text{weights,} \tag{1.14}$$

$$\beta_0, \beta \quad (M+1) \text{ weights} \tag{1.15}$$

Next, a loss function $\mathcal{L}(\theta)$ is selected. This is some measure of the error between the training data Y and the result of the network $f(X)$. An example loss function is the sum-of-squared errors:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N}(y_i - f(x_i))^2, \tag{1.16}$$

Typically to minimize $\mathcal{L}(\theta)$ one uses a variant of gradient descent. The gradient of the loss with respect to the neural network weights is typically computed via automatic differentiation. For exposition, we symbolically illustrate back propagation on a single layer MLP, a common automatic differentiation algorithm. Let $z_{mi} = \sigma(\alpha_{0m} + \alpha_m^T x_i)$ and let $z_i = (z_{1i}, z_{2i}, \dots, z_{Mi})$. The derivatives of $R(\theta)$ are

$$\frac{\partial \mathcal{L}_i}{\partial \beta_m} = -2(y_i - f(x_i))g'(\beta^T z_i)z_{mi}, \tag{1.17}$$

$$\frac{\partial \mathcal{L}_i}{\partial \alpha_{m\ell}} = -2(yi - f(x_i))g'(\beta^T z_i)\beta_m \sigma'(\alpha_m^T x_i)x_{i\ell}. \tag{1.18}$$

A gradient descent update at the $(r+1)$-st iteration looks like

$$\beta_m^{(r+1)} = \beta_m^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial \mathcal{L}_i}{\partial \beta_m^{(r)}}, \tag{1.19}$$

$$\alpha_{m\ell}^{(r+1)} = \alpha_{m\ell}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial \mathcal{L}_i}{\partial \alpha_{m\ell}^{(r)}}. \tag{1.20}$$

where γ_r is the learning rate.

Define a quantity $\delta_i = -2(y_i - f(x_i))g'(\beta_k^T z_i)$ which represents the loss at the output layer. This can be used to define another quantity

$$s_{mi} = \sigma'(\alpha_m^T x_i)\beta_m \delta_i \tag{1.21}$$

which is the loss at the hidden layer. These are also known as the back-propagation equations and can be used to rewrite the derivatives as

$$\frac{\partial \mathcal{L}_i}{\partial \beta_m} = \delta_i z_{mi} \tag{1.22}$$

$$\frac{\partial \mathcal{L}_i}{\partial \alpha_{m\ell}} = s_{mi} x_{il}. \tag{1.23}$$

With the back-propagation equations we can implement the gradient descent step shown above in a two pass algorithm. First we perform a forward pass where the current weights are fixed and the predicted values $\hat{f}(x_i)$ are computed using Eq. 1.13. In the backwards pass the δ_is are computed and then put into Eq. 1.21 to get s_{mi}. Then both losses are used to compute the desired gradients for the next step of gradient descent.

1.4 3D Reconstruction from Images

The problem of constructing a 3D geometry from multiple images has been long-studied in the computer vision literature. For use with the embedded boundary method, we are particularly interested in producing 3D implicit signed distance functions from several images, and we will briefly summarize several categories of approaches for obtaining 3D implicit SDFs. The first involves uses only classical techniques, constructing a 3D point-cloud using structure-from-motion (SfM), meshing that point-cloud, and then calculating a signed distance function from that mesh. The second category encompasses neural approaches which involve learning the SDF directly from the input images. The neural approaches can be further divided into two categories: large-data supervised methods and self-supervised methods. Large-data supervised methods require large datasets of image and 3D ground truth pairs to train on. In Chapter 2, we discuss many of these methods. For many scenes and objects of interest in scientific computing, we do not have access to this kind of data, so we turn to methods that use only information obtained in the input images themselves as supervision. Neural Radience Fields (NeRFs) are one such methods, so we introduce NeRF and its various SDF-based derivatives in detail, as they are currently the best suited for this type of work.

1.4.1 Structure from Motion

Structure-from-Motion (SfM) [81] is probably the best known reconstruction method in the scientific community. It is based on the principle that humans perceive information about 3D structure by moving around an object. Given a set of 2D images of an object or scene, matching features are tracked in each image. These features are often things like corners or line segments. Using these features we can triangulate to estimate camera locations and orientations and produce a 3D point-cloud of (x, y, z) coordinates of these features. There are several open-source implementations and variations on this technique, but in this paper we use COLMAP [81] as our SfM library.

There exists some work in the community using SfM to reconstruct a scene of scientific interest and use that to answer scientific questions [110], yet to our knowledge, there has not been much work doing full scale PDE simulations with these geometries. This is likely because high fidelity PDE simulations often require higher fidelity geometries than we usually get from SfM. For example, with a drone flying over a forest capturing images SfM might be able to produce a good canopy map, but it cannot fill in novel views below the canopy. There is also noise present in SfM generated point-clouds that can create sharp corners and changes in surface geometry that are particularly challenging to handle in EB simulation. We will discuss this problem in more depth in Chapter 3.

SfM produces a 3D point cloud. To convert a point-cloud to an SDF, the classical approach

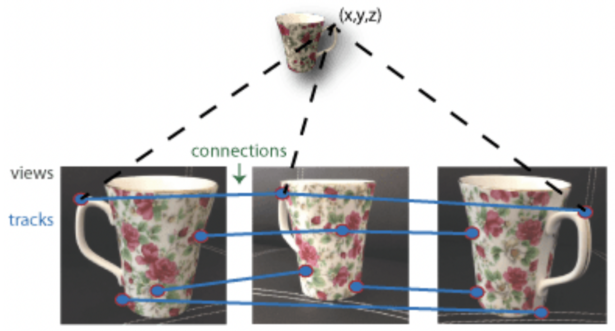


Figure 1.6: An illustration of feature mapping as implemented in the MATLAB SfM package.

is to first mesh the point-cloud and then ray-cast to calculate the SDF. There are many standard point-cloud to surface-mesh algorithms such as Poisson Surface Reconstruction [42]or Ball-Pivot [5]. All of these methods are very useful in creating geometries for simulation, but they each introduce new error and biases, that need to be tracked and better understood.

1.4.2 Neural Methods

Neural Radiance Fields (NeRFs)

A Neural Radiance Field (NeRF) is a neural network, f, trained on a set of input images, $\mathcal{I}$, with parameters, θ. The trained network acts as a function that when given a 3D point location in space, $\boldsymbol{x}$, and a viewing direction, d, returns a color, $\hat{C}$:

$$f_\theta(\boldsymbol{x}, d) = \hat{C} \tag{1.24}$$

That color is determined using volume rendering. For any given point, the volume density σ_i and color c_i are predicted using a coordinate MLP and the color of the pixel viewed at a particular angle is predicted by summing along the viewing-angle ray.

$$\hat{C} = \sum_{i=1}^{n} w_i c_i \tag{1.25}$$

where the weights are $w_i = T_i \alpha_i$, the opacity of the i-th ray segment is $\alpha_i = 1 - exp(-\sigma_i \delta_i)$, the distance between adjacent samples $\delta_i = t_{i+1} - t_i$, and the accumulated transmittance is $T_i = \prod_{j=1}^{i-1}(1 - \alpha_j)$. The parameters are determined by minimizing the loss between the predicted color, $\hat{C}$, and the true color seen in the input images, $C_g t$.

$$\mathcal{L} = ||\hat{C} - C_{gt}||_2^2 \tag{1.26}$$

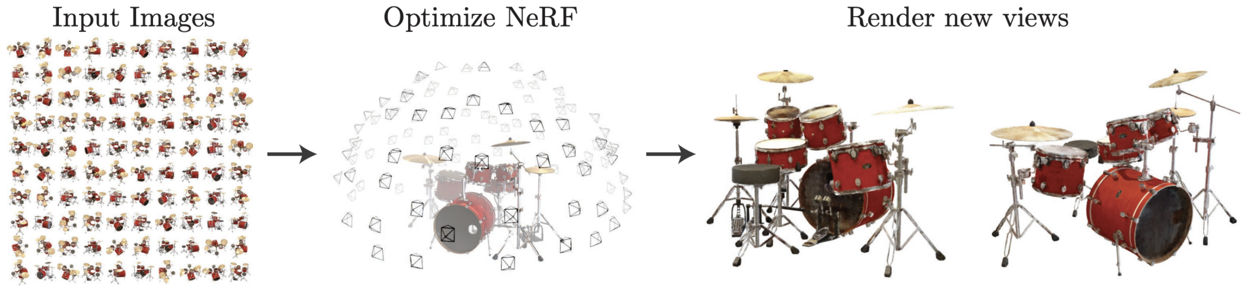


Figure 1.7: A visual explanation of the NeRF training process from [60]

Fig. 1.7 provides a visual explanation of the NeRF process. A collection of input images are given along with the viewing angle from which they were taken (often determined using SfM). The NeRF is an interpolation function that allows us to realize new views not included in the initial input data. NeRF, and it's numerous derivatives have experienced widespread adoption in 3D rendering, but NeRF by itself is a 2D rendering of views, not truly a 3D geometry. Using marching cubes, we can derive where the surface of the object is and obtain a mesh or a point-cloud, but these geometries are often not of the same high visual quality as the NeRF.

Neural Signed Distance Functions

NeuS [100] was developed to address the problem of poor-quality meshes being derived from NeRFs. They add in a signed-distance function prediction network and then render the surface of that SDF in the loss. While this method was designed to produce higher quality mesh, and is typically studied and used for that purpose (in experiments with these methods, the SDF is always rendered and it is the mesh accuracy that is discussed, not the SDF), we see great value in directly predicting the SDF.

NeuS uses the same color function (Eq. 3.2), but re-defines the weights as the normalized s-density, ϕ_s:

$$w(\boldsymbol{x}) = \frac{\phi_s(f(\boldsymbol{x}))}{\int_0^{+\infty} \phi_s(f(\boldsymbol{x}))dx} \tag{1.27}$$

where $f(\boldsymbol{x})$ is the signed-distance function. The s-density is a probability density function also known as the *logistic density distribution*

$$\phi_s(\boldsymbol{x}) = \frac{e^{-\boldsymbol{x}}}{(1+e^{-\boldsymbol{x}})^2} \tag{1.28}$$

The s-density is the derivative of the Sigmoid function, $\Phi_s(\boldsymbol{x})$, i.e. $\phi_s(\boldsymbol{x}) = \Phi'_s(\boldsymbol{x})$.

$$\Phi_s(\boldsymbol{x}) = (1 + e^{-\boldsymbol{x}})^{-1} \tag{1.29}$$

For ease of computation, we can also think of the weights as the product of the transmittance, T_i, and opacity, α_i as defined in NeRF, $w_i = T_i\alpha_i$. In this case transmittance is defined to be the same as in NeRF, but the opacity comes out to

$$\alpha_i = max\left(\frac{\Phi_s(f(\boldsymbol{x}_i)) - \Phi_s(f(\boldsymbol{x}_{i+1}))}{\Phi_s(f(\boldsymbol{x}_i))}, 0\right). \tag{1.30}$$

The full derivation of this equation is provided in the supplementary material of the NeuS paper [100].

For supervision, NeuS uses the same color loss equation as NeRF (Eq. 4.1), but they add in a regularization term, the Eikonal term $||\nabla f(\boldsymbol{x})||_2 = 1$, which is applied to points randomly sampled near the surface, enforcing that the predicted distances locally form a signed distance field. When it was released, NeuS was shown to produce better meshes on sets of test objects than NeRF, the major disadvantage at the time was the speed. Two variants, Neus2 [101] and NeuralAngelo [49] were released later that follow the same rendering scheme, but take advantage of the new neural graphics primitives in instant-ngp [63] and other computational tricks to greatly improve speed. In this thesis, we perform experiments on the neus-facto and neus-angelo implementations in nerfstudio [92], but results should be generalizeable to all methods in the NeuS family.

Chapter 2

Learning SDFs from Single Images

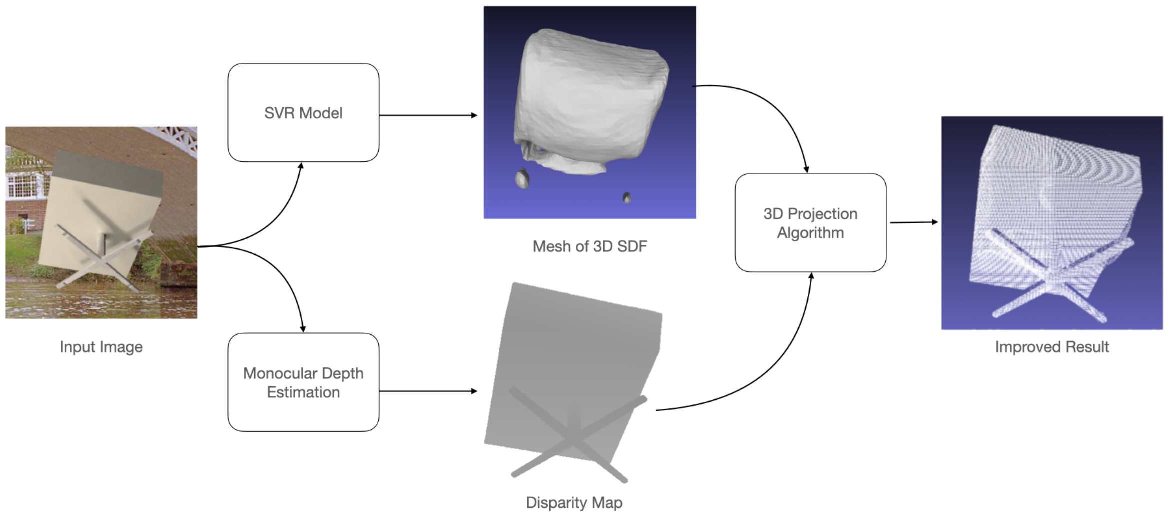


Figure 2.1: Our method projects the output of a monocular depth estimator to 3D, using it to improve the appearance of a 3D geometry from a single view reconstruction model.

In this chapter, we present a novel method[1] for improving the detail in signed-distance

[1]This work was conducted from mid-2021 to mid-2022 (roughly three years before the writing of this

functions of objects learned from single images. We start with an SDF of an object learned from an image that is unsatisfactory in its result. See Fig. 2.1 as an example. From the SDF we can visualize a 3D mesh of the chair, but we see it is lacking all of the detail of the legs and wheels. Our method fixes this problem, using the output of two different types of neural networks: A point-cloud sampled from the SDF of a single-view reconstruction network, Λ, and disparity map, d given by a monocular depth estimation network. We then learning camera parameters $\omega = s, t, fov, z_t, d$ which allow us to project the disparity map into a 3D point cloud, Ψ. To approximate the parameters we fix the disparity map, d, and use stochastic gradient descent to minimize the Chamfer distance (defined below) between the projected point-cloud, Ψ, and the visible portion of the SVR-generated point-cloud, Λ_{vis}.

$$\mathcal{L}_{fitting} = \frac{1}{|\Psi|} \sum_{\boldsymbol{x} \in \Psi} \min_{\boldsymbol{y} \in \Lambda_{vis}} ||\boldsymbol{x} - \boldsymbol{y}||_2^2 + \frac{1}{|\Lambda_{vis}|} \sum_{\boldsymbol{y} \in \Lambda_{vis}} \min_{\boldsymbol{x} \in \Psi} ||\boldsymbol{x} - \boldsymbol{y}||_2^2$$

$\boldsymbol{x}$ and $\boldsymbol{y}$ are 3D points in the point-cloud Λ_{vis}.

The intuition for the approach comes from the assumption that the initial prediction is roughly the right general shape and size as the desired object, and is only lacking in detail. By fitting the disparity map to the initial visible points, we should recover camera parameters that lead to a point cloud of roughly the right shape.

We next describe in detail the various component parts, including the two neural-networks we use black-box, and then present an overview of our method. We provide experiments showing that the method did improve shape on two datasets, and conclude with a discussion of the limitations of the method, particularly as it relates to generating geometries for scientific computing.

2.1 Single View Reconstruction

Single View Reconstruction (SVR) networks are thus named because they reconstruct a 3D geometry (in this case a signed distance field) from a single image. For each image input, we want an implicit function $\hat{f}(\boldsymbol{x}) : \mathbb{R}^3 \rightarrow \mathbb{R}$, that represents the 3D geometry of the object in the image as the output. To train the network, we gather ground truth example pairs $\{I, \hat{f}\}$ of input images and their matching ground-truth SDF evaluated at grid points. Using this data, we fit parameters θ of Multi-Layer Perceptron (MLP), $f(\boldsymbol{x}; \theta)$, where $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, so that it approximates a signed distance function that matches the input image. To achieve this, we minimize a loss function that computes the distance between the predicted SDF

thesis) and was presented at the ECCV 2022 Workshop on Learning to Generate 3D Shapes and Scenes, co-authored by Thibault Groueix, Matheus Gadelha, and Radomir Mech [70] The motivation and discussion of related works reflects the state of the field at the time, which has since progressed significantly.

values, $\hat{f}$ near the surface and the ground truth SDF values, f.

$$\mathcal{L}_{SDF} = ||f - \hat{f}||_2^2 \tag{2.1}$$

As we want not just any implicit surface function, but a signed distance function in particular, many add an additional loss term known as the *Eikonal regularization term* [33].

$$\mathcal{L}_{eik} = (||\nabla_x f(\boldsymbol{x}; \theta)|| - 1)^2 \tag{2.2}$$

This term encourages the gradients $\nabla_x f$ to be of unit 2-norm, satisfying the Eikonal equation $||\nabla_x f(x)|| = 1$. A solution f to the Eikonal equation, where f vanishes on $\mathcal{X}$, with gradients $\mathcal{N}$, will be a signed distance function [15]. Together these two terms yield the complete loss function:

$$\mathcal{L}_{SVR} = \lambda_1 \mathcal{L}_{SDF} + \lambda_2 \mathcal{L}_{eik} \tag{2.3}$$

where each term is weighted by weights λ_1, λ_2 of the user's choosing. In our method, we use the SVR network to predict a signed distance field of the object of interest at all points on a 3D grid. We then use marching cubes to convert the SDF to a surface mesh, and then sample points on the surface to create a point-cloud, Λ. While learning the SDF in the SVR network leads to a better shape, a point-cloud is easier to manipulate, particularly in combination with the disparity map which is also converted to a 3D point-cloud.

While this training procedure is sound, in practice SVR methods are biased because of where the training data tends to come from. SVR networks need 3D ground truth data which is hard to come by, so many networks rely on the same dataset, ShapeNet [8], a synthetic 3D mesh dataset with tens of thousands of shapes. They then synthetically render the 2D images to match the 3D data. These images are often limited in their realism, so the resulting networks often do not perform very well on real photos with varied lighting conditions and backgrounds. This domain gap between the synthetic training data and the real images of interest is a known limitation of these SVR models. Unlike 3D signed-distance functions, depth data is easier to access in the real world, so monocular depth estimation networks do not tend to suffer from the same biases, which is why we wish to leverage them to improve SVR.

2.2 Monocular Depth Estimation

Depth estimation refers to the problem of inferring depth from RGB images. The word *monocular* is often put in front of it to specify that we are only inferring depth from one single image. In the literature, people often discuss the task of "depth" prediction of a single image, but in most cases what is being predicted is disparity, or the difference between two matching points in two stereo images. In these models, an image, I, is the input, and a 2D disparity map I_{disp} indicating a normalized disparity value for each pixel, is the output.

A simple monocular depth estimation network can be achieved with supervised learning. In this case, we have a set of input images, I_{train}, of size $m \times n$ paired with ground truth disparity maps $D : m \times n \to \mathbb{R}$. We use this data to learn weights of a network $g(x; \theta)$ where $g : \mathbb{R}^2 \to \mathbb{R}$ that predicts a disparity value given a pixel location. We learn those weights by minimizing the loss between the predicted depth and the ground truth.

$$\mathcal{L}_{disp} = ||I_{disp}^{pred} - I_{disp}^{gt}||_2^2 \tag{2.4}$$

2.3 Projecting Disparity to 3D

What we are given after passing the image through the black-box monocular depth estimation network is a normalized disparity map. To use this information to improve the point-cloud obtained from the SVR method, we need to convert it into a 3D point cloud as well. This is a two step process which involves first converting the disparity map to a depth map and then projecting the depth into 3D. We describe our novel approach to this process in detail in this section.

Disparity is used in monocular depth estimation [98, 73, 109] for several reasons. The first being the availability of disparity data and the difficulty of converting disparity to depth. In mixed datasets it is easier to convert depth maps to disparity maps than the other way around, so disparity maps are popular training sets. Second, because of its relationship to inverse depth, using a disparity map for training effectively weights the loss function by depth, biasing towards objects in the foreground. As these networks predict normalized disparity, d, rather than depth, we must find a way to convert I_{disp} to depth values. Disparity is related to depth Z by an affine transformation:

$$\frac{1}{Z} = \frac{d - (c_x^R - c_z^L)}{fb}, \tag{2.5}$$

where b is the camera baseline, f is the focal length, and c_x^R, c_x^L are the horizontal coordinates of the principal points from the stereo camera pair. For simplicity, we rewrite this equation as

$$\frac{1}{Z} = s * d + t, \tag{2.6}$$

where $s = \frac{1}{fb}$ and $t = -\frac{(c_x^R - c_z^L)}{fb}$.

