UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Inverse Rendering of Geometry

Permalink

https://escholarship.org/uc/item/5tk886jq

ISBN

9798186186090

Author

Mehta, Ishit

Publication Date

2026-08-05

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA SAN DIEGO

Inverse Rendering of Geometry

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Ishit Bhadresh Mehta

Committee in charge:

Professor Manmohan Chandraker, Co-Chair
Professor Ravi Ramamoorthi, Co-Chair
Professor Tzu-Mao Li
Professor Yusu Wang

2026

Copyright

Ishit Bhadresh Mehta, 2026

All rights reserved.

The Dissertation of Ishit Bhadresh Mehta is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2026

TABLE OF CONTENTS

Dissertation Approval Page iii

Table of Contents iv

List of Figures vii

List of Tables xi

Acknowledgements xii

Vita xv

Abstract of the Dissertation xvii

Chapter 1 Introduction 1
1.1 Motivation 2
1.2 Contributions 2
1.3 Dissertation Organization 4
1.4 Publications 4

Chapter 2 A Level Set Theory for Neural Implicit Evolution under Explicit Flows . . . 6
2.1 Introduction 6
2.2 Related Work 9
2.3 Background 11
2.4 Method 12
2.4.1 Lagrangian Deformation 12
2.4.2 Eulerian Deformation 13
2.5 Theoretical Comparisons 14
2.5.1 Differentiable Iso-Surface Extraction 15
2.5.2 Differentiable Surface Rendering 17
2.6 Applications 19
2.6.1 Curvature-based Deformation 19
2.6.2 Inverse Rendering of Geometry 22
2.6.3 User-defined Shape Editing 25
2.7 Additional Derivations and Results 26
2.7.1 Differentiable Surface Rendering 26
2.7.2 Curvature-based Deformation 29
2.7.3 Inverse Rendering 32
2.7.4 User-defined Deformation 32
2.8 Conclusion 36

Chapter 3 A Theory of Topological Derivatives for Inverse Rendering of Geometry . . 37
3.1 Introduction 37

3.2 Related Work . . . 39
3.3 Background and Problem Formulation . . . 40
3.4 Curves in 2D . . . 42
3.4.1 Shape Derivatives . . . 43
3.4.2 Topological Derivatives . . . 44
3.5 Surfaces in 3D . . . 49
3.5.1 Background on Shading and Visibility . . . 50
3.5.2 Topological Derivatives . . . 52
3.6 Additional Derivations and Results . . . 60
3.6.1 Topological Derivatives in 2D . . . 60
3.6.2 Initial Demonstration on Real Data . . . 63
3.7 Implementation Details . . . 66
3.8 Discussion . . . 68

Chapter 4 Locally Orderless Images for Optimization in Differentiable Rendering . . . 70
4.1 Introduction . . . 70
4.2 Related Work . . . 74
4.3 Method . . . 75
4.4 Results . . . 81
4.4.1 Differentiable Vectorization . . . 81
4.4.2 Differentiable Path Tracing . . . 83
4.4.3 Variational Optimization . . . 86
4.4.4 Differentiable Rasterization . . . 89
4.5 Choosing kernel parameters . . . 90
4.6 Runtime . . . 93
4.7 Implementation Details . . . 93
4.8 Discussion . . . 94

Chapter 5 Conclusion . . . 100
5.1 Future Directions: Inverse Design with Intuitive Control . . . 101

Appendix A Modulated Periodic Activations for Generalizable Local Functional Representations . . . 103
A.1 Introduction . . . 103
A.2 Related Work . . . 106
A.3 Method . . . 108
A.3.1 Modulated Periodic Activations . . . 109
A.3.2 Expressivity of Modulation . . . 111
A.3.3 Local Functional Representations . . . 111
A.3.4 Training procedure . . . 113
A.4 Experiments . . . 114
A.4.1 Global Functional Representation . . . 115
A.4.2 Local Functional Representations . . . 120
A.4.3 Image-based Relighting . . . 122