In our method, we take a normalized disparity map and convert it to a partial 3D point-cloud of the visible points of the object, Ψ. Eq. 2.6 is used to convert disparity to depth, but to get the partial point-cloud, we must also convert depth coordinates (u, v) to 3D coordinates (X, Y, Z). To do this, we invert the perspective projection equation in the pinhole camera model:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.7}$$

where (f_x, f_y) is the focal length and (u_0, v_0) is the optical center of the image plane.

Solving for (X, Y, Z) we get

$$u = \frac{f}{Z} X + u_0 \implies X = \frac{(u - u_0)}{f_x} Z \tag{2.8}$$

$$v = \frac{f}{Z} Y + v_0 \implies Y = \frac{(v - v_0)}{f_y} Z \tag{2.9}$$

where f is the focal length defined as $f = \frac{x}{2tan(fov/2)}$ and fov is the field of view and x is the diagonal of the image. We assume (u_0, v_0) to be $(0, 0)$. When we are given only a normalized disparity map, s, t, and fov are all unknown, so we must find acceptable approximations of these parameters in order to get the desired 3D point cloud.

The pinhole camera model assumes the origin of the coordinate system to be the camera position, while most view-centered reconstruction models assume the reconstructed geometry to be placed at the origin. To relate the two coordinate systems, we need to predict an additional parameter, a translation constant z_t. We translate our projection to get $Z' = Z + z_t$ where Z' is aligned with the initial object reconstruction. In total this gives a set four parameters, $\omega = \{s, t, fov, z_t\}$ that we must predict.

Note that if the single-image predictor were object-centric, we would have to predict a rotation to relate a view-centric coordinate-system with an object-centric one. Predicting this rotation is subject to local minima. In cases where a canonical coordinate system is important, we argue that it is better to learn this rotation during training of the single-view predictor rather than optimize it in post-processing, since learning on a data collection might help alleviate these minima from the loss landscape.

The two-step process of using the equations above can be thought of as a function $f(s, t, fov, z_t, d) = \Psi$ which takes a normalized disparity map along with other parameters and transforms it into the corresponding 3D point cloud, Ψ. To approximate the parameters ω, we fix the disparity map, d, and use stochastic gradient descent to minimize the Chamfer distance between the projected point-cloud, Ψ, and the visible portion of the SVR-generated point-cloud, Λ_{vis} via Eq. 2.1. The parameters that minimize Eq. 2.1 are used to create the final point cloud used throughout the rest of the fine-tuning procedure.

It is important to note that this is an under-determined function. There are infinite choices of ω that will result in the same point cloud. There are also many local minima. This makes this approach very sensitive to initialization. To improve robustness, we perform the fitting several times, each time randomly initializing, and then choosing ω that resulted from initialization with best loss as the final parameters.

2.4 Recovering Detail in 3D Shapes Using Disparity Maps

Now that we understand each of the component parts, we can combine them together. We start with an off-the-shelf and pre-trained single-view reconstruction network that produces an unsatisfactory result and aim to use disparity to improve that result to better match the input image. Take the image of a chair shown in Fig. 2.1. The single-view reconstruction network produces a geometry that looks like the body of the chair, but the legs are missing. Our method takes that initial prediction and uses a disparity map to improve the appearance of the geometry.

The fine-tuning procedure is as follows:

1. Input an image, I into a SVR network yielding a 3D SDF of the object.
2. Convert that SDF into a point cloud, Λ, representation of the initial prediction.
3. Split Λ into two sub-pointclouds of points that are visible, Λ_{vis}, and occluded, Λ_{occ}, from the camera viewpoint of the input image.
4. Obtain a 2D disparity map, I_{disp}, of image I using a monocular depth estimation network
5. Given I_{disp} find the focal length, scale, displacement, and translation constants, $\omega = \{fov, s, t, z_t\}$, necessary to convert I_{disp} into Ψ, the 3D point cloud most closely resembling Λ_{vis}.
6. Combine Λ_{occ} and Ψ and remove any points from Λ_{occ} that are now visible. Call this cleaned and combined final point cloud Ψ'.

In step 1, we start by inputing an image, I, into a SVR network to get a 3D SDF of the object. We define an image, I, to be a real-valued matrix of size $m \times n \times 3$ where m and n are the pixel dimensions, and for every pixel value there is defined a size 3, real-valued vector representing the red, blue, and green (RGB) components of that pixel. Next, we obtain point cloud representation from the mesh of the SDF. Note that while we use an SDF as our initial prediction in this method, one could use another method to obtain a point-cloud of the object and start at step 2.

After we have our initial prediction, we split the point cloud into visible and occluded points. The idea here is to leverage the information we have from the disparity map while keeping the points for which we have no disparity information (the occluded points) the same.

In step 4, we use a monocular depth estimation network to predict a normalized disparity map for I. This map is a real-valued m by n matrix with a normalized disparity value for each pixel (u, v). We take the predicted 2D disparity map, I_{disp}, and convert it into a 3D point cloud, Ψ, that will replace the visible part of the object. This is the most technically challenging step and our key contribution that we discuss in section 2.3.

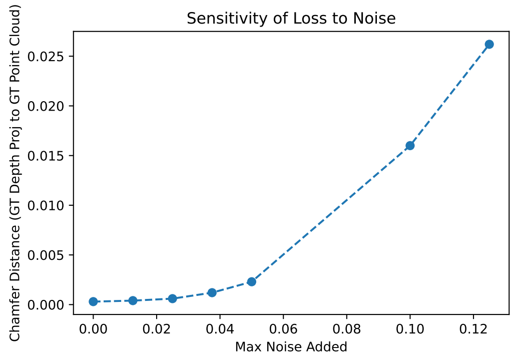
The final step is to combine the occluded point cloud, Λ_{occ} from the initial prediction with the projected disparity points, Ψ. Then a final cleaning is performed, to remove any occluded points that are no longer occluded by the new visible geometry. This is done by iteratively removing points from Λ_{occ} that are not occluded by the projected disparity points, Ψ. This is to account for errors in the initial prediction. Imagine looking at the back of a chair that is mostly open, connected only by spokes. If the initial prediction erroneously reconstructed the chair with a solid back, replacing the visible points with the projected disparity map should restore this detail, but only on the visible side of the back of the chair. The other side will still be solid. We must also remove extraneous points from the original set of occluded points, Λ_{occ}. The final result is the point cloud Ψ'.

For simplicity, we leave the geometry as a point cloud in the experiments, but it can easily be transformed into a mesh [41] or an implicit function [3] as desired because Ψ' is dense. This meshing step will also get rid of any discontinuity between the previous occluded points and the new visible points added from the depth map.

2.5 Experimental Results

2.5.1 Disparity Projection Sensitivity Studies

Before discussing full-scale experiments, we first present two studies to examine the sensitivity of our method to errors in the disparity map and the initialization of parameters. To examine the effect of error in disparity prediction, we simulate error by uniformly adding noise to ground truth disparity maps. Using our disparity projection method, we estimate camera parameters. We then use those camera parameters to project the ground truth disparity and measure the Chamfer distance from the ground truth object. As can be seen in Fig. 2.2, the Chamfer distance stays pretty small for small amounts of noise, but as more noise is added, the Chamfer distance increases, indicating some level of sensitivity to errors in prediction.



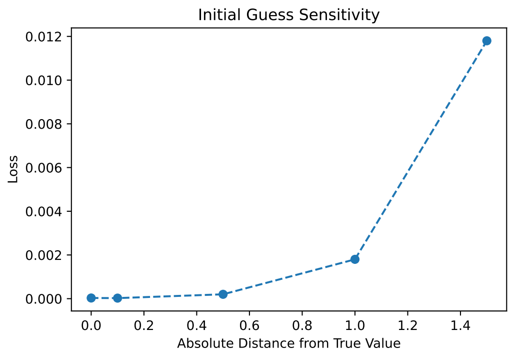


Figure 2.2: Left: our 3D projection is robust to some amount of noise in the input disparity map, but deteriorates if the noise level goes beyond 0.05. **Right:** the error in the projection grows as a function of the distance between the initial guess and the true camera parameters, because of local minima in our optimization problem. We address this issue by randomly choosing multiple initializations.

However, it is important to remember that errors in disparity prediction are often not uniform. The more common failure mode is that errors are concentrated in one area (i.e. the back leg predicted to be closer than it actually is, or fading into the background completely).

The second case of sensitivity we examine is sensitivity to initialization. To measure this we pick a case in which the initial parameters are known and perturb the initial guess steadily further away from the known value. As can be seen in Fig. 2.2, the final loss grows exponentially with distance from the initial value, indicating a significant sensitivity. This finding motivates the random initialization seen in our algorithm. We run the fitting multiple times and pick the final result with the lowest lost.

2.5.2 Full-Method Validation

To validate our method, we perform experiments on two different off-the-shelf reconstruction networks that were trained on two different data sets. The first dataset is a synthetic dataset formed by rendering ShapeNet objects in different poses and lighting conditions. The second is Pix3d, a real-world data set. In both cases we focus on the category of chairs.

We evaluated the final result using two metrics: f-score and Chamfer distance.

Method		f-score	CD
Baseline (3DShapeGen)	Min.	0.0577	0.0239
	Max.	0.5653	0.2629
	Mean	0.2250	0.1086
GT Depth	Min.	0.2639	0.0156
	Max.	0.7824	0.1702
	Mean	0.4938	0.0640
GT Disparity	Min.	0.0331	0.0216
	Max.	0.7060	0.2444
	Mean	0.2827	0.0944

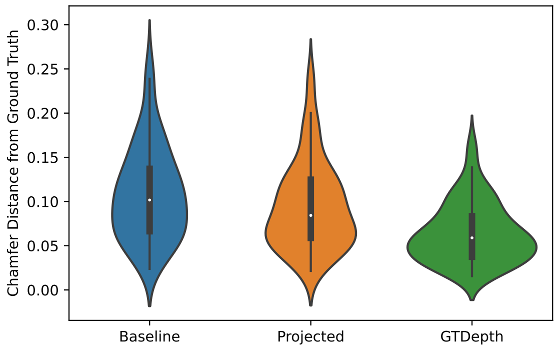


Figure 2.3: We compare a baseline SVR model, 3DShapeGen [95], with two oracle versions of our approach using ground truth depth, and projected ground truth disparities, on synthetic renderings of ShapeNet chairs. Using ground truth depth yield a 180% improvement in the f-score@1 in average. Despite not knowing camera parameters perfectly, our method still provides a 32% improvement in the f-score@1 in average.

Synthetic Images

We present two oracle experiments on synthetic data to clarify the sources of errors in our approach. In this section, we use a dataset of renderings of ShapeNet [8] objects from 3DShapeGen authors [95]. We use their image-only network to perform single-image reconstruction.

First, we try merging the visible points of the ground truth to the 101 initial predictions provided by 3DShapeGen [95]. This achieves the maximum benefit possible with a perfect disparity map and a perfect projection of this map to 3D. On average, each object saw an average improvement of 180% in the f-score at 1 and a 40% improvement in Chamfer distance. This result demonstrates the potential for depth maps to improve the overall appearance and detail of a reconstructed object by an SVR method.

As was discussed in section 2.3, the challenge is that in most cases, monocular depth estimation does not predict absolute depth maps, but instead normalized disparity maps. We use ground truth maps to eliminate any confusion about the source of error. We further test our parameter fitting method on ground truth disparity maps, assuming an oracle monocular depth estimation. We took the same 101 object predictions from 3DShapeGen, applied our method on the ground truth normalized disparity maps and merged the initial prediction and projected disparity maps. We found a 32% improvement in f-score at 1 and a 12% improvement in Chamfer distance. While the magnitude of the improvement is not as large as with the ground truth, there is still a significant improvement. Figure 2.3 compares the performance of the baseline 3DShapeGen prediction, the oracle with ground truth depth and the oracle with ground truth disparity. We show qualitative examples in Figure 2.4.

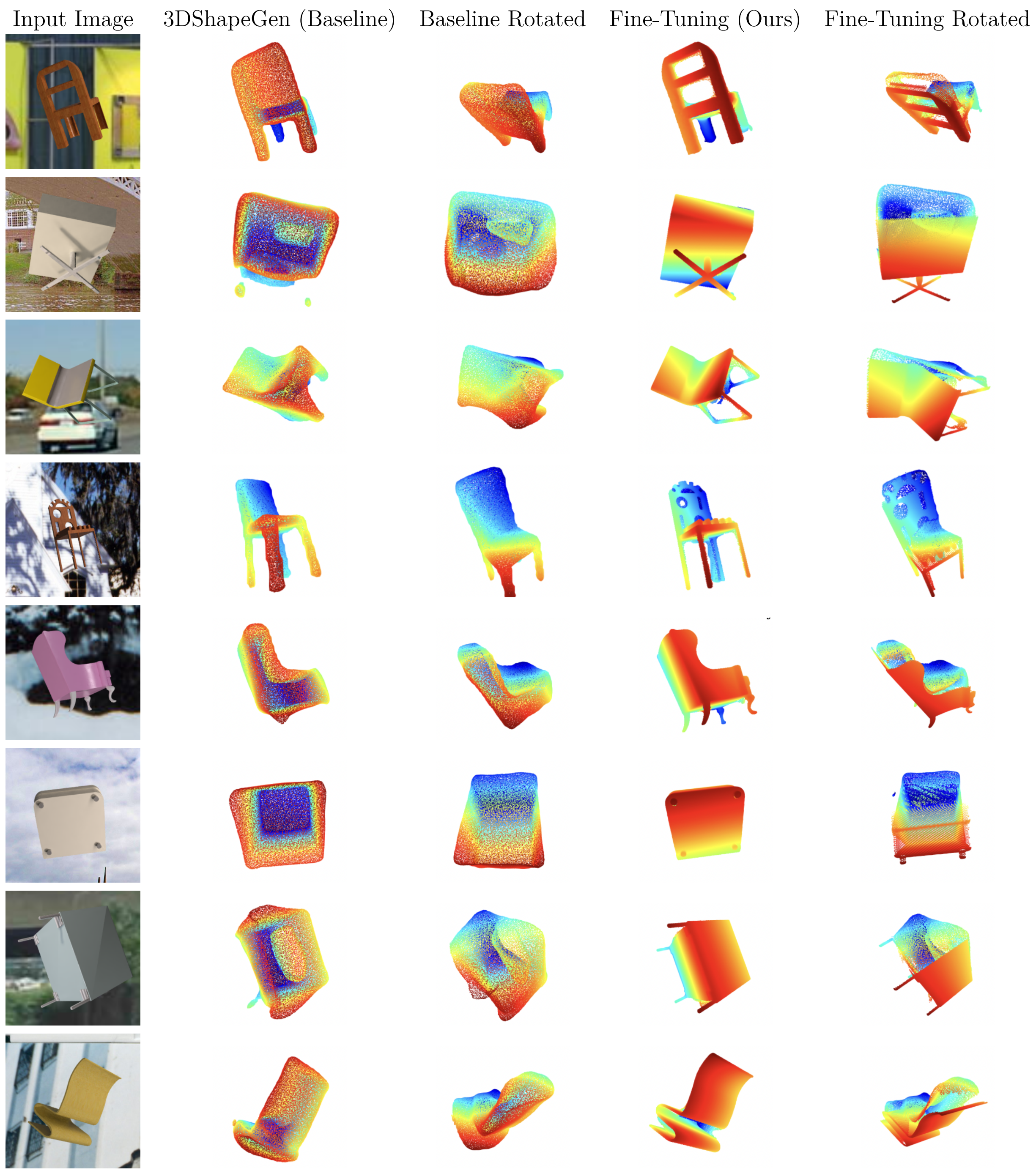


Figure 2.4: Selected qualitative results from study on synthetic images with ground truth normalized disparity. Our method successfully adds much needed detail to the baseline reconstruction. That detail is preserved not just from the view of the image, but upon rotation as well.

Real Images

Method		f-score	CD
Baseline (Mesh R-CNN)	Min.	0.0074	0.0680
	Max.	0.2857	0.7242
	Mean	0.1410	0.1458
Predicted Disparity	Min.	0.0068	0.0642
	Max.	0.3632	0.6536
	Mean	0.1319	0.1476

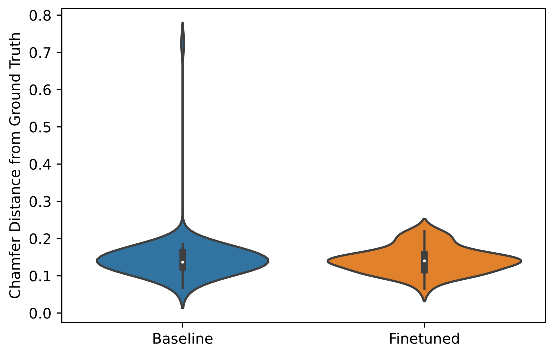


Figure 2.5: We compare the performance of Mesh R-CNN [28] with and without our fine-tuning, on real images from the Pix3d dataset [88]. We use AdelaiDepth [109] to predict disparities. See section **??** for a detailed discussion on performance.

We proceed to test our method on real images. We use 88 images of chairs from Pix3D [88] as our dataset and Mesh R-CNN [28] as our initial method. In this experiment, we use AdelaiDepth [109] to predict disparity, but use ground truth segmentation masks, again to isolate error. In Figure 2.5, we show the distribution of Chamfer distances among the objects in the dataset. After application of our method you can see a slight overall improvement in Chamfer distance.

We found that 44% of objects improved in Chamfer distance from the ground truth with application of our method. In cases that did not see improvement, it was most often from clear errors in the depth estimation. We show qualitative examples in Figure 2.6.

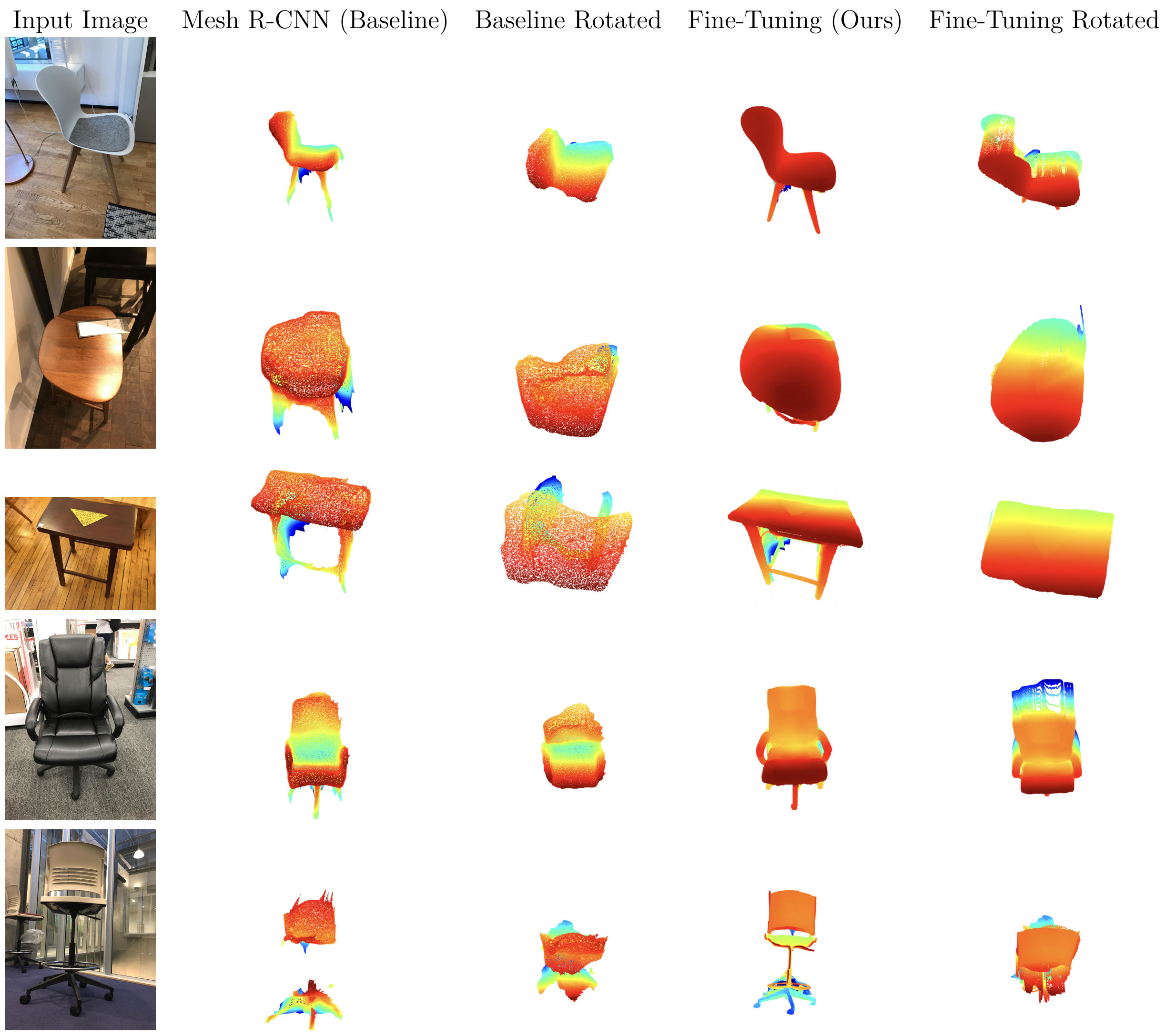


Figure 2.6: Selected qualitative results from the study on real images from Pix3D with estimated normalized disparity.

2.6 Discussion & Limitations

The experiments demonstrate the great potential of depth for fine-tuning 3D geometries and the success of our disparity projection approach. In particular, our method is good for recovering fine detail that SVR models often miss. An example is shown in Fig. 2.4. Take a look at the chairs in the first and fourth rows. The single-view reconstruction model, reconstructs two fairly similar looking chairs with rounded back and no arms, roughly accurate in shape, but missing most of the detail. Applying our fine-tuning method restores the fine detail that is necessary to distinguish the two chairs.

Another great use case is fine detail in the legs. Again looking at Fig. 2.4, rows two, five, six, and seven are all examples of chairs with very thin, short, or otherwise unusual legs that the single view reconstruction model failed to reconstruct. Our model was able to reconstruct these legs, creating a more complete and distinctive object.

2.6.1 Limitations of The Models Used

The objects that fail to see significant improvement in appearance after our fine tuning method tend to fall into two categories: insurmountable errors in the initial prediction and insurmountable errors in depth estimation. Both of these errors are ones that we could expect to improve as base SVR and Monocular Depth Estimation models get better.

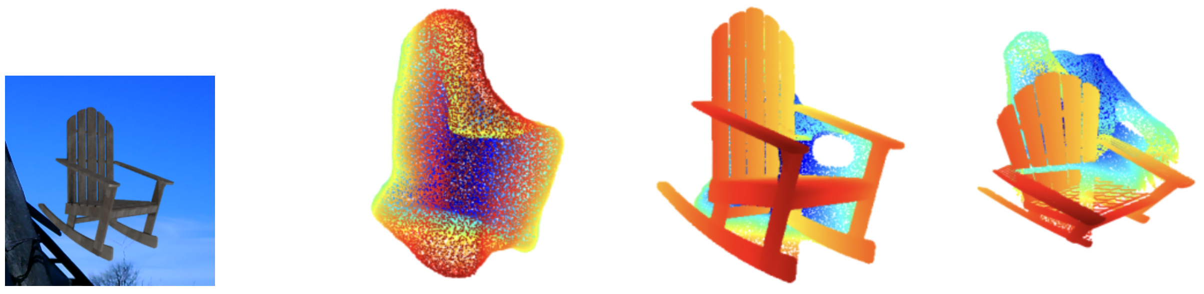

(a) Input Image **(b)** Baseline **(c)** Fine-Tuning (2 views)

Figure 2.7: An example of a flattened projection due to errors in the initial prediction.

Fig. 2.7 shows a typical example of an insurmountable error in the initial prediction, where the network creates a blobby initial prediction. Because of this blobby shape, the visible points of the initial prediction are mostly flat and the estimated disparity is thus projected to a flat point cloud even though the actual visible pixels are not flat.

The second major source of errors lies in the disparity estimation. Our method is fairly robust to minor errors which will occur in any depth estimation method, but there are large

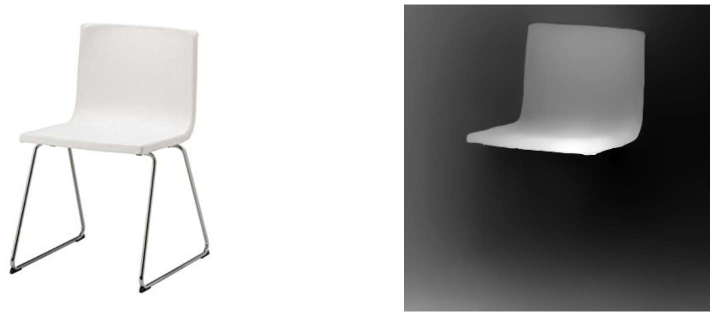

(a) Input Image **(b)** Predicted Disparity

Figure 2.8: An insurmountable mistake in disparity prediction: the legs a treated as background.

fundamental errors that are insurmountable with our approach. An example is the chair in Fig. 2.8 where the monocular depth estimation failed to detect the legs of the chair, so they blend in with the background. Even with perfect segmentation and camera parameter estimation, the legs of the chair would be placed in the background away from the chair. These types of errors account for most of the objects in our dataset that failed to improve, as evidenced by the difference in performance between the ground truth normalized disparity maps and the predicted ones. As monocular depth estimation continues to improve, these errors will become less common.

2.6.2 Limitations of SVR for Scientific Use Cases

Even if SVR and Monocular Depth Estimation methods are greatly improved, there are limitations with how they could be used in a scientific simulation workflow. The main motivation of this method when it was developed was for photo editing. One can imagine a situation where one has a single image that he or she would like to edit. Perhaps he or she would like to rotate an object in the image or change the lighting (and thus the shadows). In order to do this, we must infer a reasonable understanding of that object's shape from one single image. This type of problem does not occur often in scientific situations. Typically, if a scene or object is important enough for us to perform a simulation on, we are able to capture more than one image. Also, a “reasonable" estimation of the rest of the object is not usually good enough for simulation. We typically want a better guarantee that it is a true reconstruction. Additionally, we simply do not have enough 3D ground truth geometries of scenes of interest in scientific simulation to perform a supervised learning approach. If we could reasonably obtain 3D geometries in the first place, there would be less motivation to infer them from images. For these reasons, the rest of this thesis focuses on multi-view

NeuralSDF methods that are supervised on images, not 3D ground-truth geometries.

Despite these objections, there are many concepts addressed in this work that are relevant for scientific work flows (which is why it is included in this thesis):

(1) This class of methods were some of the first to embrace neuralSDFs in learning 3D geometries from images, and was what gave us the initial inspiration to pair neural SDFs with the Embedded Boundary method. The original multi-view neuralSDF method NeuS[100], which will be featured heavily in the coming chapters, was developed concurrently with this method and took inspiration from many of the related works cited. Most notably, it makes use of the Eikonal regularization loss term that was used in many SVR methods.

(2) The main idea of this method is leveraging depth/disparity information to aid in reconstruction from images. While we implement it in a single view reconstruction method, there is no reason that the broader concept must be limited to single view reconstruction. Depth information can be useful in other reconstruction settings as well. In chapter 5, we will discuss on-going work that uses depth information in a tree canopy to help scale a geometry generated from many images of a forest.

(3) While it is true that for scientific applications, we prefer to have images of all angles of the geometry to reconstruct from, rather than taking an educated guess based on the class of objects, that is not always possible. An example will be discussed in further detail in chapter 5, of a forest where the tops of trees can be imaged very well, but the understory (where the trunk connects with the ground) is not visible. While this is not a case of single view reconstruction, there is significant overlap because certain angles of the geometry are simply not visible, so we must look to other information (such as other examples of trees) to complete the geometry.

2.7 Related Work

2.7.1 Monocular Depth Estimation

In their seminal work make3D, Saxena et al. [80] cast depth estimation as a supervised learning problem trained on a dataset of laser scans. Several subsequent papers have improved the architectures [20, 45, 77, 53, 47], the losses [19, 87, 24] and post-processing steps [6, 58]. We organize our discussion around the training data since the robustness and generality of monocular depth estimation approaches stems from the diversity of the training datasets [72]. Laser scanners [82, 27, 37] based on time-of-flight as well as sensors based on structured light [44, 22] provide ground truth depth but sparse annotations for dynamic scenes. Structure-from-motion can also be used to obtain sparse 3D ground-truth, up to scale, from multi-view images of a static scene [50]. Garg et al. [26] propose to use rectified stereo pairs as supervision [29, 30, 55] but the corresponding datasets are not all calibrated, providing disparity up to scale and shift. Chen et al. created a dataset where ordinal relationships between pixels are manually annotated [10]. MidasNet [72] pioneered leveraging those diverse sources of data by estimating normalized disparity and achieving breakthrough results in generalization capabilities of monocular depth estimation. Recent monocular depth estimation models [109, 58] show remarkable performance in various scenarios. Those models can capture the visible part of the scene geometry but have little knowledge about shapes. For example if three legs of a chair are visible, the depth estimation models will not be very helpful for completing the shape and generating the remaining leg.

2.7.2 Single View Reconstruction

Computational approaches to SVR date back to at least Roberts' PhD thesis [76] in 1963. Most related to our work are recent deep learning approaches which currently present state-of-the-art results for this problem. There are two main strategies in SVR to target performance on real images. The first one is through the training data *i.e.* to have real images in the training set. Existing datasets with associated ground truth 3D model have clear limitations. Pix3D [88] does not have good diversity in terms of shapes (only about 700) and mostly contains furniture. ObjectNet [4] and Pascal3D [106] do not have good image-shape alignment. Some approaches aim to learn SVR using image-only datasets using differentiable reprojection losses [97, 48, 40, 31, 96, 52, 108], though this is an extremely challenging task. Another strategy is to use domain adaption techniques to bridge the gap between synthetic renderings and real images, for instance by imposing depth and normal as an intermediate representation between RGB images and the full 3D geometry [103, 105, 114, 85]. These techniques show some improvement on real images but within the scope of the categories spanned by the 3D dataset. They thus do not fully leverage the generality of monocular depth estimation methods. Whereas those approaches perform better than vanilla models trained only on ShapeNet or Pix3D, they still fail to generalize to real scenarios. The choice of 3D data representation is one of the main discriminating factors to analyze deep learning approaches for SVR. Choy et al. [13] propose a volumetric representation using 3D voxel-grids [61, 104], as a natural extension to 2D pixel grids on which to perform 3D convolutions. Several solutions have been explored to mitigate the prohibitive memory consumption of using 3D grids in deep learning. Previously, these solutions were focused on occupancy grids [36, 74, 93], but several more recent radiance fields alternatives using hash tables and tensor decomposition [60, 63, 79, 9] were also proposed. Instead of using grids, three papers [68, 57, 11] concurrently pioneered the use coordinate-based Multi-Layered Perceptrons to model a 3D volume [95]. Several other approaches model a surface instead of a volume. Fan et al. [21] pioneered an approach to generate point clouds on a surface [25, 54]. Other techniques model 3D surfaces via parametric deformations from a reference surface [34, 99, 28]. Despite a lot of progress in representing shapes, all the aforementioned SVR methods suffer when applied to real images.

2.7.3 3D Diffusion and Shape Completion

While not strictly single-view reconstruction, there is another line of research that has exploded since this method was initially developed that is very relevant to this work. Diffusion models are a class of generative AI models that were initially introduced to produce very high quality 2D images. They have since been applied to 3D and are very popular for shape completion. At its core, single view reconstruction is a shape completion problem. We are given one view of an object and wish to reconstruct the occluded portion. Diffusion models have been combined with SDFs [12] and NeRF [17, 35, 107] to complete shapes from single views.

Chapter 3

Simulating Physics with NeuralSDFs

Now that we have seen how SDFs can be learned from images, we wonder, how do those neuralSDFs perform when used in numerical simulation? Because they are learned rather than calculated via traditional methods, we expect them to have different errors and convergence properties. In this chapter we explore how neuralSDFs perform with the Embedded Boundary method. We discuss what types of neuralSDF methods are most suitable for scientific problems and introduce a complete image-to-simulation workflow. We next analyze the various failure modes and convergence properties of these neuralSDFs, connecting how geometric error relates to simulation error. We finally present results of the pipeline end-to-end, demonstrating how we can go from a set of images to a 3D physics simulation while understanding and controlling for error.

3.1 Learning SDFs for Scientific Simulation

In the previous chapter, we discussed methods for learning SDFs from images that involve 3D (or 2.5D) supervision. In these methods, we pair images with their corresponding 3D SDFs, or depth maps, or other 3D geometries and learn a network that best fits the data. This involves gathering thousands of these pairs as training data. Unfortunately, this data is not readily available for many scenes and objects of scientific interest. Imagine you are interested in simulating a wildfire through a particular forest, it is not feasible to gather data of thousands of other forests to train on. If gathering such data were easy, we would simply use that method to obtain our 3D geometry. This motivates the need for some sort of “self-supervision." In other words, in these cases, we would much prefer to use information already in the input images to determine if the network accurately represents the geometry.

In recent years, methods have been developed that take such an approach. Neural Radiance Fields (NeRF) [59] predict the color of an object, given a location and a viewing angle. The parameters of the NeRF network are determined by minimizing the loss between the predicted color, $\hat{C}$, and the true color seen in the input images, C_gt,

$$\mathcal{L} = ||\hat{C} - C_{gt}||_2^2, \tag{3.1}$$

so only the input images themselves are needed in the training process. While NeRF was developed to render novel views of an object given a set of input images, being able to render infinite 2D views gives some sense of 3D structure. NeRF can easily be converted into a 3D mesh using the Marching Cubes algorithm.

While you can get a 3D mesh from NeRF, derivative methods have been developed that improve on the quality of the geometry. One such method, that then spawned derivatives of its own, is Neural Surface reconstruction (NeuS) [100]. NeuS has very similar architecture to NeRF, including the same loss function, but also predicts an SDF value in addition to the pixel color, feeding the two networks into each other, ultimately improving the quality of the geometry.

Recall that in NeRF, the pixel color, C, given a viewing angle is determined by the following function:

$$\hat{C} = \sum_{i=1}^{n} w_i c_i. \tag{3.2}$$

where the weights are $w_i = T_i \alpha_i$, the opacity of the i-th ray segment is $\alpha_i = 1 - exp(-\sigma_i \delta_i)$, the distance between adjacent samples $\delta_i = t_{i+1} - t_i$, and the accumulated transmittance is $T_i = \prod_{j=1}^{i-1}(1 - \alpha_j)$.

NeuS introduces the SDF function, $f(x)$, into the color prediction, by changing the definition of α to

$$\alpha_i = max\left(\frac{\Phi_s(f(\boldsymbol{x}_i)) - \Phi_s(f(\boldsymbol{x}_{i+1}))}{\Phi_s(f(\boldsymbol{x}_i))}, 0\right). \tag{3.3}$$

where Φ_s is called the *s-density* defined as

$$\Phi_s(\boldsymbol{x}) = (1 + e^{-\boldsymbol{x}})^{-1}. \tag{3.4}$$

NeuS has been shown to produce a higher quality 3D mesh than vanilla NeRF, but not much as been studied of the SDF itself. In many graphics applications, a signed-distance function is merely an intermediary to obtaining a high quality 3D surface mesh, but in Embedded Boundary PDE simulation, the SDF provides the geometry representation. In this chapter, we study neuralSDFs generated by the NeuS family of methods (including neus-facto and NeuralAngelo, two methods with identical loss and opacity functions to NeuS that have been modified for faster performance using NVIDIA's new neural graphics primatives, instantNGP[63]) as inputs to the Embedded Boundary method for numerical PDE simulation.

3.2 Methodology for Evaluating NeuralSDFs in EB Simulation

Implicit representations have long played a role in representing complex geometries for numerical partial differential equations (PDEs). Level set methods are useful in tracking complex moving fronts. The Embedded Boundary method[39] uses implicit geometry representations as the basis for solving partial differential equations via finite volumes.

While the work of implicit functions in physical simulation and implicit functions in 3D geometry reconstruction have largely been independent from each other, we see a great opportunity to bring these methods together to solve one of the great challenges in scientific computing: obtaining high quality geometries of complex scenes. Most work thus far in 3D reconstruction has focused on the visual quality of the reconstructed scenes, assuming applications in virtual reality or graphics, but there are also many scientific inquiries that rely on 3D geometries of natural scenes. Forests, glaciers, sea-floors, and city-scale urban environments all have complex geometries that are difficult to model. Performing an accurate 3D simulation of these environments typically requires expensive scanning techniques that often yield noisy results that do not easily pair with existing numerical PDE packages. We see the rise of neural implicit geometries as an opportunity to greatly increase computational scientists' access to high quality geometries, but as of now there hasn't been an in-depth study of these geometries' suitability for scientific simulation.

Like most numerical methods, there are several sources of error in the EB approximation. Assuming the implicit signed distance function that represents the initial geometry is exact, the relevant geometric information could be calculated to arbitrarily high order of accuracy, but in practice there is truncation and floating point error in the moment calculation. Even assuming moments were calculated exactly, the finite volume discretization introduces error depending on the mesh size of the Cartesian grid. These methods are also sensitive to sharp changes in geometry [18]. Previous work has shown that when there is a sharp corner in the geometry, errors in the final simulation can be produced. Slightly smoothing those corners results in a more accurate solution. Counter-intuitively, a less accurate geometry can aid in producing a more accurate final solution. Smoothness of geometry (i.e. differentiability across the surface) is also an important factor in performance for EB simulation. While this fact is implicitly included in most neuralSDF reconstruction methods (most of these methods include an Eikonal regularization term which enforces this property), it is not accounted for in most popular measures of 3D reconstruction quality.

There are also several input attributes that have great effect on output. These include the number of input images, the locations of those images, lighting conditions, geometry conditions (e.g. convexity, occlusions, symmetry etc.) and grid spacing of the predicted SDF. Some of these attributes have fairly obvious effects. If part of and object is occluded or there are no input images of that section, the reconstruction will likely be less accurate in that section. As the method is using a type of interpolation to fill in novel views, if the uncertain region is more similar to the regions that surround it, it is more likely to have a more accurate reconstruction. NeRF has been particularly popular in computer graphics applications where lighting and relighting are of particular interest, so it has been well studied in many lighting conditions. As would be expected, non-lambertian materials such as reflective glass pose some difficulties, but methods have been developed to provide better performance in complex lighting conditions and to allow for relighting of scenes [86]. In the following section, we further explore the effect of these various input attributes, both on geometry quality and on PDE solution quality. Because of the plethora of input configurations and the inherent stochasticity of the network, the experiments that follow are not intended to demonstrate any hard "rule" about neural-implicit methods, but are designed to illustrate the impact of the various factors that go into the creation of these geometries and provide some intuition for how to understand them.

3.2.1 Geometry Quality Metrics

First, a note on how to measure error in 3D reconstruction. There are several approaches to measuring similarity between two 3D geometries. Which metric is best depends on the downstream application of the geometry. Depending on the application there are certain geometric characteristics that might be more important to get right than others. Two common metrics in the computer vision literature are the Chamfer Distance, $\mathcal{CD}$, which measures the similarity of two point clouds. It is defined as,

$$\mathcal{CD} = \sum_{a \in A} |a - b_{nn}| + \sum_{b \in B} |b - a_{nn}|, \tag{3.5}$$

the sum of the distances between all points in point cloud A and their nearest neighbors b_{nn} in point cloud B and the distances of all points in point cloud B with their nearest neighbors a_{nn} in point cloud A. While this measures point cloud similarity, it is also used to evaluate error in SDFs and surface meshes by sampling points on those surfaces to form a point cloud. Another popular metric is IoU or intersection over union:

$$IoU = \frac{A \cap B}{A \cup B}. \tag{3.6}$$

It measures the volume of intersection of two geometries A, B divided by the volume of their union.

While 3D reconstruction quality numbers are reported for most methods, the commonly accepted metrics in the computer vision literature are also commonly accepted to be flawed [94], so they are typically accompanied by significant qualitative results as well. This works well in the computer vision literature because results are most often reported with the main intent of communicating visual quality, not suitability for simulation. When discussing geometry in the context of suitability for Embedded Boundary simulation, new metrics are needed.

In this chapter, we propose our own geometry quality metric that more accurately assesses geometry quality for EB simulation. Importantly it measures accuracy of the SDF directly (not a by-product such as the mesh) and includes a measure of smoothness which is essential for a successful EB simulation.

$$err = \sqrt{mean\left(\left(\frac{f^*_{i,j,k} - f_{i,j,k}}{h}\right)^2\right)} + \lambda K \tag{3.7}$$

where $f^*_{i,j,k}$ is the predicted SDF, $f_{i,j,k}$ is the ground truth SDF, h is the grid spacing, λ is a scaling factor, and K is a measure of the noise on the scale of h

$$K = max(f^*_{i+1,j,k} + f^*_{i-1,j,k} + f^*_{i,j+1,k} + f^*_{i,j-1,k} + f^*_{i,j,k+1} + f^*_{i,j,k-1} - 6f^*_{i,j,k}). \tag{3.8}$$

We illustrate these metrics on a benchmark problem. All methods start with 482 input images of a globe and we compare three different SDF reconstruction methods: (1) Structure-from-Motion generated point cloud, meshed with Poisson meshing, and then ray-casting to determine the SDF (2) Structure-from-Motion generated point cloud with neural network approach NeuralPull[3] to learn the SDF (3) Neural network NeuralAngleo[49] which learns the SDF directly from images. Finally, we also add in an ellipsoid generated manually as a control comparison.