A.5 Conclusion 122

Bibliography 125

LIST OF FIGURES

Figure 2.1. Lagrangian representation: surface points **x** are advected by the flow field **V**. 7

Figure 2.2. Eulerian representation: each point stores an implicit value ϕ, evolved as $\phi \rightarrow \phi'$. 7

Figure 2.3. Inverse-rendering recovery. 8

Figure 2.4. **Method Overview.** 12

Figure 2.5. **MeshSDF$^{\theta}$ does not follow the level-set equation**. 15

Figure 2.6. **Tangential flow fields may deform surfaces when DVR/IDR is used for surface extraction.** 19

Figure 2.7. **The laplacian $\Delta\Phi$ of an MLP-defined level-set function is noisy**. 20

Figure 2.8. **Surface smoothing on parametric level-sets.** 21

Figure 2.9. We use an explicit mean-curvature flow-field to deform a parametrically defined implicit surface. 21

Figure 2.10. **Inverse rendering of high-genus shapes.** 22

Figure 2.11. **Optimizing triangle-meshes can introduce self intersections.** 24

Figure 2.12. **Qualitative comparison for inverse rendering using implicit representations.** 25

Figure 2.13. User-defined editing on parametric level-sets. 26

Figure 2.14. **Surface smoothing evolution on a half-noisy sphere.** 29

Figure 2.15. **Qualitative comparisons for surface smoothing.** 31

Figure 2.16. Ablation on smoothness regularization term in the flow-field for inverse rendering. 32

Figure 2.17. **LSIG [112] converges to a poor estimate with correct topology at initialization.** 33

Figure 2.18. **Qualitative comparisons for Inverse Rendering of Geometry.** 34

Figure 2.19. **User-defined rotation on an implicitly defined cylinder.** 35

Figure 3.1. **Pathologies of shape derivatives.** 38

Figure 3.2. (a) We derive topological derivatives for 2D rendering functionals. 45

Figure 3.3. **Reconstruction of vector images from raster input.** 46

Figure 3.4. Caption for LOF 48

Figure 3.5. **Visibility gradients drive topological changes.** 49

Figure 3.6. (a) We derive topological derivative at a point **y** on a surface Γ using a vanishing cone of perturbation originating at the camera C. 51

Figure 3.7. **Large steps in visibility optimization.** 54

Figure 3.8. Conical perturbation in 3D. 54

Figure 3.9. **Inverse rendering of geometry with complex topology.** 55

Figure 3.10. **Shape ambigram from a single image.** 59

Figure 3.11. Secondary visibility. 59

Figure 3.12. **Area sampling v/s edge sampling.** 62

Figure 3.13. **Preliminary results on 3D shape recovery with real data.** 62

Figure 3.14. **Generating complex vector graphics using score-distillation [129]** We show additional results for the task of reconstructing vector images using a non-standard reconstruction loss. 64

Figure 3.15. Secondary visibility. 65

Figure 3.16. **Reconstructing shape ambigrams** We aid the `GEB` illustration (Figure 9) shown in the manuscript with mid-optimization renderings for NIE [97] + Mitsuba 3 [60] and our method. 66

Figure 3.17. Samples from the set of images used for multi-view reconstruction. 67

Figure 3.18. A preliminary experiment jointly recovering geometry and color for 2D vectorization. 69

Figure 4.1. Image gradients are sparse with respect to optimization parameters that induce motion in the image space ⑥. 71