Visually, we can see that NeuralAngelo provided the best reconstruction, but it was not necessarily a fair comparison as NeuralAngelo had the exact camera locations as input. However, the point of this experiment is not to judge the best method, but to find the best *metric*. What gives us more insight into this question are the two methods that relied on a SfM generated point cloud as an intermediary. In both methods, once can see errors in the same region of the sphere, where the Pacific Ocean was represented on the globe. This is because it is a large expanse of a single color, so the method has less distinct visual cues to distinguish between. However the secondary methods, Poisson meshing and NeuralPull, took very different approaches to handling that region. As NeuralPull learns a signed distance function, it is biased towards smootheness, so it smoothed out that region resulting in an indent in the sphere. The Poisson mesh created surface triangles between all of the noisy points in that region and resulting in a bumpy and complicated geometry that from far away is more sphere-like, but up close has significant complexity. Depending on the application, one could imagine scenarios in which one would prefer one versus the other, but in EB simulation the choice is obvious. As EB is an SDF based solver, it also prefers smoothness and attempting to solve a PDE on an SDF of the Poisson Mesh resulted in a forced termination of the code due to failure to converge. This is a significant error that a good error metric must be able to convey.

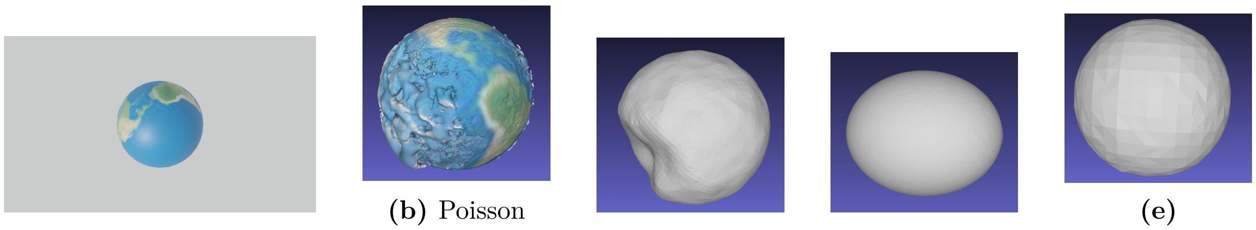

(a) Input Image **(b)** Poisson Mesh **(c)** NeuralPull **(d)** Ellipsoid **(e)** NeuralAngelo

Figure 3.1: Input image with meshses produced by two different toolchains and ellipsoid as control.

In the following table we compare Chamfer Distance, IoU, our SDF error metric, and the error of the PDE simulation. In this case we are solving Poisson's equation and comparing point-wise to the EB computed solution on a perfect sphere and taking the mean.

	CD $\downarrow$	IoU $\uparrow$	SDF Error (Ours)	PDE Error Mean
NeuralAngelo	0.0039	0.83	1.837e-3	5.947e-05
NeuralPull	0.024	0.63	2.138e-3	1.231e-4
Ellipsoid	0.052	0.60	9.066e-3	4.057e-4
Poisson Mesh	0.039	0.14	3.817e-2	NC

Table 3.1: A comparison of our four geometries by two traditional 3D reconstruction metrics (Chamfer Distance and IoU) and our SDF metric. From a simulation perspective, the Poisson Mesh is by far the worst as it causes the PDE solver to fail, but this is not obvious from the Chamfer Distance. Both IoU and the SDF error more accurately capture the PDE performance with ours more directly tied to the quantities of interest in EB simulation.

Chamfer Distance really does a poor job of predicting performance with EB. It ranks the Poisson mesh as fairly similar in quality to the NeuralPull SDF and better than the Ellipsoid when in fact, the Poisson geometry results in no convergence in the PDE solver. Because Chamfer Distance focuses on the locations of points along the surface, this makes sense. IoU gives a better understanding of the whole geometry, so it is intuitive that it would be a better indication of suitability for simulation. It gets the ranking right and shows the Poisson

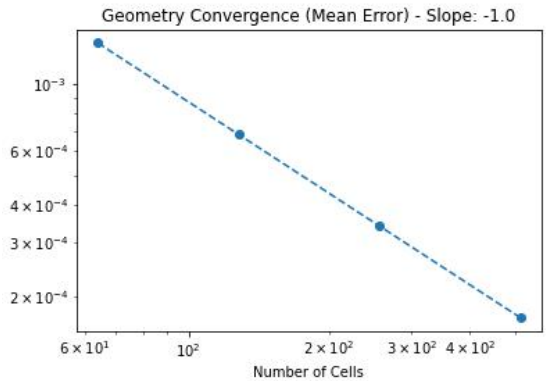
Mesh as far worse than the other three options. The only quibble would be that it ranks the NeuralPull and Ellipsoid geometries as closer in quality than they actually are. There is around a 300% difference in the PDE error, but only a 5% difference in the IoU. In this case, we believe our combination of curvature and SDF error does the best job at predicting what the PDE error will be, clearly showing the Poisson Mesh as an outlier and roughly spacing the other three along with their respective errors. While this method of geometry error analysis worked well in this case, it is important to understand that there is no post-facto analysis of just the geometry alone that will perfectly predict error for any PDE. Simulation error depends heavily on what PDE is being simulated and more practically, what question are scientists trying to answer with that question. If you are trying to understand if a levee in San Francisco is going to break during a flood, it may not matter if the top of the skyscraper next to it is reconstructed perfectly, but if you're doing an earthquake simulation on the same geometry, the top of the building might be very important. We will discuss this problem in more detail in the next chapter which extends this question how to deal with not just error (which is only obtainable when we have some sense of ground truth), but uncertainty?

3.2.2 NeuralSDF Convergence

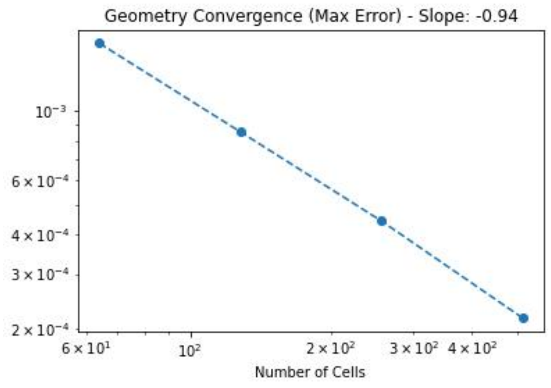
To better understand the performance of neural-SDF methods under various conditions, we vary several parameters an analyze how they affect solution accuracy. For this set of experiments we use a sphere as our test shape as it has a very simple SDF that is easy to compare against, but please note that while these tests are designed to give general intuition around the performance of neural-SDF methods, these results will not necessarily generalize to different geometries. In particular, these methods are known to experience more challenges with non-convex shapes and non-lambertian materials, but based on testing the cube and other shapes, we do expect these results to extend to simple convex geometries with lambertian materials. We are also assuming camera viewpoints and intrinsics are known, as the purpose of this experiment is not to test the camera pose estimation methods that are often paired with neural-SDF methods.

Grid Spacing At evaluation-time Neuralangelo predicts SDF values at discrete grid-points. In this experiment, we refine the grid and measure the SDF error on grid points near the surface. We see linear convergence towards the solution. Similarly when running each of those geometries through an EB simulation of the Poisson Equation, looking at the error we see it approaching linear convergence as we increase the number of cells (although interestingly this behavior is not immediately present in the regime of fewer grid cells.) We see similar behavior on a cube geometry as well.

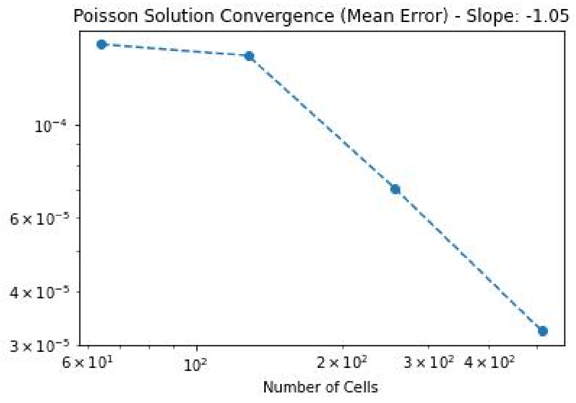
Input Images In this experiment we vary the number of input images, testing Neu-



(a) Average error in signed distance value for grid points near the surface as we increase the number of grid cells in each dimension.

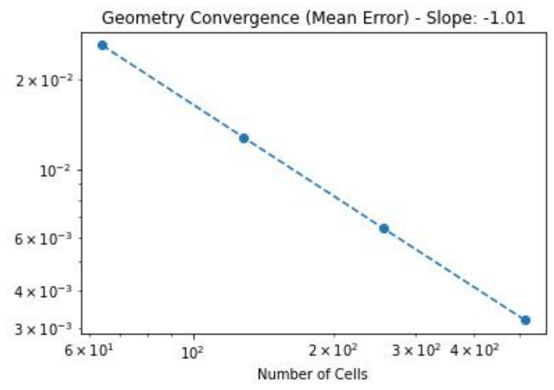


(b) Max error in signed distance value for grid points near the surface as we increase the number of grid cells in each dimension.

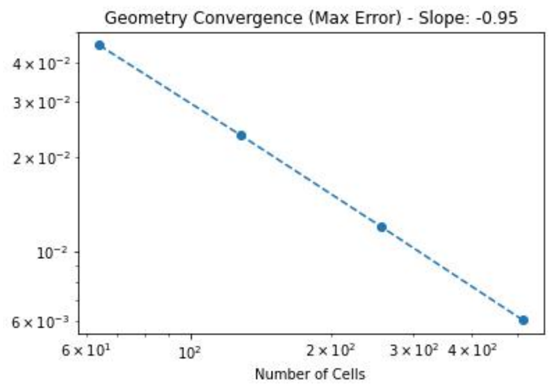


(c) Mean error in the PDE solution between the manually generated sphere and the learned sphere as we increase the number of grid cells in each direction.

ralAngelo's ability to reconstruct novel views. We then measure the SDF error from a manually generated sphere. All images are taken from evenly spaced points on a larger sphere encompassing the sphere of interest. We would expect radically different results if images were clustered in one location. We generally see convergence towards the true solution as we increase the number of input images given to the network, but there is slight deviance from that trend when there are relatively few input images. We take this to be a reminder of the stochastic nature of the network, indicating that reconstructions are not strictly worse as fewer input images are given, but significantly more variable. We also include a visual assessment



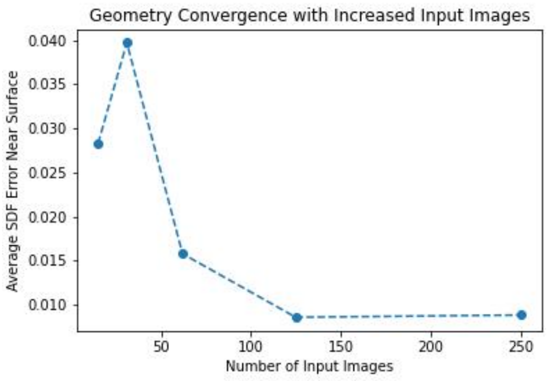
(a) Average error in signed distance value for grid points near the surface as we increase the number of grid cells in each dimension.



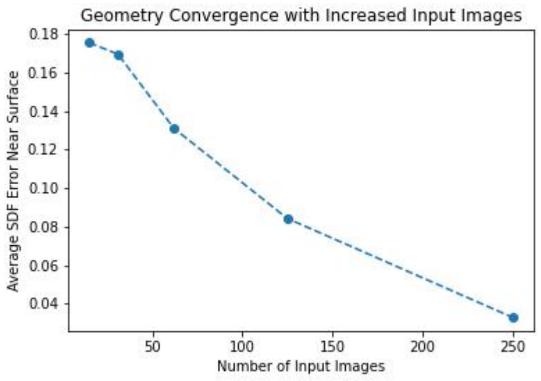
(b) Max error in signed distance value for grid points near the surface as we increase the number of grid cells in each dimension.

Figure 3.3: Convergence results with grid refinement on a cube geometry.

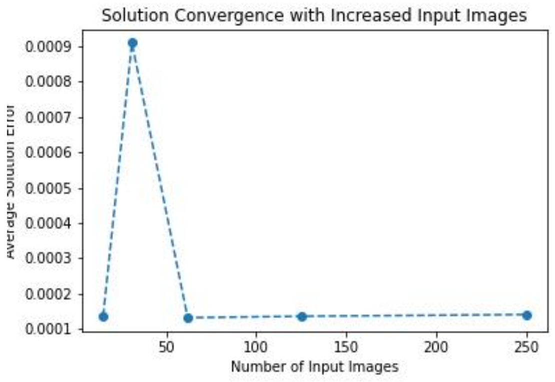
of the geometries in Fig. 3.5 to give the reader intuition of what the SDF error corresponds to visually. After running the image-generated SDFs as well as the manually generated one through a simulation of Poisson's equation via the Embedded Boundary method, we see that the error in geometry roughly translates to solution error. Again, we don't expect it to perfectly line up as solution error is dependent on both the geometry and the equation being performed, but we do see that it gives some general intuition.



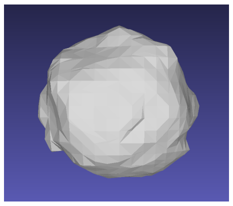
(a) Average error in signed distance value for grid points near the surface with decreasing number of input images.

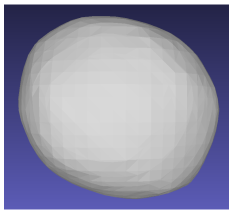


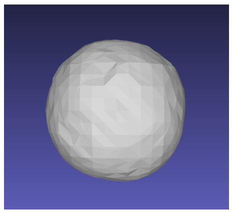
(b) Max error in signed distance value for grid points near the surface with decreasing number of input images.

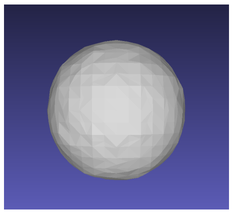


(c) Mean error in the solution to Poisson's equation between the predicted geometries and a manually calculated sphere geometry as we vary the number of input images.


(a) 15 Input Images


(b) 31 Input Images


(c) 62 Input Images


(d) 125 Input Images

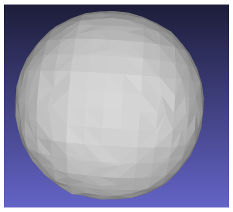

(e) 250 Input Images

Figure 3.5: Sphere geometries produced with NeuralAngelo as we increase the number of input images given to the neural network.

3.3 Image to Simulation on Complex Geometries

In Figs 3.6 and 3.7, we demonstrate this pipeline from beginning to end on an example problem. In this example, we take images of a building, recreate the SDF, then put the geometry into EB-Chombo and simulate Poisson's equation on the geometry. This is a toy problem as the building in the image is quite literally a toy (rather than a real urban scene) and Poisson's equation is a much simplified version of the PDEs that would be of interest in a real urban environment, but this serves as a good demonstration of the potential of this workflow. One could easily see this workflow aiding in the simulation of electrostatics or fluid flow on a real urban environment with images captured via drone.

3.4 Conclusions

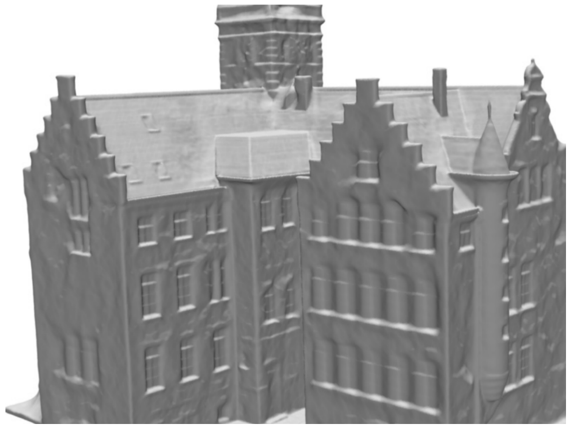
In this section we construct an image to simulation pipeline using NeuS-style methods and an Embedded Boundary PDE solver. We show that we can take 2D images of an object, learn its SDF using NeuS, put that predicted signed distance field into EB-Chombo and end up with a reasonable simulation of a PDE on that geometry.

We then explored some of the variables that can change the quality of that process. We note that the methods we tested rely on COLMAP to estimate the camera viewing angles of the input images, and in our experience SDF prediction seems to be much more sensitive to errors in those estimated viewing angles than color and density prediction models like NeRF. However, when camera angles are known, which is not unlikely in many scientific applications (e.g. if one is collecting images via drone, the drone often saves those camera parameters), the NeuS-style methods we used worked very well. Other variables seem to work fairly intuitively: Geometry converges linearly as the grid of the signed-distance field is refined and it also converges as more input images are added.

We also discussed error measurements in 3D geometry. Much of the computer vision literature uses Chamfer Distance as a measure of error in the geometry, but we show how for Embedded Boundary simulation, this measure can be misleading. Another popular metric, IoU, does a better job, but considering that we are using the SDF values directly in the simulation, we demonstrate that it makes the most sense to look at SDF error directly when discussing suitability for such simulation.

The use of neural geometries in scientific simulation is still in its infancy. This constitutes the first systematic study of these geometries in embedded boundary simulation and there is still much more to understand. While we examined many different types of common errors in this chapter on several test examples, we still do not know how those errors will extend to other geometries more generally. The behavior that we witnessed on a sphere may not extend to an object that is not convex or has different lighting conditions. It is impossible to enumerate every variation in geometry we may encounter and categorize error accordingly. The experiments in this chapter help us build an intuition, but the next level of understanding necessary to use neuralSDFs in PDE simulation is an understanding of error on a per-model basis. In the following chapter, we expand upon this notion of error and uncertainty and present an uncertainty model for communicating uncertainty for any given neuralSDF.

(a) Input Image


(b) High resolution SDF mesh (from NeuS[100] paper)

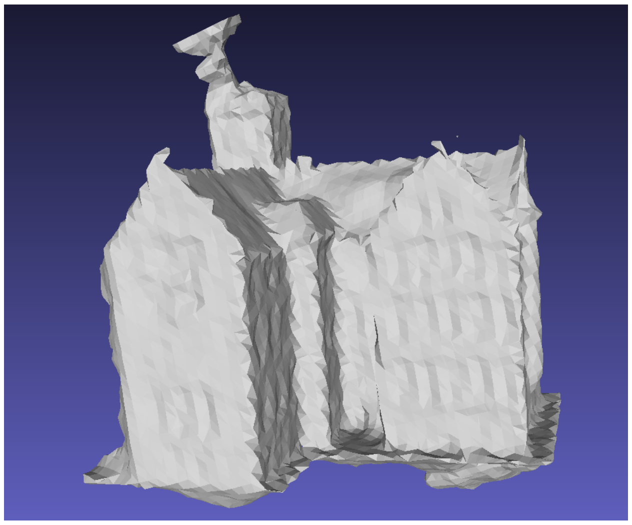

(c) Lower resolution SDF mesh for ease of computation.

Figure 3.6: An input image of a building geometry and two corresponding meshes derived from NeuS-predicted SDFs. We use the bottom geometry (SDF predicted on a 64x64x64 grid) for computational purposes as the corresponding physics problem is small enough to run on a laptop, but we show the top geometry (a higher resolution SDF from the NeuS paper) to show that a higher resolution geometry is easily obtainable.

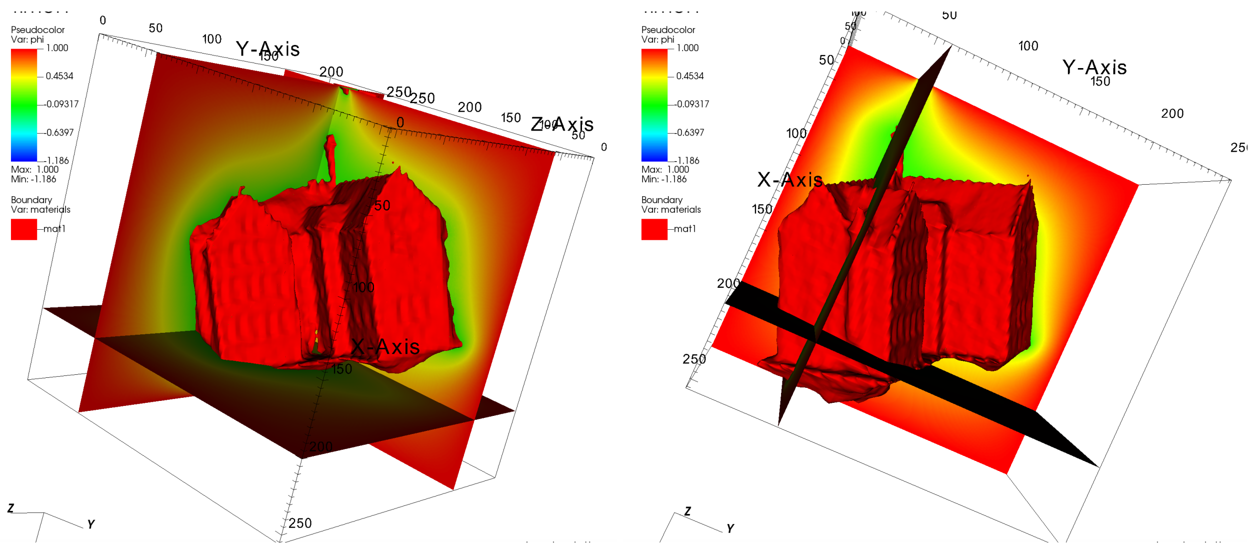


Figure 3.7: A simulation of Poisson's equation with a geometry of a building learned from images. This toy problem serves as a proof of concept for the potential of this pipeline in aiding in electrostatics and fluid flow simulations of urban environments.