Figure 4.2. **Inner and Extent scale spaces.** 73

Figure 4.3. **Scale-space matching extends gradient support.** 76

Figure 4.4. **Tonal Separation.** 78

Figure 4.5. **Histogram matching is less sensitive to noise.** 79

Figure 4.6. **Comparisons with multi-scale methods.** 81

Figure 4.7. **Quantitative comparisons with parameter-space blurring.** 84

Figure 4.8. **Why PRDPT fails on Shadow Mini.** 85

Figure 4.9. **Variational Optimization with Locally Orderless Images.** 87

Figure 4.10. **Using a real photograph.** 89

Figure 4.11. **Texture Recovery.** 91

Figure 4.12. **Ablation on α, σ.** 92

Figure 4.13. **Runtime Analysis.** 92

Figure 4.14. **Histogram visualization.** 94

Figure 4.15. Our method remains robust to different choices of the β-kernel width when recovering a textured plane. 95

Figure 4.16. **Visual comparisons on rasterization benchmark.** 96

Figure 4.17. **More comparisons on Differentiable Path Tracing.** 97

Figure 4.18. **2D Comparisons.** 98

Figure 4.19. Our method is not designed to recover parameters that do not influence the image, such as the rotation of the mug [41]. 99

Figure A.1. (*Top*) We propose a new method to encode discrete signals as neural-functional representations. 104

Figure A.2. **SIREN struggles with high-res signals.** 106

Figure A.3. **Sine vs. ReLU synthesizer.** 109

Figure A.4. **Our modulator improves expressivity.** 112

Figure A.5. **Continuity at tile boundaries.** 112

Figure A.6. **Comparison Baselines.** 115

Figure A.7. **Functional representation of CelebA images.** 117

Figure A.8. **Implicit shape encoding.** 118

Figure A.9. **SIREN struggles with high-frequencies in temporal domain.** 120

Figure A.10. **Encoding large 3D scenes.** 121

Figure A.11. **Image-based relighting using MLPs.** 123

LIST OF TABLES

Table 2.1. **Quantitative evaluation on inverse rendering of geometry.** 23

Table 4.1. **Quantitative comparisons for 2D.** 82

Table 4.2. We evaluate our method and PRDPT [41], which employs parameter-space blurring, across five synthetic scenes of varying difficulty. 83

Table 4.3. **Quantitative results with differentiable rasterization.** 88

Table A.1. **Comparison of generalization ability on CelebA.** 115

Table A.2. **Comparison of generalization ability on CIFAR-10.** 116

Table A.3. **Comparison of generalization ability on ShapeNet.** 119

Table A.4. **Comparison of generalization ability on Vimeo-90k**. 119

Table A.5. **Encode high-res images faster.** 119

Table A.6. **Implicit encoding of large 3D scenes.** 121

Table A.7. **Encoding high-res videos faster.** 121

ACKNOWLEDGEMENTS

This dissertation is the collective effort of a benevolent community, and only through some unexplainable fortune did I end up as the chosen voice to put it all together. The work has been possible only because of the innumerable people who believed in the ideal of shared common good and fought for education, science and public welfare. Thank you to all of them.

I will forever be in debt to my advisors Manmohan and Ravi. They believed in a kid equipped with nothing but ambition and invested a great part of themselves in teaching me how, and perhaps more importantly, why to pursue research. They gave me a sandbox to explore niche research ideas while insulating me from external expectations. The initial kernel of differentiable rendering of geometry was provided by them in our very first email chain, and that foresight of a potentially productive research area still surprises me to this day. More than anything, they are great examples of how to lead lives with the highest integrity.

Tzu-Mao's pioneering work on differentiable rendering was the starting point of this thesis. I am grateful that he agreed to be on my committee and since then I have had the fortune of collaborating with him on multiple projects. He is a great mentor, and dare I say a friend, who inspires me to valiantly seek ideas from within and communicate them without fear. Yusu's class on topological data analysis was the impetus for the ideas I explored in the topological derivatives work. She was very gracious in agreeing to serve on the committee and without her generosity this thesis would not have come to fruition.

I came to UCSD vastly unprepared for what lay ahead. Time and again, I met mentors and friends along the way who nudged me in the right direction, shaped my world view, pulled me up when my heart was not in it, and they did all of this with genuine kindness. Michael, my mentor since the beginning of my PhD, has been an incredible positive force in my life. His unmatched humility, wit and taste for research problems are unattainable peaks that I will spend the rest of my life climbing. I am greatly indebted to him for teaching me both the craft and the art of doing research. I lucked out with my mentors from my undergraduate studies: Parikshit, Kalpit, Rajvi and PJN, who showed me the ropes and are still a constant source of inspiration. I

was also fortunate to get internship experience during the PhD and found incredible collaborators: Connelly, Eli, Noam, Michael, Changil and Abhi. I also found mentors in my friends Yash, Zhengqin, Alex T, Nithin, Sina, Wesley, Xuanda, Kai-En, Tarun, Kunal and Mustafa who have all spent so much time educating me on varied topics that I can't really claim the ideas in the thesis to be mine. It's like the parable of the stone soup, where I only added the final spice. My friends, Matin, Zilu, Hesper, Bing, Mukund, Oliver, Mohammad, Yu-Ying, Zimo, Richie, Jeffrey, Haolin and Liwen have all contributed to the soup.

Through my friends Tara, Martha, Tanny and Mustafa, I learnt to love community as the ideal closest to my heart. Their love and laughter are deeply woven into my memories of San Diego. My roommates Udayan and Anisha have been my family away from home. Their contribution to this thesis is immeasurable, and I am grateful to have found them.

This is my first and only thesis, so I want to acknowledge the behind-the-scenes work my family did to make it possible. My parents and my brother have let me pretend for many years that science and my career are the most important things in life, and the time I spend with them shakes me out of that illusion. Lastly, my partner, Ishita, and I regrettably did not share enough time during the PhD, but I find refuge in the thought that we shared the same heart, the same dreams, and the same name.

Chapter 2, in full, is a reprint of the material as it appears in "A Level Set Theory for Neural Implicit Evolution under Explicit Flows", Ishit Mehta, Manmohan Chandraker, Ravi Ramamoorthi, European Conference on Computer Vision (ECCV), 2022. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in full, is a reprint of the material as it appears in "A Theory of Topological Derivatives for Inverse Rendering of Geometry", Ishit Mehta, Manmohan Chandraker, Ravi Ramamoorthi, International Conference on Computer Vision (ICCV), 2023. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in full, is a reprint of the material as it appears in "Locally Orderless Images for Optimization in Differentiable Rendering", Ishit Mehta, Manmohan Chandraker, Ravi

Ramamoorthi, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025. The dissertation author was the primary investigator and author of this paper.

Appendix A, in full, is a reprint of the material as it appears in "Modulated Periodic Activations for Generalizable Local Functional Representations", Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, Manmohan Chandraker, International Conference on Computer Vision (ICCV), 2021. The dissertation author was the primary investigator and author of this paper.

VITA

2018 Bachelor of Technology (Honors) and Master of Science in Computer Science, International Institute of Information Technology, Hyderabad

2026 Doctor of Philosophy in Computer Science, University of California San Diego

PUBLICATIONS

Kunal Gupta, Ishit Mehta, Kun Wang, Nicholas Chua, Abhimanyu Krishna, Yan Deng, Ravi Ramamoorthi, Manmohan Chandraker. *INTERIORAGENT: LLM Agent for Interior Design aware 3D Layout Generation*. 3DV, 2026.

Yash Belhe, Ishit Mehta, Wesley Chang, Iliyan Georgiev, Michaël Gharbi, Ravi Ramamoorthi, Tzu-Mao Li. *Automatic Sampling for Discontinuities in Differentiable Shaders*. SIGGRAPH Asia, 2025. (Best Paper)

Mustafa B. Yaldiz, Ishit Mehta, Nithin Raghavan, Andreas Meuleman, Tzu-Mao Li, Ravi Ramamoorthi. *Spectral Prefiltering of Neural Fields*. SIGGRAPH Asia, 2025.