Chapter 4

Uncertainty in NeuralSDFs

One of the biggest barriers to adoption of neural implicit functions for scientific modeling is a lack of understanding around model error and confidence. In the previous section, we explored various classes of error in neural implicit functions and how those errors effect numerical simulation. In this section, we focus on modeling uncertainty. Quantifying uncertainty is a vital task in many machine learning systems.

The term “uncertainty" does not have a single agreed upon mathematical definition. Generally speaking, we want uncertainty to communicate something about the confidence of a result and to roughly correlate with error, but defining what mathematical expression properly quantifies that knowledge is a key contribution of any uncertainty model. In this chapter we discuss a specific approach to quantifying epistemic uncertainty of an already trained neural network known as Laplace Approximation. Specifically, we explore a method known as BayesRays [32] which applies Laplace approximation to NeRF. We apply BayesRays to neuralSDF methods and introduce a modification that improves performance in quantifying uncertainty of the predicted neuralSDF.

4.1 Uncertainty in NeRF

BayesRays[32] is the current state of the art method in representing uncertainty in NeRF. We borrow heavily from their notation. Recall from section 1.4.2, that a NeRF is a neural-implicit function that given a position x and a viewing direction d returns a color c and density σ. It is trained on several images of the scene with a loss function that minimizes the difference between the predicted color (integrated along the viewing ray) C and the true color C^{gt} (for a more detailed description please refer to the background section).

$$\mathcal{L}_{nerf} = ||C - C^{gt}||_2^2 \tag{4.1}$$

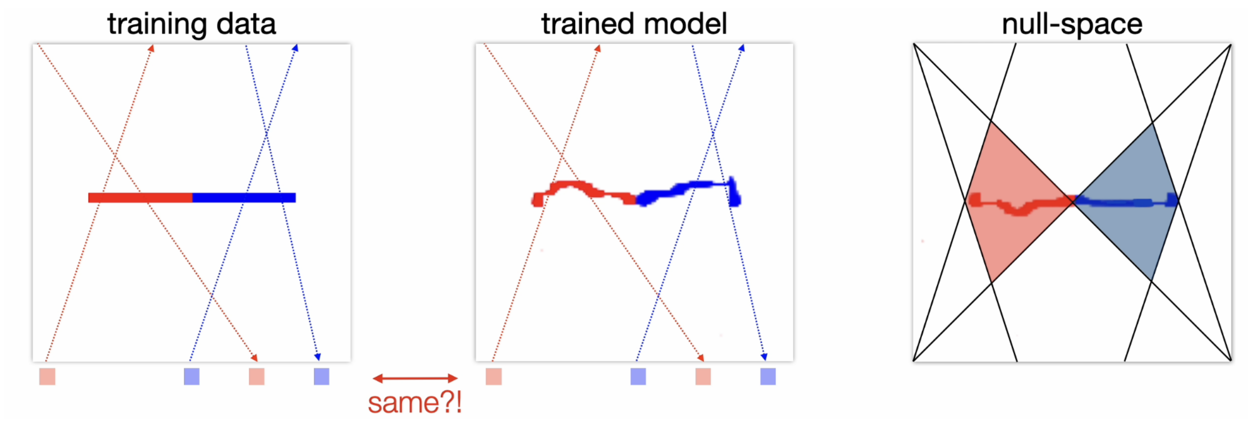


Figure 4.1: An example given by Goli et. al. [32] to illustrate the intuition behind BayesRays.

4.1.1 Intuition

The intuition behind BayesRays comes from an understanding that there are many valid solutions that minimize the loss function of a NeRF. The camera views given as training data do not completely constrain the solution. As can be seen in Fig. 4.1, the example given in the BayesRays paper, there are several deformations of the surface that can result in the same loss. In this figure, we show a toy 2D example of a NeRF where cameras are observing red and blue color values. There are several valid reconstructions of this data that would minimize the loss function. Once can see how the trained model in the picture does in fact minimize the loss function (i.e. the colors viewed from each camera along the viewing angle matches the colors see in the training data), but the surface does not match the surface in the training data, so from other positions and viewing angles the colors may not be correct. In this particular example, the surface could be perturbed quite a bit and still technically minimize the loss function. The regions where the surface can be perturbed without affecting the loss are what we call the solution null-space. BayesRays attempts to quantify uncertainty by quantifying this null-space with the intuition that a NeRF with a large null-space would be relatively uncertain whereas a NeRF with a smaller null-space region would be relatively more certain.

4.1.2 Spatial Deformation

To do this, we introduce a spatial deformation field $\mathcal{D} : \mathbb{R}^D \to \mathbb{R}^D$, which perturbs each point

$$x \mapsto x + \mathcal{D}(x).$$

We choose a parameterization θ in the form of D-dimensional vector displacements stored on the vertices of a grid of length M, allowing us to represent θ as a vector of size $M^D \times D$. We also define a deformation for every individual spatial coordinate via trilinear interpolation:

$$\mathcal{D}_{\theta(x)} = \text{Trilinear}(x, \theta). \tag{4.2}$$

Now we take the already optimized NeRF with optimized parameters ϕ^* and re-parameterize with θ by perturbing each coordinate $\boldsymbol{x}$. This results in perturbed density, color, and color accumulation functions:

$$\tilde{c}_\theta(x) = c(x + \mathcal{D}_\theta(x), d) \tag{4.3}$$

$$\tilde{\sigma}_\theta(x) = \sigma(x + \mathcal{D}_\theta(x)) \tag{4.4}$$

$$\tilde{C}_\theta(x, d) = \sum_{i=1}^{N} w_i(\tilde{\sigma}_\theta)\tilde{c}_{\theta i} \tag{4.5}$$

Recalling from the background section that for a NeRF, the weights are $w_i = T_i\alpha_i$, the opacity of the i-th ray segment is

$$\alpha_i = 1 - exp(-\sigma_i\delta_i), \tag{4.6}$$

the distance between adjacent samples $\delta_i = t_{i+1} - t_i$, and the accumulated transmittance is $T_i = \prod_{j=1}^{i-1}(1 - \alpha_j)$. Recall that any quadratic loss can be interpreted as the negative log-likelihood of a Gaussian with appropriate mean and constant covariance matrix, $\boldsymbol{A}$, i.e.

$$\begin{aligned}\min_x ||x - \overline{x}||_2^2 &= \min_x \left\{(x - \overline{x})^T \boldsymbol{A}^{-1}(x - \overline{x})\right\} \\ &= \min_x \left\{(x - \overline{x})^T \boldsymbol{A}^{-1}(x - \overline{x}) + \log\det \boldsymbol{A}\right\} \\ &= \min_x \left\{-\log \Pr_{x\sim\mathcal{N}(\overline{x},\boldsymbol{A})}(x)\right\}.\end{aligned} \tag{4.7}$$

The NeRF loss is also quadratic, so we can model its uncertainty as $\boldsymbol{A}$, the covariance of the Gaussian. Comparing the NeRF loss, Eq. 4.1, and Eq. 4.7, we see that Eq. 4.1 is equivalent to minimizing the negative log-likelihood assuming the predicted colors are drawn from independent normal distributions around the ground-truth color with variance $1/2$, i.e.

$$\tilde{C}_n \sim \mathcal{N}(C_n^{gt}, 1/2). \tag{4.8}$$

Assuming that we have found optimal parameters ϕ^* in training, meaning we are in a local minimum, we can expect that a deformation will not significantly decrease the loss. To encourage this, we introduce a quadratic additive loss on the deformations, which we interpret as a regularizing independent Gaussian prior $\theta \sim \mathcal{N}(0, \lambda^{-1})$. The resulting posterior, $p(\theta \mid I)$, has a negative log-likelihood, $h(\theta)$, which is given by

$$h(\theta) = \mathbb{E}_n \mathbb{E}_{r \sim I_n} ||\tilde{C}_\theta(r) - C_n^{gt}(r)||_2^2 + \lambda ||\theta||^2. \tag{4.9}$$

This equation is the negative log-likelihood of the posterior distribution of the spatial NeRF parameters conditioned on the training data, $\mathcal{I}$, up to a constant, which can give us some quantification of the epistemic uncertainty of the neural network.

4.1.3 Laplace's Approximation

In order to understand the uncertainty induced by Eq. 4.9 on the distortion field, θ, we leverage Laplace's approximation. Laplace's approximation [75, 16] provides an analytical expression for a posterior probability distribution by fitting it with a Gaussian distribution. The mean of this Gaussian is equal to the maximum a posteriori probability (MAP) estimate (i.e. the mode of the posterior distribution) and its precision is equal to the the negative Hessian of the log-likelihood (i.e. the observed Fisher information)[1]. Specifically, take a model with dataset $\{x_n, y_n\}_{n=1,\ldots,N}$ comprising inputs x, outputs y, with parameters θ. The likelihood is denoted $p(y \mid x, \theta)$, the parameter prior is $p(\theta)$ and the joint density of the outputs and parameters

$$p(y, \theta|x) = p(y|x, \theta)p(\theta|x) = p(y|x)p(\theta|y, x). \tag{4.10}$$

The joint density is equal to the product of the likelihood and the prior, which by Bayes' rule is equal to the product of the marginal likelihood and the posterior. Laplace's approximation approximates the joint by an un-normalized Gaussian,

$$\tilde{q}(\theta) = Zq(\theta), \tag{4.11}$$

where q is the approximate density, $\tilde{q}$ is the un-normalized density, and Z is a constant (independent of θ). The probability density function of a Gaussian distribution $q(\theta)$ is given by

$$q(\theta) \overset{\text{def}}{=} \exp\left(-\frac{1}{2}(\theta - \hat{\theta})^T \Sigma^{-1} (\theta - \hat{\theta})\right) \tag{4.12}$$

where $\hat{\theta}$ is the mean and Σ is the covariance matrix. The marginal likelihood, $p(y \mid x)$ is independent of θ and the posterior $p(\theta \mid y, x)$ normalizes over θ, we can identify them with Z and $q(\theta)$ yielding:

[1]This can also be formalized by taking a second-order Taylor expansion of the negative log-likelihood

$$p(y,\theta|x) \simeq \tilde{q}(\theta) = p(y,\hat{\theta}|x)\exp\left(-\frac{1}{2}(\theta-\hat{\theta})^T\Sigma^{-1}(\theta-\hat{\theta})\right), \tag{4.13}$$

where

$$\hat{\theta} = \text{argmax}_\theta \log p(y,\theta) \tag{4.14}$$

$$\Sigma^{-1} = \boldsymbol{H}_\theta(-\log p(y,\theta|x)|_{\theta=\hat{\theta}}) \tag{4.15}$$

Laplace's Approximation on the NeRF Deformation Field

To apply Laplace's approximation on our reparameterized NeRF, note that in Eq. 4.9 the minimum is obtained when $\theta = 0$ as that implies $\tilde{\sigma}_0(x) = \sigma_{\phi^*}(x)$, $\tilde{c}_0(x) = c_{\phi^*}(x,d)$, and $\tilde{C}_0(r) = C_{\phi^*}(r)$. Therefore, 0 is the mode of the distribution $p(\theta|I)$ and we are able to apply the laplace approximation around $\theta^* = 0$. This results in a belief distribution (i.e., uncertainty):

$$\theta \sim \mathcal{N}(0, \boldsymbol{H}_*^{-1}), \tag{4.16}$$

where $\boldsymbol{H}_*$ is the Hessian matrix $h(\theta)$ (Eq. 4.9 evaluated at 0).

4.1.4 Computing the Hessian

The introduction of a deformation field allows us to make certain assumptions that enable us to efficiently compute the Hessian. Specifically we assume:

1. We do not have multiple color observations for a single ray.
2. The deformations' influence is limited spatially, e.g. a small deformation does not obscure a large field of view.

We will see that assumption 1 enables first conditioning expectation on the ray, and then the color given the ray. Assumption 2 will justify a sparsity assumption $\boldsymbol{H}_*$.

To start, the Hessian of the log-likelihood with respect to the parameters of a statistical distribution is known as the Fisher information.

$$\mathcal{I}(\theta) \overset{\text{def}}{=} -\mathbb{E}_{X\sim p_\theta}\left[\frac{\partial^2 h(X;\theta)}{\partial\theta^2} \mid \theta\right] = -\boldsymbol{H}(\theta). \tag{4.17}$$

Assuming regularity, we can also define this as the variance of the parametric score [46]

$$\mathcal{I}(\theta) = -\mathbb{E}_{x\sim p_\theta}\left[\frac{\partial h(X;\theta)}{\partial\theta}^T \frac{\partial h(X;\theta)}{\partial\theta} \mid \theta\right]. \tag{4.18}$$

Now, take the pair of random variables corresponding to a ray and its predicted color, (r, y), where $r \sim \{I_n\}$ and $y = C_n^{gt}(r)$. Then the equation takes the form

$$\mathcal{I}(\theta) = -\mathbb{E}_{(r,y)}\left[4\epsilon_\theta(r)\boldsymbol{J}_\theta(r)^T\boldsymbol{J}_\theta(r)\right] - 2\lambda\boldsymbol{I} \tag{4.19}$$

where the residual error of the ray is

$$\epsilon_\theta(r) = ||\tilde{C}_\theta(r) - C_n^{gt}(r)||^2, \tag{4.20}$$

and the Jacobian of first derivatives is

$$\boldsymbol{J}_\theta(r) = \frac{\partial\tilde{C}_\theta(r)}{\partial\theta} \tag{4.21}$$

which can be computed via back-propagation. Using assumption 1 and the definition of conditional expectation, we can simplify further:

$$\mathcal{I}(\theta) = -\mathbb{E}_r\left[4\mathbb{E}_{y|r}[\epsilon_\theta(r)\boldsymbol{J}_\theta(r)^T\boldsymbol{J}_\theta(r)]\right] - 2\lambda\boldsymbol{I}. \tag{4.22}$$

By definition $\mathbb{E}_{y|r}[\epsilon_\theta(r)] = \text{var}(\mathcal{N}(C_n^{gt}, \frac{1}{2})) = \frac{1}{2}$, so

$$\mathcal{I}(\theta) = -\mathbb{E}_r\left[2\boldsymbol{J}_\theta(r)^T\boldsymbol{J}_\theta(r)\right] - 2\lambda\boldsymbol{I}. \tag{4.23}$$

Applying Eq. 4.17 and approximating the expectation via sampling R rays, yields the following approximation for $\boldsymbol{H}$:

$$\boldsymbol{H}(\theta) \approx \frac{2}{R}\sum_r \boldsymbol{J}_\theta(r)^T\boldsymbol{J}_\theta(r) + 2\lambda\boldsymbol{I}. \tag{4.24}$$

Now, we take advantage of the reparameterization to simplify this even further in estimating Σ. By assumption 2, since each vector entry in θ corresponds to a vertex on the grid, its effect is spatially limited to the cells containing it. This results in most deformation parameters having little correlation which implies $H(\theta)$ is sparse and dominated by its diagonal. Using this assumption, we can effectively approximate Σ by only computing diagonal entries of $\boldsymbol{H}$.

$$\Sigma \approx \text{diag}\left(\frac{2}{R}\sum_r \boldsymbol{J}_\theta(r)^T\boldsymbol{J}_\theta(r) + 2\lambda\boldsymbol{I}\right)^{-1}. \tag{4.25}$$

Measuring Spatial Uncertainty

We've transformed the problem of measuring uncertainty to one of computing Jacobians. Specifically, the covariances of our deformation field, Σ, encode the spatial uncertainty of the radiance field, i.e. large variance in the deformation parameters signals a less constrained surface reconstruction. The diagonal entries of Σ define a marginal variance vector $\boldsymbol{\tau} = (\tau_x, \tau_y, \tau_z)$. At each grid vertex τ defines a spatial ellipsoid which can be deformed with minimal change in the loss. The norm of this vector $\tau = ||\boldsymbol{\tau}||_2$ is a positive scalar that measure the local spatial uncertainty of the radiance field at each grid vertex, and form a spatial uncertainty field,

$$\mathcal{U}(x) = \text{Trilinear}(x, \tau). \tag{4.26}$$

This uncertainty field can be computed for any NeRF architecture, including sdf-derivatives NeuS and NeuralAngelo as we will show in the following sections, but it continues to represent the uncertainty of the radiance field, not the signed-distance function, which while closely related in NeuS-like methods, are not the same thing.

4.2 BayesRays with NeuralSDFs

For our workflow, we wish to adapt BayesRays for NeuS-style methods. From a mathematical standpoint, NeuS is constructed very similarly to NeRF, so there is a pretty straight-forward way to use NeuS with BayesRays in an almost black-box manner. However, while this is very simple from engineering perspective, it is not in line with our downstream applications and what we truly wish to gain from an uncertainty model of NeuS. In numerical modeling, we are interested in using the predicted SDF directly with our numerical model and want some understanding of error/uncertainty in the SDF geometry. We do not have any direct use for rendering or color information, and in some cases the quality of the rendering can be dramatically different than the quality of the geometry (see Fig. 4.2). Accordingly, we wish to make sure the uncertainty model directly captures the uncertainty in the 3D geometry and does not include any extraneous information. For this reason, we propose modifications to the naive approach and demonstrate that these modifications better reflect true geometric error through preliminary experiments.

Because BayesRays is designed to work black-box with any NeRF method, naively, you can swap Eq. 4.6 for the NeuS opacity equation (please refer to Sec. 1.4.2 for more background on NeuS),

$$\alpha_i = max\left(\frac{\Phi_s(f(\boldsymbol{x}_i)) - \Phi_s(f(\boldsymbol{x}_{i+1}))}{\Phi_s(f(\boldsymbol{x}_i))}, 0\right), \tag{4.27}$$

and perturb the s-density function similarly to how the density function in Eq. 4.4:

$$\tilde{\Phi}_s = \Phi_s(f(x) + \mathcal{D}_\theta(x)). \tag{4.28}$$

However, while this perturbs the spatial component of the opacity, it does not perturb the SDF itself, which is the main quantity of interest. Instead we suggest perturbing the SDF more directly:

$$\hat{\Phi}_s = \Phi_s(f(x + \mathcal{D}_\theta(x))). \tag{4.29}$$

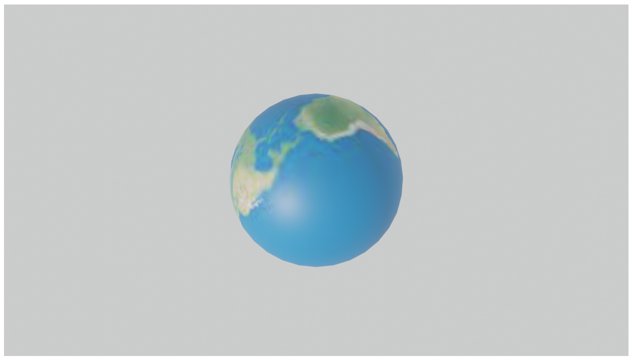
If what we are truly interested in is the SDF which is already being computed on a grid, and are not at all interested in color uncertainty, there is no need to introduce a spatial perturbation of the individual color predictions, so we can modify Eq. 4.5 to instead be:

$$\hat{C}_\theta(x, d) = \sum_{i=1}^{N} w_i(\hat{\Phi}_s)c_i. \tag{4.30}$$

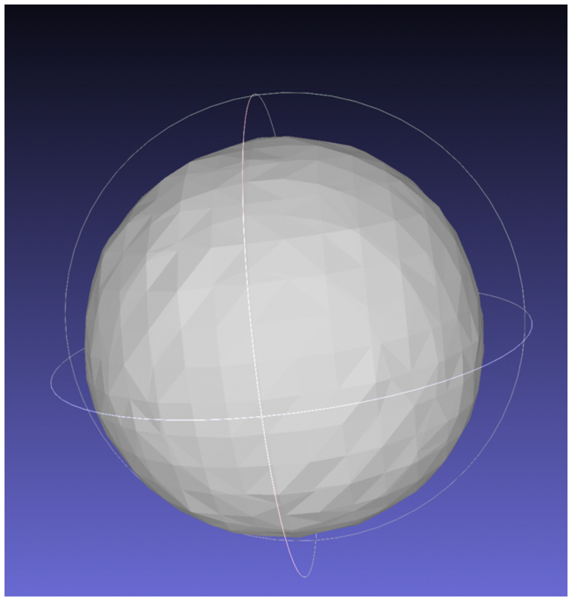
To compute this modified uncertainty, the approximation in Eq. 4.25 still holds, but the Jacobian, $\boldsymbol{J}_\theta$, is now taken with respect to our proposed color value perturbation, $\hat{C}_\theta$, as opposed to the original BayesRays quantity $\tilde{C}_\theta$. The computed derivatives will be imbued with information of the uncertainty in the SDF network f via the chain-rule, but the color network c is no longer parameterized by θ, so it is independent. In doing this, we do lose information about uncertainty in the rendering, but for applications where the desired output is the SDF, such as numerical simulation, only uncertainty in the SDF is of interest.