Ishit Mehta, Manmohan Chandraker, Ravi Ramamoorthi. *Locally Orderless Images for Optimization in Differentiable Rendering*. CVPR, 2025. (Highlight)

Ishit Mehta, Manmohan Chandraker, Ravi Ramamoorthi. *A Theory of Topological Derivatives for Inverse Rendering of Geometry*. ICCV, 2023.

Ishit Mehta, Manmohan Chandraker, Ravi Ramamoorthi. *A Level Set Theory for Neural Implicit Evolution under Explicit Flows*. ECCV, 2022. (Oral, Best Paper Honorable Mention)

Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, Manmohan Chandraker. *Modulated Periodic Activations for Generalizable Local Functional Representations*. ICCV, 2021.

Anurag Ghosh, Vijay Lingam, Ishit Mehta, Akshay Uttama Nambi, Venkata N. Padmanabhan, Satish Sangameswaran. *Smartphone-based Driver License Testing: Demo Abstract*. SenSys, 2019.

Akshay Uttama Nambi, Ishit Mehta, Anurag Ghosh, Vijay Lingam, Venkata N. Padmanabhan. *ALT: Towards Automating Driver License Testing Using Smartphones*. SenSys, 2019.

Parikshit Sakurikar, Ishit Mehta, P. J. Narayanan. *Defocus Magnification using Conditional Adversarial Networks*. WACV, 2019.

Akshay Uttama Nambi, Shruthi Bannur, Ishit Mehta, Harshvardhan Kalra, Aditya Virmani, Venkata N. Padmanabhan, Ravi Bhandari, Bhaskaran Raman. *HAMS: Driver and Driving Monitoring using a Smartphone*. MobiCom, 2018.

Ishit Mehta, Parikshit Sakurikar, P. J. Narayanan. *Structured Adversarial Training for Unsupervised Monocular Depth Estimation*. 3DV, 2018.

Parikshit Sakurikar, Ishit Mehta, Vineeth N. Balasubramanian, P. J. Narayanan. *RefocusGAN: Scene Refocusing using a Single Image*. ECCV, 2018.

Pranjal Kumar Rai, Sajal Maheshwari, Ishit Mehta, Parikshit Sakurikar, Vineet Gandhi. *Beyond OCRs for Document Blur Estimation*. ICDAR, 2017.

Ishit Mehta, Parikshit Sakurikar, Rajvi Shah, P. J. Narayanan. *SynCam: Capturing Sub-Frame Synchronous Media using Smartphones*. ICME, 2017. (Oral)

FIELDS OF STUDY

Major Field: Computer Science

Studies in Computer Graphics and Computer Vision
Professors Ravi Ramamoorthi and Manmohan Chandraker

ABSTRACT OF THE DISSERTATION

Inverse Rendering of Geometry

by

Ishit Bhadresh Mehta

Doctor of Philosophy in Computer Science

University of California San Diego, 2026

Professor Manmohan Chandraker, Co-Chair
Professor Ravi Ramamoorthi, Co-Chair

Computer graphics algorithms are efficient, accurate, and predictive at forward simulation. Photo-realistic images can be generated from precise physical quantities that describe the geometry of objects in the scene, their reflectance properties, and the lighting conditions. The inverse-rendering problem is to invert these algorithms and recover the scene description from the induced images, recovering the cause from the effect. This work addresses inverse rendering of scene geometry via analysis-by-synthesis. The method starts with an initial hypothesis of the geometry, renders an image, compares it with a reference, and uses the error to update the hypothesis. There are three crucial components to this approach: (a) suitable mathematical

representations to describe the geometry, (b) mathematical tools and algorithms that estimate gradients for a rendering algorithm, and (c) objective functions to compare estimates with reference images. The design and analysis of these three components are described in this work.

(a) We propose a method that uses Lagrangian gradients from a differentiable renderer for Eulerian geometry representations. Deformation, simulation, and optimization objectives defined for Lagrangian mesh representations can be applied to implicit surface representations. The proposed level-set theory shows that existing level-set update rules violate the normal-velocity condition required for valid surface deformation, and a method that builds on the proposed theory can be used for accurate surface smoothing, curvature flows, and inverse rendering.

(b) We develop a theoretical framework for discrete topology changes under smooth rendering objectives. Topological derivatives are derived for the variational optimization of image functionals. Prior methods rely on silhouette gradients for topology changes, but such signals are sparse. The proposed theory uses dense topological derivatives that relate the nucleation of holes and phase changes to changes in image intensity.

(c) We introduce a new optimization objective based on the locally-orderless image, in which each pixel maps to a histogram of intensities that preserves local variations in appearance but discards spatial information. The resulting formulation leads to faster convergence to better local minima for inverse-graphics problems that optimize scene parameters whose updates induce motion in image space, where gradients for standard optimization objectives are sparse and lead to poor convergence.

1 | Introduction

Computer graphics has, for most of its history, been devoted to the forward problem of rendering, the synthesis of images from a physical description of a scene. The inverse problem, the recovery of that description from its images, is fundamentally harder. It has been studied under several names, among them inverse rendering, inverse graphics, and computer vision, and each confronts the same question: given a photograph, what can be inferred about the scene that produced it?

The focus of this dissertation is to design mathematical tools and algorithms that recover the geometry of the objects in a scene from a set of images. The history of this problem is rooted in cognitive science, as shape perception, and in computer vision, as shape from shading; more recently, the graphics community, including our own work, has approached it as an inverse-rendering-of-geometry problem. The overall approach is to pose the inverse problem as an analysis-by-synthesis optimization, and to differentiate through the forward-rendering function to compute gradients for recovery. As with any gradient-based optimization, success depends on the parameterization of the optimization variables, the choice of objective function, and the quality of its gradients.

In the current research climate, where methods often require objectionable amounts of data, compute, and capital, there is an acute need for methods that are efficient along several axes and that illuminate the underlying structure of the problem and improve our understanding. This dissertation, I hope, is a step in that direction.

1.1 Motivation

Modern artificial-intelligence systems are good at simulation. They produce photorealistic images and video, plausible motion, and high-fidelity audio and text. What they rarely provide is an account of how they arrived at any particular output. For systems that operate where the cost of an error is high, such as healthcare, autonomous driving, manufacturing, and the justice system, this is a real limitation. We would prefer models that reconstruct the causes behind an observation and let us inspect them, over models whose predictions and decisions stay hidden from view.

Reasoning of this kind is an inverse problem, and physics already supplies the forward models it needs. The light-transport equation and the Navier-Stokes equations are causal and interpretable: given the description of a scene, they tell us what we should observe. A generative model that has only learned to paint pixels can show water sloshing inside a container, but it cannot easily say how the water will move next, because that depends on the geometry of the container, the material of the fluid, and the forces acting on it. Running the physics backward recovers these quantities. By differentiating through the simulation, we can recover the latent causes of an image and then generate new images that obey the same laws.

This dissertation studies one part of that program, the recovery of geometry, and treats it as a foundation on which the rest can be built.

1.2 Contributions

A level-set theory for implicit geometry (Chapter 2).

Our first contribution is about the representation. Coordinate-based neural networks parameterizing implicit surfaces have emerged as efficient representations of geometry. They act as parametric level sets, with the zero-level set defining the surface of interest. We present a theoretical framework that applies deformation operations defined for triangle meshes onto such implicit surfaces. Several of these operations can be viewed as energy-minimization problems

that induce an instantaneous flow field on the explicit surface. Our method uses the flow field to deform parametric implicit surfaces by extending the classical theory of level sets. We also derive a consolidated view for existing methods on differentiable surface extraction and rendering, by formalizing connections to the level-set theory. We show that these methods drift from the theory and that our approach improves substantially on them for applications like surface smoothing, mean-curvature flow, inverse rendering and user-defined editing on implicit geometry.

Topological derivatives for inverse rendering (Chapter 3).