4.2.1 Rendering Uncertainty

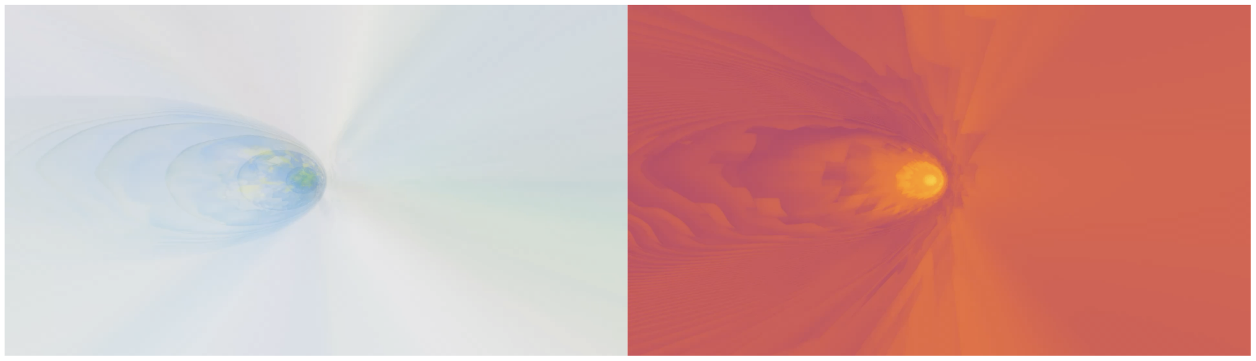
Rendering uncertainty is the main method of evaluation shown in the BayesRays paper, and while it makes sense for understanding uncertainty in a NeRF, it does not necessarily translate to neuralSDFs. This is particularly true when the rendering of NeuS and the geometry produced do not line up as can be seen in Fig. 4.2. The converged NeRF (shown in the left side of part c) is not a very accurate representation of the input images. BayesRays using deformation $\tilde{\Phi}_s$ captures the uncertainty in the rendering. It shows high uncertainty in the space between the camera and the object, where there are many floating pixels and the world patter appears to be smeared across the view. It shows higher certainty in the center where the sphere is actually located. By only looking at the image rendering and the uncertainty rendering together, we see what could be interpreted as a good depiction of uncertainty, but looking at the meshed SDF (part b), gives a very different story. NeuS has two outputs that work in conjunction with each other: (1) a color and density prediction network similar to NeRF that can be used to render novel views and (2) a prediction of SDF values along a grid. When using NeuS as a geometry generation method for numerical simulation, we do not care at all about the rendering, we only use the predicted SDF. In this particular case, the SDF appears to be very accurate even though the rendering has obvious flaws. A scientist ought to feel very comfortable using this geometry in a simulation, but looking at a BayesRays rendering does not communicate that.



(a) Input image



(b) Meshed output SDF

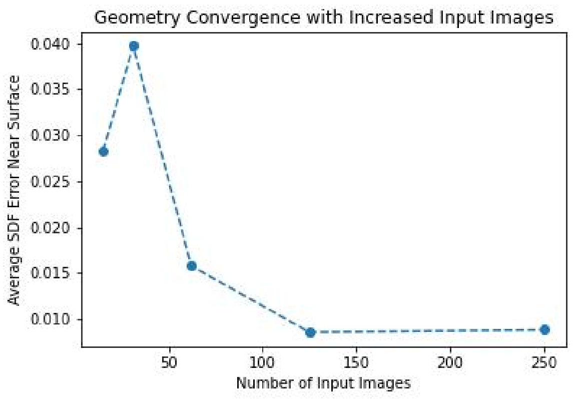


(c) Output rendering with accompanying uncertainty rendering

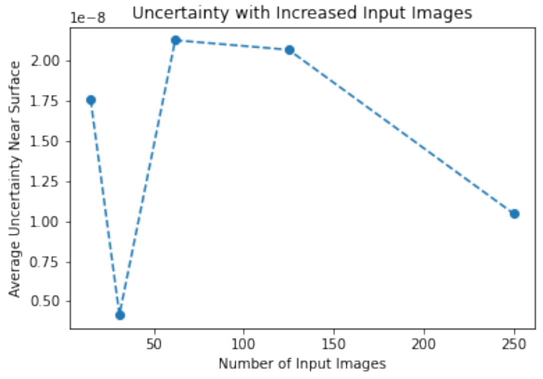
Figure 4.2: A comparison of the inputs of NeuS (image a) with two outputs of the converged network (the rendering of the object, image c, and the mesh of the predicted SDF, image b). Although the rendered novel view is of very poor quality, the mesh is of relatively high quality. Therefore it is impossible to have one uncertainty quantity that accurately describes both rendering and geometric error. BayesRays using deformation, $\tilde{\Phi}_\theta$, accurately models uncertainty in the rendering (as can be seen in the left part of image c), it does not give much useful information about the uncertainty of the geometry shown in image b, which is produced by the same network.

4.2.2 Uncertainty and Geometric Error

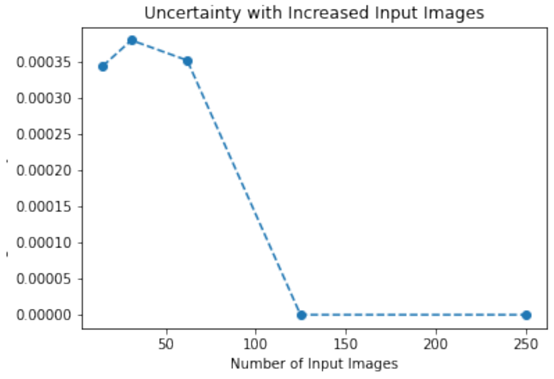
In Fig 4.2 we demonstrate that volumetric rendering of uncertainty obscures information about the surface of the SDF and does not make sense for SDF based applications, so how should we use uncertainty in SDF-based applications? Intuitively, in these applications we want uncertainty to serve as a proxy for geometric error. We also expect that as we increase the number of input images given to NeuS, we would see less error in the predicted SDF correspondingly, less uncertainty. This intuition is roughly demonstrated in the convergence study we did in Ch. 3. In Fig. 4.3.a, we see this general trend of error decreasing as number of input images increases. The exception is with the first data point of 12 input images, which we can attribute to networks sometimes getting lucky. Now given this trend of error, we look at which deformation field yield an uncertainty estimate that more closely tracks with error. Clearly $\hat{\Phi}_s$ outperforms $\tilde{\Phi}_s$ in this case. The "black box" application of BayesRays to NeuS which perturbs the density and color fails to give meaningful information about the geometric error of the object.



(a) Error in the SDF as we increase the number of input images.



(b) Uncertainty at SDF grid points near the surface using deformation $\tilde{\Phi}_s$



(c) Uncertainty at SDF grid points near the surface using deformation $\hat{\Phi}_s$

Figure 4.3: We can see that measuring uncertainty with deformation field $\hat{\Phi}_s$ tracks with observed error much better than using $\tilde{\Phi}_s$

4.3 Future Work: Propagating Uncertainty in Embedded Boundary Simulation

In the previous sections, we presented a method for estimating uncertainty of NeuS-style neuralSDFs. While on their own, this method can advise scientists of the suitability of a geometry before using it to perform a numerical simulation, what is of more use is an uncertainty estimate of not just the geometry but the entire physical simulation.

Given the nature of 3D neural reconstruction, there will always be some level of geometric uncertainty, but one can imagine situations where that uncertainty would greatly affect the overall simulation quality and ones where the effect would be negligible. Imagine performing a flood simulation on a reconstructed geometry of a city. The simulation could be very sensitive to even small geometric errors in certain regions such as dams or levees, whereas other regions such as the top of tall buildings could be completely missing with no effect at all. Geometric uncertainty can only tell us so much without a connection to the downstream PDE simulation. In future work, we plan to connect uncertainty estimates in 3D neural signed distance functions to the Embedded Boundary method to quantify the effect of geometric uncertainty on numerical PDE computations.

In addition to providing better uncertainty quantification in numerical simulation, connecting the two methods has the potential to lead to further research by then using the results to go back and improve the geometry generation methods. Literature has shown that physics and numerical methods can be used to effectively guide the output of machine learning methods, and we believe this could be true of geometry generation methods as well.

Chapter 5

Applications to Forestry

In the previous chapters, we explored various computer vision technologies from the lens of scientific computing. We used simple toy problems to more clearly analyze the benefits and potential errors in such technologies, but the value of these technologies lies in their ability to handle real scientific data and answer impactful scientific questions. In this chapter, we demonstrate how the various technologies discussed perform on real data from the Open Forest Observatory and their potential real-world impacts.

The Open Forest Observatory (OFO) is an open source software and data repository created in a joint effort by UC Davis, University of Arizona, and CU Boulder to facilitate drone-based forest mapping for forest ecology and management applications. The dataset in development includes drone-derived 2D photos and videos of forests, photogrammetry-derived 2D and 3D forest maps [110], and field-based tree geolocation and species classification information, among other forest data. The objective of the OFO is to make current state-of-the art drone-based mapping methods and data accessible to forest ecologists and managers with limited technical background in imagery processing and remote sensing. By enabling rapid collection of forest inventory data over broad extents, the OFO aims to increase the pace and scale of forest research and management that has traditionally been dependent on time- and cost-intensive, spatially-constrained ground-based survey work. In the following sections we discuss work we have done[1] in incorporating contemporary 3D reconstruction techniques into the OFO, highlighting the impact of these models and also describing future research directions.

[1]Some of the work presented in this chapter will also appear as workshop paper for the 2024 ICLR Workshop on Climate AI, co-authored with Arjun Rewari, Trevor Darrell, and Derek J. N. Young [71].

5.1 Forest Mapping

Forest inventories are critical resources for understanding ecological processes and informing forest management, but they have traditionally required time-consuming ground-based surveys. Individual tree-level measurements including location, height, stem diameter, health status, and species identity require substantial time and effort to obtain. For example, a standard approach to forest mapping is to establish a plot centerpoint and measure each tree's position relative to it in radial coordinates using a laser rangefinder (distance) and compass (azimuth). The spatial extent of the survey is thus limited to the mapper's visual line of sight from the plot centerpoint; for larger maps, multiple centerpoints must be established and additional steps must be taken to keep the multiple subplots aligned, contiguous, and non-duplicative. Individual tree-level measurements including height, stem diameter, health status, and species identity require additional time and effort to obtain. Advances in drone and imagery processing technology are enabling a new era of forest research in which individual trees can be mapped, measured, and identified to genus or species across broad areas without extensive ground surveys [62, 7], reducing survey time to days rather than months. Drone-based mapping approaches generally involve executing a drone mission to collect a series of images with high (70-90%) image-to-image overlap over a contiguous study area. The images are then traditionally processed using photogrammetry into downstream data products such as an orthomosaic, canopy height model, and a 3D mesh model of the scene. In turn, one or more of these products can be used extract forest inventory data; for example, individual treetops may be detected as local maxima in the canopy height model [110] and trees may be classified to species based on their appearance in the orthomosaic [23, 102]. Forest areas many hectares in extent can be surveyed in a day of imagery collection followed by automated processing; traditional ground-based surveys of the same area may take months. Although drone-based inventories do not capture forest conditions to the same level of detail as ground-based methods, particularly in the forest understory, the tools are continually improving.

5.1.1 Structure-from-Motion (SfM)

Currently, the 3D forest models in the OFO are created using structure-from-motion (SfM) [81], a classical photogrammetry technique. As we detail in Chapter 1, SfM consists of two main processes, correspondence search and incremental reconstruction. Correspondence search takes in a set of images, processes them through feature extraction, matching, geometric verification, and outputs a scene graph which shows relationships between images. This scene graph is then used in the incremental reconstruction which consists of initialization, image registration, triangulation of scene points, and outlier filtration. Since the resulting

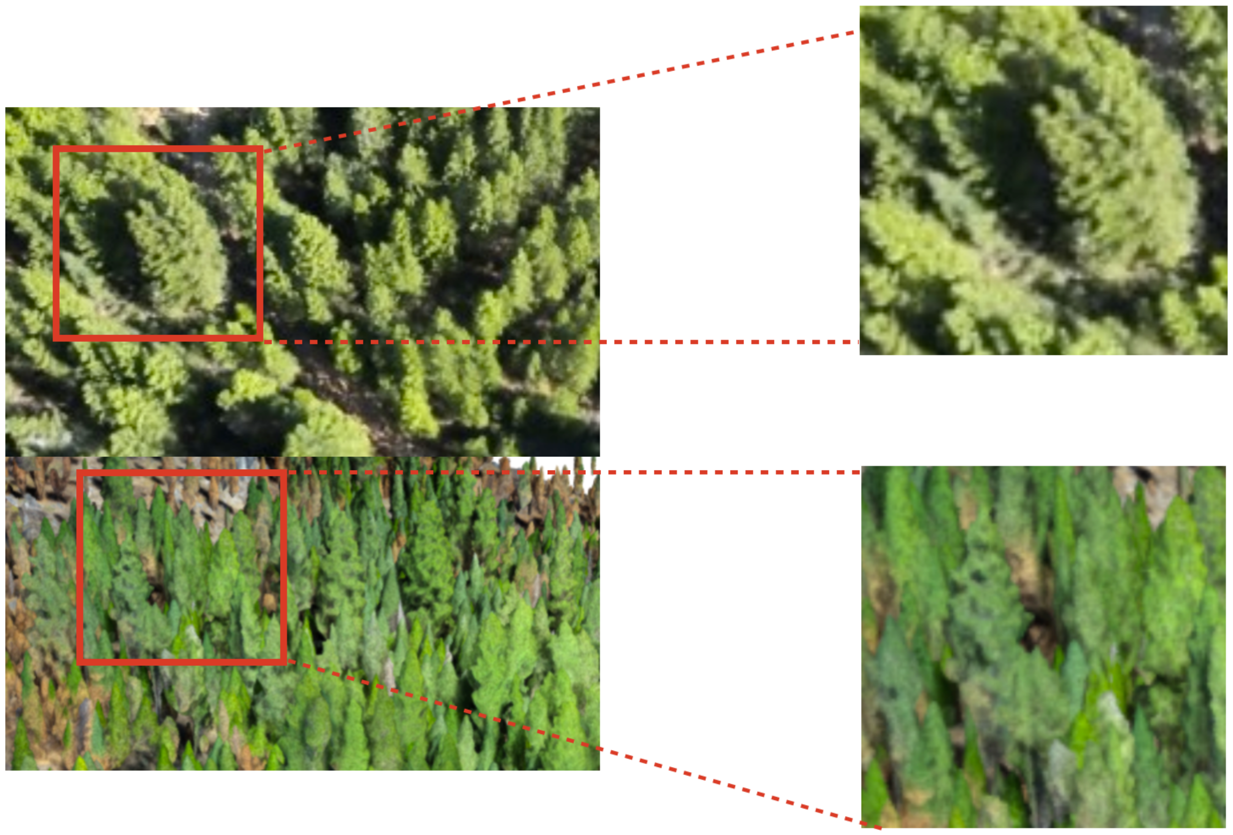
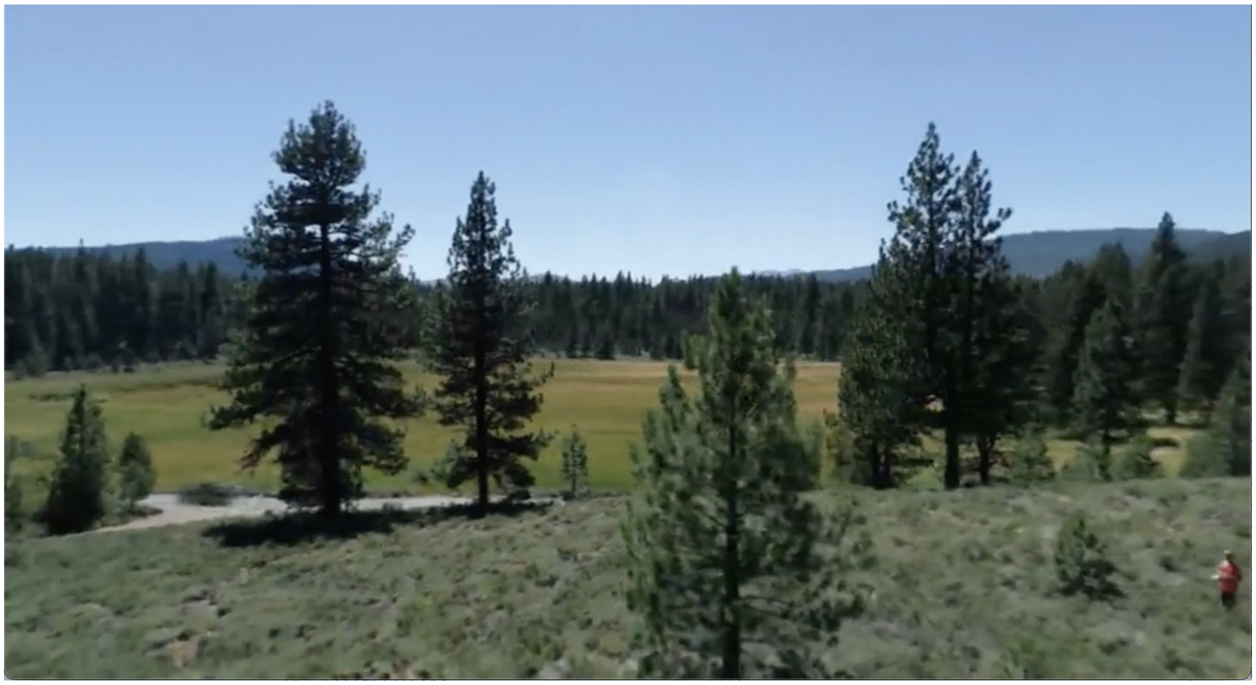
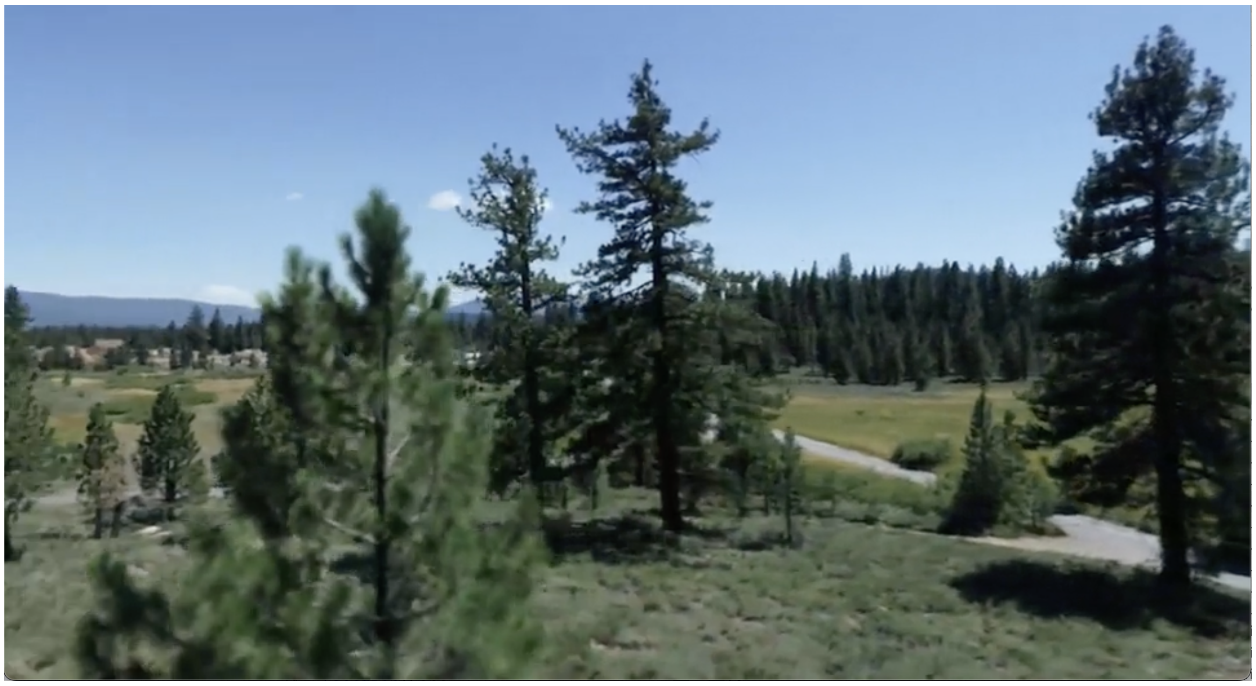


Figure 5.1: NeRF (top) versus current SfM mesh reconstruction (below) on overhead drone imagery. The NeRF retains more fine detail in the trees, most notably leaf and branch structure.

reconstructed scene often contains errors after triangulation, a refinement process called bundle adjustment is often applied. Structure-from-Motion is successful in creating 3D structure from 2D images, but it is ultimately limited by the data given it. It is not designed to reconstruct novel views, which can limit its usefulness in situations where certain views are difficult to collect, such as the understory of a dense forest.