Our second contribution is about the gradients. We introduce a theoretical framework for differentiable surface evolution that allows discrete topology changes through the use of topological derivatives for variational optimization of image functionals. While prior methods for inverse rendering of geometry rely on silhouette gradients for topology changes, such signals are sparse. In contrast, our theory derives topological derivatives that relate the introduction of vanishing holes and phases to changes in image intensity. As a result, we enable differentiable shape perturbations in the form of hole or phase nucleation. We validate the proposed theory with optimization of closed curves in 2D and surfaces in 3D to lend insights into limitations of current methods and enable improved applications such as image vectorization, vector-graphics generation from text prompts, single-image reconstruction of shape ambigrams and multiview 3D reconstruction.

Locally orderless images for optimization (Chapter 4).

Our third contribution is about the objective. Problems in differentiable rendering often involve optimizing scene parameters that cause motion in image space. The gradients for such parameters tend to be sparse, leading to poor convergence. While existing methods address this sparsity through proxy gradients such as topological derivatives or Lagrangian derivatives, they make simplifying assumptions about rendering. Multi-resolution image pyramids offer an alternative approach but prove unreliable in practice. We introduce a method that uses locally orderless images, where each pixel maps to a histogram of intensities that preserves

local variations in appearance. Using an inverse rendering objective that minimizes histogram distance, our method extends support for sparsely defined image gradients and recovers optimal parameters. We validate our method on various inverse problems using both synthetic and real data.

1.3 Dissertation Organization

Chapters 2, 3, and 4 develop the three contributions above, and Chapter 5 concludes. Appendix A includes earlier work that introduces a neural representation for geometry which is a useful building block for the inverse rendering studied here. There, we present a representation that generalizes to multiple instances and achieves state-of-the-art fidelity, using a dual-MLP architecture: a synthesis network maps a low-dimensional input to the output domain, and a modulation network maps a latent code to parameters that modulate the periodic activations of the synthesis network. We also propose a local-functional representation that partitions the signal's domain into a regular grid, with each tile represented by a latent code, which enables generalization across images, videos and shapes.

1.4 Publications

The technical chapters of this dissertation are based on the following publications:

- Chapter 2: *A Level Set Theory for Neural Implicit Evolution under Explicit Flows*. Ishit Mehta, Manmohan Chandraker, Ravi Ramamoorthi. ECCV 2022.
- Chapter 3: *A Theory of Topological Derivatives for Inverse Rendering of Geometry*. Ishit Mehta, Manmohan Chandraker, Ravi Ramamoorthi. ICCV 2023.
- Chapter 4: *Locally Orderless Images for Optimization in Differentiable Rendering*. Ishit Mehta, Manmohan Chandraker, Ravi Ramamoorthi. CVPR 2025.

- Appendix A: *Modulated Periodic Activations for Generalizable Local Functional Representations*. Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, Manmohan Chandraker. ICCV 2021.

2 | A Level Set Theory for Neural Implicit Evolution under Explicit Flows

The introduction framed inverse rendering of geometry as an analysis-by-synthesis problem that turns on three choices: how the geometry is represented, how gradients are obtained from the renderer, and how images are compared. This chapter takes up the first of these. It develops a representation that carries the deformation machinery of explicit meshes over to neural implicit surfaces, so that a surface stored in the weights of a network can be moved as if it were a mesh. That capability is what the gradient and objective questions of Chapters 3 and 4 later build on.