(a)



(b)

Figure 5.2: Two views rendered views from a single NeRF created from drone-captured forest imagery. The NeRF provides significantly enhanced levels of photorealism.

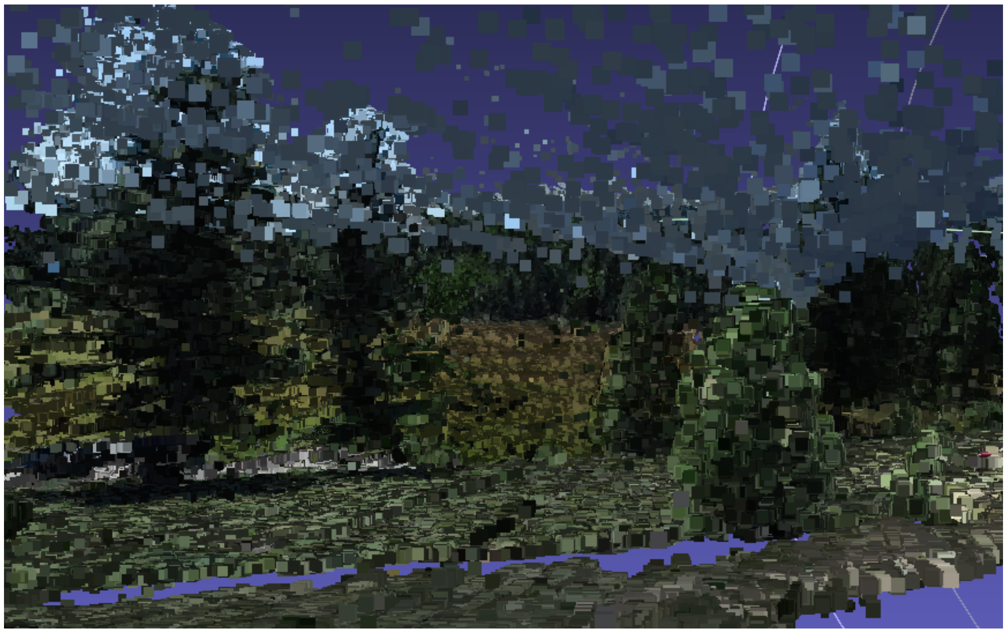
5.2 Introducing NeRFs to the OFO

The OFO is a program created by forest ecologists for other ecologists, land managers, and the general public. The OFO aims to integrate existing and emerging imagery processing tools and techniques, tune them for forestry applications, and present them through user-friendly interfaces. The OFO's initial focus on classical photogrammetry-based approaches has achieved high-quality mapping of the forest overstory [110], but the limited utility of these methods for understory reconstruction poses some constraints to widespread adoption by the forest ecology and management community. In particular, most forest inventory methods, management prescriptions, and ecological analyses rely critically on measurements of tree stem diameter at breast height (DBH), which can be challenging to obtain using photogrammetry even in sparse stands [90, 89].

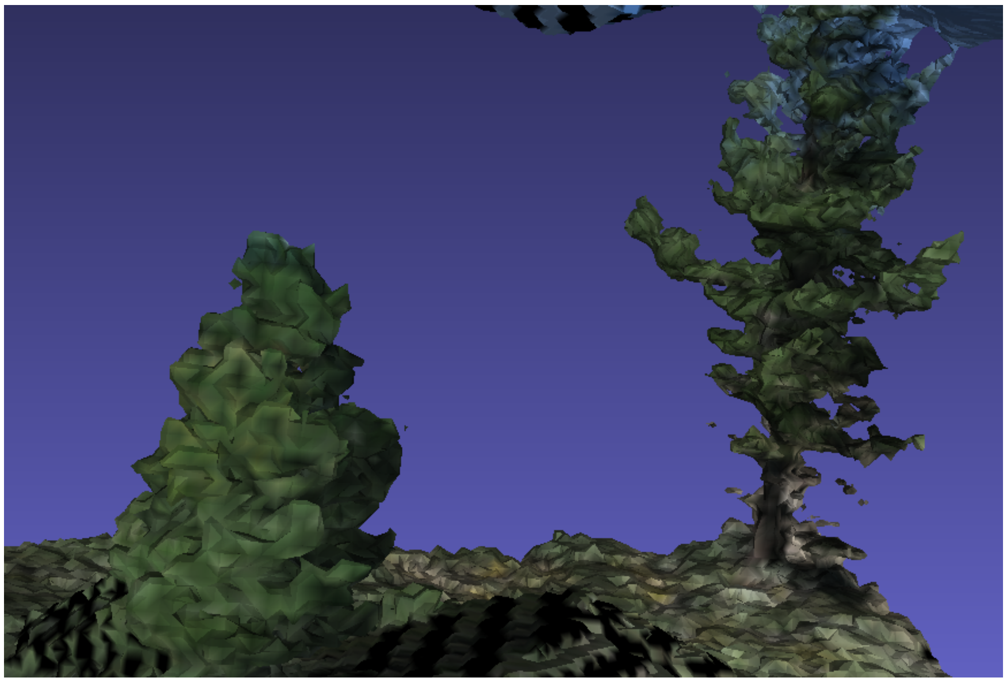
For this proof-of-concept work, we used an OFO video acquisition, collected using a DJI Phantom 4 Pro v2 drone, that included footage from many angles of an isolated lodgepole pine (*Pinus contorta*) on the Tahoe National Forest near Graegle, California (39.67 deg W, 120.62 deg N). We processed the data in nerfstudio [92] and used their nerf-facto method to create a NeRF of the scene (Fig. 5.2). It is immediately obvious the significantly enhanced photo-realism of the NeRF compared to a SfM created mesh (Fig. 5.3a). Detail such as branches, leaves, and trunks can be made out much more clearly which can aid in a variety of downstream tasks including species classification and trunk measurement. In Fig. 5.1, we show another example of a NeRF, this time created from overhead imagery. As the images were taken from further overhead, less detail is visible, but it is still more photorealistic than the current 3D model in the OFO taken on a similar stand of trees with the same image collecting procedure.

The mesh extracted from the NeRF also has improved detail. A SfM point-cloud generated from the same data has a significant number of floaters. In Fig. 5.3 we compare the two. The high number of artifacts prevents us from reasonably meshing the point-cloud. Those floaters can be removed through post-processing, but in general we are not seeing much detail in the individual trees, which is consistent with the SfM data already present in OFO. In addition to providing a richer visual experience, having more detail can serve a various scientific purposes. There are several qualitative tasks that typically require an expert walking through a forest, such as fuel load estimation or density verification, that could potentially be done virtually with a NeRF, saving significant travel time and costs as many forests are quite remote. NeRFs also allow the 3D qualitative experience to be saved, enabling more robust comparisons across time.

This mesh also has the potential to be used for PDE simulation. Relatively little work has been done in high fidelity 3D wildfire simulations for two primary reasons: (1) it is difficult to get access to high quality 3D geometries, so 2D elevation maps are used instead; (2) Correctly



(a) Point-cloud generated via SfM.



(b) Mesh extracted from NeRF.

Figure 5.3: A side by side of a 3D point-cloud and a 3D mesh generated via different reconstruction methods of the same imagery. The COLMAP point-cloud is shown instead of a mesh, because there were so many stray points and other artifacts that the point-cloud could not be reasonably meshed. NeRF by comparison yields a higher-quality, detailed mesh reconstruction of individual trees.

simulating the physics in 3D is time consuming and is not well-suited to real-time information which is what is most desired in an active fire scenario. However, if 3D geometries become easily accessible, we believe this could spur development on the simulation side. There are several applications of wildfire modeling that do not require real-time solutions, including predictive modeling for insurance and development purposes.

5.3 Climate and Environmental Impacts

Broad-extent forest inventory data is critical for informing management of forests in our era of changing climate, increasing drought stress, and unnaturally high-severity wildfires. Due to a century of intensive fire suppression and exclusion of indigenous forest stewardship through fire, dry forests in the western U.S. – and many other areas around the globe – have become unnaturally dense [78]. With more trees competing for the same finite pool of resources, particularly water, such "overstocked" stands are at high risk of mortality due to drought [112]. The closely spaced trees, with an abundance of flammable material in the understory, put the stands at high risk of near complete mortality in the event of a wildfire [2]. Forest mortality, whether through drought, fire, or other environmental stressors, has clear implications for carbon storage and myriad other ecosystem services. Understanding tree density is essential for proper forest management.

Forest management such as mechanical thinning, prescribed fire, and reintroduction of beneficial wildfire can greatly ameliorate the stresses associated with unnaturally dense forest stands and improve forest resistance to drought and fire [111], thus reducing the risk of catastrophic forest and carbon loss. However, resources for such forest management are stretched thin, and only a small fraction of forest area needing management each year receives treatment [66, 65]. Thus, data to inform efforts to prioritize forest management across space and time is critical. Because in-person assessments and surveys for all forests are not practical, modern tools to automate and virtualize forest measurement could thus provide substantial value to forest management efforts. This data (e.g. species identification, density and biomass estimation, DBH etc.) also serve as an additional metrics to judge reconstruction quality. Such prioritization depends on inventory data, which is costly and time-consuming to collect by traditional means. The OFO aims to greatly increase the efficiency of such data collection using drone-based alternatives to traditional ground-based surveys and enable individuals such as forest managers to both collect and process the data into extensive forest inventory maps suitable for informing management decisions. The work presented here, and future expansions upon it, will help to improve the fidelity of drone-derived products to the real-world stand conditions, thus increasing their relevance for informing management to maintain healthy forests.

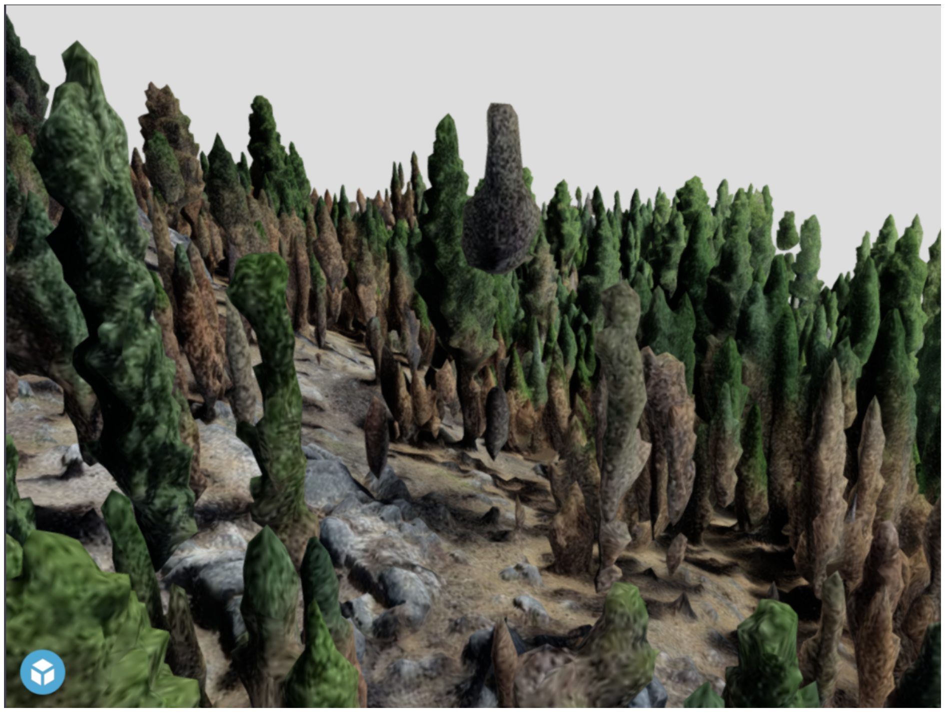


Figure 5.4: A SfM reconstruction where trees are floating off the ground. Future work is motivated around using NeRF-based technologies to fill in the missing geometry enabling 3D simulation.

5.4 Uncertainty in Forest Renderings

The wealth of data in the Open Forest Observatory gives us an opportunity to demonstrate various aspects of our Image-to-PDE pipeline as well as highlight points of friction that would benefit from improvement. Starting with geometry generation, we showed in the previous sections how a NeRF can be built from OFO data. Neuralangelo similarly can be used to get high quality renderings. While the geometry in this case leaves much to be desired, we can still apply our uncertainty model to the color. In Fig. 5.5 we see a novel view rendering created with NeuralAngleo. There are small artifacts at the tree top level with rays being hallucinated from some of the trees. The uncertainty heat-map correctly identifies these areas as areas of high uncertainty while indicating that the rest of the image is relatively certain.

As the SDF that was learned with this rendering was not of the desired quality, we did not apply our SDF uncertainty method (only the color uncertainty), but we expect it could be applied similarly. In this application we could see several uses of this uncertainty information.

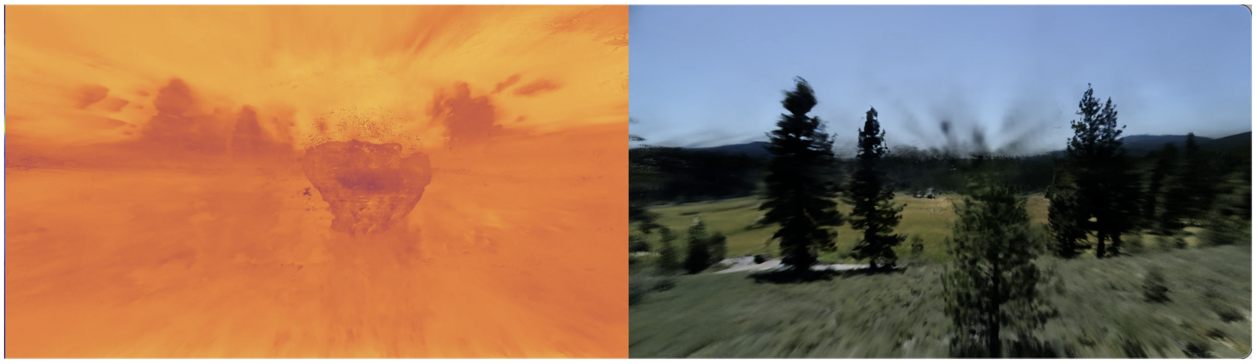


Figure 5.5: Novel image rendering of an OFO tree scene using NeuralAngelo, rather than NeRF, with accompanying uncertainty heat-map generated by the method outlined in Ch. 4. Darker colors indicates higher uncertainty

By indicating to the user where the rendering is uncertain, it enables the user to decide to collect more imagery or not. Of further usefulness would be connecting the uncertainty to the physics to give an indication of how this uncertainty will affect the simulation. In a wildfire simulation, it may or may not matter how accurately the tree-tops are rendered.

5.5 Future Work

Using traditional NeRF can already transform how the Open Forest Observatory can be used by forest managers, significantly improving reconstruction quality enabling virtual assessments among other applications. However, there are several variants of NeRF that could provide additional improvements. The next immediate goal is to greatly scale up the size of the NeRF taking inspiration from other large-scale NeRF efforts [91]. Next, we wish to explore methods to improve the quality of the underlying geometric structure. We have done some preliminary experiments with signed-distance function based methods such as neus-facto (implemented in SDFstudio [113]) and Neuralangelo [49], but despite decent quality renderings we were failing to see higher quality meshes. In practice, we have seen that NeuS and Neuralangelo have high sensitivity to camera angles in producing their mesh, but by converting the camera information saved by the drone into a SDFstudio compatible format, we do think it is feasible to generate higher quality SDFs for simulation. This work is still on-going. We also wish to do a better job filling in the hard-to-image understory: Trees often appear to be floating floating (Fig. 5.4). We plan to explore approaches that combine diffusion models with NeRF [17, 35, 107]. We believe forests are particularly well suited to this approach as we have a strong prior that most everything we are imaging is a tree, and there are basic principles such as “trees should be connected to the ground" that we can enforce to improve reconstruction quality. We are also currently exploring 3D in-painting methods such as Inpaint3D [69] which use 2D images to in-paint a 3D scene, fixing errors. Can we use the photorealisitic NeRF reconstruction to get a good 2D image of a novel view and then use that to fill-in the missing pieces of the SfM reconstruction? Could we also use the SfM as an initialization for NeuS? There are many ideas for combining the various 3D methods available to improve the final 3D geometry.

Another exciting extension of NeRF that could aid in forestry management is the incorporation of language models. LERF [43] allows for natural-language searching through a NeRF, which if fine-tuned on species level forest data could be very useful for finding and counting species. As we see more multi-model models that can handle question answering, we could imagine merging those with NeRF as well to give more information about the 3D environment such as stand density, basal area, or biomass, all of which are important metrics in estimating carbon sequestration and fuel sources for wildfires.

The OFO is compiling extensive imagery and ground truth data (in the form of geospatial locations, species labels, and diameter measurements) of forest stands across the western U.S. and ultimately around the world. We plan to curate a subset of the data in the form of well-defined “challenge problems" to spur the development of new vision methods specifically designed for this high-impact application area. Already metrics for 3D reconstruction quality are a hotly debated topic in the community, as what constitutes an acceptable versus an unacceptable error is often dependent on the downstream application. In many forestry applications the key metric is faithful reconstruction of a tree’s stem diameter at breast height (DBH), given it is the size metric around which the vast majority of forest ecology models and management prescriptions are based. With a specific downstream application in mind, along with extensive and high-quality ground-truth DBH data, we hope to provide a new application area and validation metric to the 3D reconstruction community.

Chapter 6

Final Words

In this thesis we explored technologies from several branches of computational science and detailed how they could be put together to create an image-to-simulation pipeline that would greatly broaden the real-world scenarios we are able to simulate. We explored several approaches for learning signed distance functions from images, finding much potential in the NeuS family of algorithms.

We paired NeuS with Embedded Boundary PDE simulation, analyzing the various parameters that can affect error in geometry and how that error in geometry translates to error in simulation. Additionally we examined how traditional 3D geometry metrics fail to capture the errors most relevant to EB simulation, so we provided our own framework for evaluating neural reconstruction techniques in the context of PDE simulation. While the method we used in our experiments, NeuralAngelo, represents the current state of the art in NeuralSDF inference from images, the field moves very quickly and we expect new methods to soon emerge. In the past, the fast pace of 3D reconstruction research has made it difficult to transfer to scientific disciplines - it can be challenging for the uninitiated to know what to focus on. We hope that the introduction of a systematic evaluation framework specifically designed for use with numerical simulation can prepare numerical analysts to more easily evaluate new methods as the are developed.

In our analysis of the NeuS family of methods, we saw the limitations of only evaluating error post-facto. In most situations, we will not have ground truth to compare to, or really much concrete understanding of error, and there is so much variation in input data (images available, lighting of scene, geometry of the objects, etc.) that each test reconstruction problem does not easily generalize to others. This motivated the necessity of a model-specific understanding of uncertainty. We turned towards the literature in NeRF and outlined how to apply it to NeuS, along with modifications to support specifically error in the SDF geometry as opposed to just error in the rendering. We showed the efficacy of our method as opposed to just measuring uncertainty in rendering in some small test problems and also highlighted many areas for improvement. Most importantly, we wish to fully integrate this understanding of uncertainty in geometry to a fuller understanding of uncertainty in physics by propagating the uncertainty measure through the EB calculations.

Finally, as this thesis is primarily concerned with the practical integration of tools for solving real-world problems, we ground all of this work in a very impact real-world application. Proper forest management is increasingly important as our climate changes, for many reasons including wildfire prevention, carbon sequestration, and biodiversity preservation. We illustrated the data that our partners at the Open Forest Observatory have collected to aid in this vital task, and how all of the technologies outlined in this thesis have and will continue to support the creation of this valuable open-source dataset.

As this work was the first to pair these technologies together, it is just a start to building an automatic, robust pipeline. Part of our contribution is to identify and pose the interesting questions that arise by putting these technologies together. One of the most important of these questions is how to fully propagate uncertainty through the physics model and possibly even bring that physics uncertainty back in to the image reconstruction model. There exists an extensive literature in uncertainty propagation [56, 64] in computational fluid dynamics that could be applied to this problem, although there is no work yet on specifically performing uncertainty quantification through an Embedded Boundary simulation. This provides a rich area for future work. Supposing we have an uncertainty metric, the physics uncertainty could also be used as feedback to the initial 3D reconstruction network. We could penalize geometries that have highly uncertain physics or break a law of physics, leading to an improved overall geometry.

The collaboration with the Open Forest Observatory also highlights the various weak points in this pipeline. In this work, we are mostly assuming that high quality images are relatively easy to obtain. While they certainly are easier to obtain than LiDAR scans or other 2.5D/3D imagery, there are still hurdles. For example, it is difficult to obtain good imagery of the understory of a forest as manual drone navigation through a forest can be tricky. However, there are improvements in obstacle avoidance algorithms that could enable autonomous image collection. Using this imagery to build a NeRF and combining it with the overstory imagery, could increase fidelity. There are also many NeRF-based methods that incorporate diffusion and/or 3D in-painting [69] that could be used to fill in parts of the forest that are difficult to image. From there, the Embedded Boundary method could be used to perform a 3D wildfire simulation, and using the machinery we have built up in this work, we could also characterize the error and uncertainty of that simulation to be able to make informed forest management decisions.

Computer vision has huge potential to aid in scientific tasks, but work still has to be done to bridge the gap between cutting-edge research results and messy, real-world data.

Scientists must be able to quantify and understand the strengths and weaknesses of any given model, and we present this thesis as a step towards enabling trust in neural approaches in the scientific process.

Bibliography

[1] Michael J Aftosmis, Marsha J Berger, and John E Melton. “Robust and efficient Cartesian mesh generation for component-based geometry”. In: *AIAA journal* 36.6 (1998), pp. 952–960.

[2] Victoria JW Amato, David Lightfoot, Cody Stropki, and Michael Pease. “Relationships between tree stand density and burn severity as measured by the Composite Burn Index following a ponderosa pine forest wildfire in the American Southwest”. In: *Forest ecology and management* 302 (2013), pp. 71–84.

[3] Ma Baorui, Han Zhizhong, Liu Yu-Shen, and Zwicker Matthias. “Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces”. In: *International Conference on Machine Learning (ICML)*. 2021.

[4] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Joshua B. Tenenbaum, and Boris Katz. “ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models”. In: *NeurIPS*. 2019.

[5] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. “The ball-pivoting algorithm for surface reconstruction”. In: *IEEE transactions on visualization and computer graphics* 5.4 (1999), pp. 349–359.

[6] Bo Li, Chunhua Shen, Yuchao Dai, A. van den Hengel, and Mingyi He. “Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[7] Nicolò Camarretta, Peter A Harrison, Tanya Bailey, Brad Potts, Arko Lucieer, Neil Davidson, and Mark Hunt. “Monitoring forest structure to guide adaptive management of forest restoration: a review of remote sensing approaches”. In: *New Forests* 51 (2020), pp. 573–596.

[8] Angel X. Chang et al. “ShapeNet: An Information-Rich 3D Model Repository”. In: *Arxiv*. 2015.

[9] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. "TensoRF: Tensorial Radiance Fields". In: 2022. arXiv: `2203.09517 [cs.CV]`.

[10] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. "Single-Image Depth Perception in the Wild". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: `https://proceedings.neurips.cc/paper/2016/file/0deb1c54814305ca9ad266f53bc82511-Paper.pdf`.

[11] Zhiqin Chen and Hao Zhang. "Learning implicit fields for generative shape modeling". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[12] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G Schwing, and Liang-Yan Gui. "Sdfusion: Multimodal 3d shape completion, reconstruction, and generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 4456–4465.

[13] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction". In: *eccv*. 2016.

[14] P Colella, D. T. Graves, T. J. Ligocki, G.H. Miller, D. Modiano, P.O. Schwartz, B. Van Straalen, J. Pillod, D. Trebotich, and M. Barad. "EBChombo software package for Cartesian grid, embedded boundary applications". In: (2014).

[15] Michael G Crandall and Pierre-Louis Lions. "Viscosity solutions of Hamilton-Jacobi equations". In: *Transactions of the American mathematical society* 277.1 (1983), pp. 1–42.

[16] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. "Laplace redux-effortless bayesian deep learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 20089–20103.

[17] Congyue Deng, Chiyu Jiang, Charles R Qi, Xinchen Yan, Yin Zhou, Leonidas Guibas, Dragomir Anguelov, et al. "Nerdi: Single-view nerf synthesis with language-guided diffusion as general image priors". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 20637–20647.

[18] Dharshi Devendran, Daniel Graves, Hans Johansen, and Terry Ligocki. "A fourth-order Cartesian grid embedded boundary method for Poisson's equation". In: *Communications in Applied Mathematics and Computational Science* 12.1 (2017), pp. 51–79.

[19] D. Eigen and R. Fergus. "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale Convolutional Architecture". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015.

[20] David Eigen, Christian Puhrsch, and Rob Fergus. "Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. 2014.

[21] Haoqiang Fan, Hao Su, and Leonidas Guibas. "A Point Set Generation Network for 3D Object Reconstruction from a Single Image". In: *CVPR*. 2017.

[22] Péter Fankhauser, Michael Bloesch, Diego Rodriguez, Ralf Kaestner, Marco Hutter, and Roland Siegwart. "Kinect v2 for mobile robot navigation: Evaluation and modeling". In: *2015 International Conference on Advanced Robotics (ICAR)*. 2015, pp. 388–394. DOI: 10.1109/ICAR.2015.7251485.

[23] Matheus Pinheiro Ferreira, Danilo Roberti Alves de Almeida, Daniel de Almeida Papa, Juliano Baldez Silva Minervino, Hudson Franklin Pessoa Veras, Arthur Formighieri, Caio Alexandre Nascimento Santos, Marcio Aurélio Dantas Ferreira, Evandro Orfano Figueiredo, and Evandro José Linhares Ferreira. "Individual tree detection and species classification of Amazonian palms using UAV images and deep learning". In: *Forest Ecology and Management* 475 (2020), p. 118397.

[24] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. "Deep Ordinal Regression Network for Monocular Depth Estimation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018.

[25] Matheus Gadelha, Rui Wang, and Subhransu Maji. "Multiresolution Tree Networks for 3D Point Cloud Processing". In: *ECCV*. 2018.

[26] Ravi Garg, BG Vijay Kumar, Gustavo Carneiro, and Ian Reid. "Unsupervised CNN for single view depth estimation: Geometry to the rescue". In: *European Conference on Computer Vision*. Springer. 2016, pp. 740–756.

[27] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.

[28] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. "Mesh r-cnn". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9785–9795.

[29] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. "Unsupervised Monocular Depth Estimation with Left-Right Consistency". In: *CVPR*. 2017.

[30] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. “Digging into Self-Supervised Monocular Depth Prediction”. In: (Oct. 2019).

[31] Shubham Goel, Angjoo Kanazawa, and Jitendra Malik. “Shape and Viewpoints without Keypoints”. In: *ECCV*. 2020.

[32] Lily Goli, Cody Reading, Silvia Selllán, Alec Jacobson, and Andrea Tagliasacchi. “Bayes’ Rays: Uncertainty Quantification for Neural Radiance Fields”. In: *arXiv preprint arXiv:2309.03185* (2023).

[33] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. “Implicit geometric regularization for learning shapes”. In: *arXiv preprint arXiv:2002.10099* (2020).

[34] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. “AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation”. In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[35] Jiatao Gu, Alex Trevithick, Kai-En Lin, Joshua M Susskind, Christian Theobalt, Lingjie Liu, and Ravi Ramamoorthi. “Nerfdiff: Single-image view synthesis with nerf-guided distillation from 3d-aware diffusion”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 11808–11826.

[36] C. Häne, S. Tulsiani, and J. Malik. “Hierarchical Surface Prediction for 3D Object Reconstruction”. In: *3DV*. 2017.

[37] Miles Hansard, Seungkyu Lee, Ouk Choi, and Radu Horaud. *Time of Flight Cameras: Principles, Methods, and Applications*. Oct. 2012.

[38] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[39] Hans Johansen and Phillip Colella. “A Cartesian grid embedded boundary method for Poisson’s equation on irregular domains”. In: *Journal of Computational Physics* 147.1 (1998), pp. 60–85.

[40] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. “Learning Category-Specific Mesh Reconstruction from Image Collections”. In: *ECCV*. 2018.

[41] Michael Kazhdan and Hugues Hoppe. “Screened Poisson Surface Reconstruction”. In: *ACM Trans. Graph.* 32.3 (July 2013). ISSN: 0730-0301. DOI: `10.1145/2487228.2487237`. URL: `https://doi.org/10.1145/2487228.2487237`.

[42] Michael Kazhdan and Hugues Hoppe. “Screened poisson surface reconstruction”. In: *ACM Transactions on Graphics (ToG)* 32.3 (2013), pp. 1–13.

[43] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. “Lerf: Language embedded radiance fields”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 19729–19739.

[44] Kourosh Khoshelham and Sander Oude Elberink. “Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications”. In: *Sensors (Basel, Switzerland)* 12 (2012), pp. 1437–1454.

[45] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. “Deeper depth prediction with fully convolutional residual networks”. In: *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE. 2016, pp. 239–248.

[46] Erich L Lehmann and George Casella. *Theory of point estimation*. Springer Science & Business Media, 2006.

[47] Ruibo Li, Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, and Lingxiao Hang. “Deep attention-based classification network for robust depth prediction”. In: *Asian Conference on Computer Vision*. Springer. 2018, pp. 663–678.

[48] Xueting Li, Sifei Liu, Kihwan Kim, Shalini De Mello, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. “Self-supervised Single-view 3D Reconstruction via Semantic Consistency”. In: *ECCV*. 2020.

[49] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. “Neuralangelo: High-Fidelity Neural Surface Reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 8456–8465.

[50] Zhengqi Li and Noah Snavely. “MegaDepth: Learning Single-View Depth Prediction from Internet Photos”. In: *Computer Vision and Pattern Recognition (CVPR)*. 2018.

[51] T. Ligocki. “Poster: Embedded Boundary Grid Generation using the Divergence Theorem, Implicit FUnctions and Constructive Solid Geometry”. In: *Applied Partial Differential Equations Center for Enabling Technologies (APDEC)*. 2008.

[52] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. “SDF-SRN: Learning Signed Distance 3D Object Reconstruction from Static Images”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[53] Fayao Liu, Chunhua Shen, and Guosheng Lin. “Deep Convolutional Neural Fields for Depth Estimation from a Single Image”. In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition*. 2015. URL: `http://arxiv.org/abs/1411.6387`.

[54] Shitong Luo and Wei Hu. “Diffusion Probabilistic Models for 3D Point Cloud Generation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 2837–2845.

[55] Yue Luo, Jimmy Ren, Mude Lin, Jiahao Pang, Wenxiu Sun, Hongsheng Li, and Liang Lin. “Single View Stereo Matching”. In: *CVPR*. 2018.

[56] Lionel Mathelin, M Yousuff Hussaini, and Thomas A Zang. “Stochastic approaches to uncertainty quantification in CFD simulations”. In: *Numerical Algorithms* 38 (2005), pp. 209–236.

[57] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. “Occupancy networks: Learning 3d reconstruction in function space”. In: *CVPR*. 2019.

[58] S. Mahdi H. Miangoleh, Sebastian Dille, Long Mai, Sylvain Paris, and Yağız Aksoy. “Boosting Monocular Depth Estimation Models to High-Resolution via Content-Adaptive Multi-Resolution Merging”. In: 2021.

[59] 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: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[60] 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: *CVPR*. 2020.

[61] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. “AutoSDF: Shape Priors for 3D Completion, Reconstruction and Generation”. In: *CVPR*. 2022.

[62] Reason Mlambo, Iain H Woodhouse, France Gerard, and Karen Anderson. “Structure from motion (SfM) photogrammetry with drone data: A low cost method for monitoring greenhouse gas emissions from forests in developing countries”. In: *Forests* 8.3 (2017), p. 68.

[63] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding”. In: *ACM Trans. Graph.* 41.4 (July 2022), 102:1–102:15. DOI: `10.1145/3528223.3530127`. URL: `https://doi.org/10.1145/3528223.3530127`.

[64] Habib N Najm. “Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics”. In: *Annual review of fluid mechanics* 41 (2009), pp. 35–52.

[65] Malcolm North, April Brough, Jonathan Long, Brandon Collins, Phil Bowden, Don Yasuda, Jay Miller, and Neil Sugihara. “Constraints on mechanized treatment significantly limit mechanical fuels reduction extent in the Sierra Nevada”. In: *Journal of Forestry* 113.1 (2015), pp. 40–48.

[66] MP North, RA York, BM Collins, MD Hurteau, GM Jones, EE Knapp, L Kobziar, H McCann, MD Meyer, SL Stephens, et al. "Pyrosilviculture needed for landscape resilience of dry western United States forests". In: *Journal of Forestry* 119.5 (2021), pp. 520–544.

[67] Stanley Osher and James A Sethian. "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations". In: *Journal of computational physics* 79.1 (1988), pp. 12–49.

[68] 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*. 2019, pp. 165–174.

[69] Kira Prabhu, Jane Wu, Lynn Tsai, Peter Hedman, Dan B Goldman, Ben Poole, and Michael Broxton. "Inpaint3D: 3D Scene Content Generation using 2D Inpainting Diffusion". In: *arXiv preprint arXiv:2312.03869* (2023).

[70] Marissa Ramirez de Chanlatte, Matheus Gadelha, Thibault Groueix, and Radomir Mech. "Recovering Detail in 3D Shapes Using Disparity Maps". In: *arXiv preprint arXiv:2207.00182* (2022).

[71] Marissa Ramirez de Chanlatte, Arjun Rewari, Trevor Darrell, and Derek J. N. Young. "Neural Tree Reconstruction for the Open Forest Observatory". In: *arXiv preprint arXiv:2207.00182* (2022).

[72] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. "Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2020).

[73] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer". In: *IEEE transactions on pattern analysis and machine intelligence* (2020).

[74] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. "OctNetFusion: Learning Depth Fusion from Data". In: *3DV*. 2017.

[75] Hippolyt Ritter, Aleksandar Botev, and David Barber. "A scalable laplace approximation for neural networks". In: *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*. Vol. 6. International Conference on Representation Learning. 2018.

[76] Lawrence G Roberts. "Machine perception of three-dimensional solids". In: 1963.

[77] Anirban Roy and Sinisa Todorovic. “Monocular depth estimation using neural regression forest”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5506–5514.

[78] Hugh D Safford, Jens T Stevens, et al. *Natural range of variation for yellow pine and mixed-conifer forests in the Sierra Nevada, southern Cascades, and Modoc and Inyo National Forests, California, USA*. United States Department of Agriculture, Forest Service, Pacific Southwest …, 2017.

[79] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. “Plenoxels: Radiance Fields without Neural Networks”. In: *CVPR*. 2022.

[80] A. Saxena, M. Sun, and A. Y. Ng. “Make3D: Learning 3D Scene Structure from a Single Still Image”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2009.

[81] Johannes L Schonberger and Jan-Michael Frahm. “Structure-from-motion revisited”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4104–4113.

[82] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. “A Multi-View Stereo Benchmark with High-Resolution Images and Multi-Camera Videos”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[83] Peter Schwartz, David Adalsteinsson, Phillip Colella, Adam Paul Arkin, and Matthew Onsum. “Numerical computation of diffusion on a surface”. In: *Proceedings of the National Academy of Sciences* 102.32 (2005), pp. 11151–11156.

[84] Peter Schwartz, Julie Percelay, Terry J. Ligocki, Hans Johansen, Daniel T. Graves, Dharshi Devendran, Phillip Colella, and Eli Ateljevich. “High-accuracy embedded boundary grid generation using the divergence theorem”. In: *Communications in Applied Mathematics and Computational Science* 10 (1 2015), pp. 83–96.

[85] Daeyun Shin, Charless Fowlkes, and Derek Hoiem. “Pixels, voxels, and views: A study of shape representations for single view 3D object shape prediction”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[86] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. “Nerv: Neural reflectance and visibility fields for relighting and view synthesis”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 7495–7504.

[87] Wen Su, Haifeng Zhang, Jia Li, Wenzhen Yang, and Zengfu Wang. “Monocular Depth Estimation as Regression of Classification Using Piled Residual Networks”. In: 2019.

[88] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. “Pix3d: Dataset and methods for single-image 3d shape modeling”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2974–2983.

[89] Neal C Swayze and Wade T Tinkham. “Application of unmanned aerial system structure from motion point cloud detected tree heights and stem diameters to model missing stem diameters”. In: *MethodsX* 9 (2022), p. 101729.

[90] Neal C Swayze, Wade T Tinkham, Jody C Vogeler, and Andrew T Hudak. “Influence of flight parameters on UAS-based monitoring of tree height, diameter, and density”. In: *Remote Sensing of Environment* 263 (2021), p. 112540.

[91] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. “Block-nerf: Scalable large scene neural view synthesis”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 8248–8258.

[92] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. “Nerfstudio: A modular framework for neural radiance field development”. In: *ACM SIGGRAPH 2023 Conference Proceedings*. 2023, pp. 1–12.

[93] M. Tatarchenko, A. Dosovitskiy, and T. Brox. “Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs”. In: *iccv*. 2017.

[94] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. “What do single-view 3d reconstruction networks learn?” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3405–3414.

[95] Anh Thai, Stefan Stojanov, Vijay Upadhya, and James M Rehg. “3d reconstruction of novel object shapes from single images”. In: *2021 International Conference on 3D Vision (3DV)*. IEEE. 2021, pp. 85–95.

[96] Shubham Tulsiani, Nilesh Kulkarni, and Abhinav Gupta. “Implicit mesh reconstruction from unannotated image collections”. In: *arXiv preprint arXiv:2007.08504* (2020).

[97] Kalyan Alwala Vasudev, Abhinav Gupta, and Shubham Tulsiani. “Pre-train, Self-train, Distill: A simple recipe for Supersizing 3D Reconstruction”. In: *Computer Vision and Pattern Recognition (CVPR)*. 2022.

[98] Chaoyang Wang, Simon Lucey, Federico Perazzi, and Oliver Wang. "Web stereo video supervision for depth prediction from dynamic scenes". In: *2019 International Conference on 3D Vision (3DV)*. IEEE. 2019, pp. 348–357.

[99] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images". In: *ECCV*. 2018.

[100] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction". In: *arXiv preprint arXiv:2106.10689* (2021).

[101] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. "Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 3295–3306.

[102] Ben G Weinstein, Sergio Marconi, Sarah J Graves, Alina Zare, Aditya Singh, Stephanie A Bohlman, Lukas Magee, Daniel J Johnson, Phillip A Townsend, and Ethan P White. "Capturing long-tailed individual tree diversity using an airborne imaging and a multi-temporal hierarchical model". In: *Remote Sensing in Ecology and Conservation* (2023).

[103] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, William T Freeman, and Joshua B Tenenbaum. "MarrNet: 3D Shape Reconstruction via 2.5D Sketches". In: *Advances In Neural Information Processing Systems*. 2017.

[104] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling". In: *Advances in Neural Information Processing Systems*. 2016, pp. 82–90.

[105] Jiajun Wu, Chengkai Zhang, Xiuming Zhang, Zhoutong Zhang, William T Freeman, and Joshua B Tenenbaum. "Learning 3D Shape Priors for Shape Completion and Reconstruction". In: *European Conference on Computer Vision (ECCV)*. 2018.

[106] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. "Beyond PASCAL: A Benchmark for 3D Object Detection in the Wild". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2014.

[107] Guandao Yang, Abhijit Kundu, Leonidas J Guibas, Jonathan T Barron, and Ben Poole. "Learning a Diffusion Prior for NeRFs". In: *arXiv preprint arXiv:2304.14473* (2023).

[108] Yufei Ye, Shubham Tulsiani, and Abhinav Gupta. "Shelf-Supervised Mesh Prediction in the Wild". In: *Computer Vision and Pattern Recognition (CVPR)*. 2021.

[109] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. “Learning To Recover 3D Scene Shape From a Single Image”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 204–213.

[110] Derek JN Young, Michael J Koontz, and JonahMaria Weeks. “Optimizing aerial imagery collection and processing parameters for drone-based individual tree mapping in structurally complex conifer forests”. In: *Methods in Ecology and Evolution* 13.7 (2022), pp. 1447–1463.

[111] Derek JN Young, Marc Meyer, Becky Estes, Shana Gross, Amarina Wuenschel, Christina Restaino, and Hugh D Safford. “Forest recovery following extreme drought in California, USA: natural patterns and effects of pre-drought management”. In: *Ecological Applications* 30.1 (2020), e02002.

[112] Derek JN Young, Jens T Stevens, J Mason Earles, Jeffrey Moore, Adam Ellis, Amy L Jirka, and Andrew M Latimer. “Long-term climate and competition explain forest mortality patterns under extreme drought”. In: *Ecology letters* 20.1 (2017), pp. 78–86.

[113] Zehao Yu, Anpei Chen, Bozidar Antic, Songyou Peng, Apratim Bhattacharyya, Michael Niemeyer, Siyu Tang, Torsten Sattler, and Andreas Geiger. *SDFStudio: A Unified Framework for Surface Reconstruction*. 2022. URL: `https://github.com/autonomousvision/sdfstudio`.

[114] Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Joshua B Tenenbaum, William T Freeman, and Jiajun Wu. “Learning to Reconstruct Shapes From Unseen Classes”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.