2.1 Introduction

Recent successes in generative modeling of shapes [30, 123, 142] and inverse rendering [113, 185] are largely driven by implicit representations of geometry parameterized as multi-layer perceptrons (MLPs) (or neural implicits [35]). These networks can compactly represent highly-detailed surfaces at infinite resolution [96, 143, 157, 158]; they are defined continuously in $\mathbb{R}^3$ and are differentiable—enabling their usage in gradient-based optimization [68, 185] and learning [4, 27, 107] methods. Despite these advances, there is still a large body of work in geometry processing, computer vision and graphics which relies on explicit surface representations. Often these mesh-based algorithms are a better choice than their implicit counterparts. For instance, in case of inverse rendering, differentiable renderers for triangle meshes [46, 74, 79, 115, 188] are a) faster, b) more accurate, and c) can handle more complex light-transport effects, in comparison to the renderers designed for implicit

surfaces [63, 113, 185]. Similarly, geometry processing algorithms for applications like surface smoothing and deformation [21, 150, 160] are vastly superior in terms of compute and memory requirements than the ones developed for neural implicit surfaces [183]. Most of these methods, however, are highly specific to mesh geometry and are not easily adaptable to MLP-defined surfaces. Our work is a theoretical attempt to bridge this gap.

We first introduce the following insight: several mesh-based algorithms define an energy-minimization problem that is solved using gradient descent; the gradients used by the optimizer to update the geometry can be viewed as analogous to an instantaneous explicit flow-field ($\mathbf{V}$) applied on the surface. Informed by the literature on fluid simulation [15] and level-sets [120], deformation of a surface with such a flow field depends on the geometry representation.

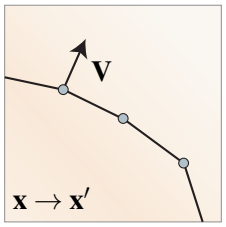


Figure 2.1. Lagrangian representation: surface points $\mathbf{x}$ are advected by the flow field $\mathbf{V}$.

The *Lagrangian* representation involves tracking the surface explicitly as a set of a points ($\mathbf{x}$) and connections (like triangles). The point-set is discrete and the connectivity is *static*, which keeps the optimization relatively simple; we can separately integrate the field at each point (update vertices $\mathbf{x} \rightarrow \mathbf{x}'$). But optimization of the resolution of the surface is non-trivial and can also get unwieldy for problems which involve surfaces with unknown topology.

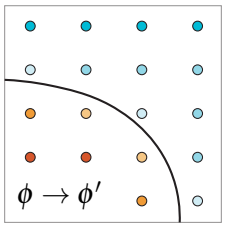


Figure 2.2. Eulerian representation: each point stores an implicit value ϕ, evolved as $\phi \rightarrow \phi'$.

Alternatively, *Eulerian* descriptions can be used. Each point in space has an object-

property ϕ associated with it, like the distance from the surface or its occupancy inside the enclosed volume. The surface here is implicitly defined with *dynamic* connectivity; one can smoothly vary the topology during optimization. Canonically, ϕ is defined only on a discrete voxel-grid and needs to be interpolated for points off the grid. Here, making instantaneous updates to the surface is more involved as it requires changing ϕ values for a large set of points. A neural implicit is a continuous variant of an Eulerian representation. Applying a flow field to such functions is non-trivial as updates are required in the parameter (θ) space as opposed to directly updating ϕ to ϕ'.

We propose a parametric level-set evolution method (§ 2.4) which propagates neural implicit surfaces according to an explicitly generated flow field. Our method comprises of three repeating steps, 1) A non-differentiable surface extraction method like Marching Cubes [90] or Sphere Tracing [56] is used to obtain a Lagrangian representation corresponding to a neural implicit, 2) A mesh-based algorithm is used to derive a flow field on the explicit surface (§ 2.4.1), and 3) A corresponding Eulerian flow field is used to evolve the implicit geometry (§ 2.4.2).

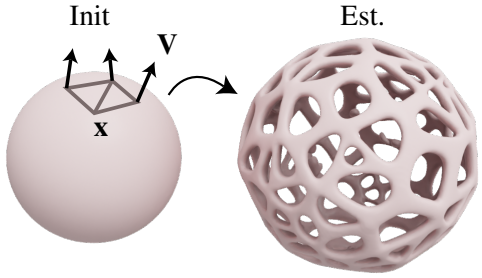


Figure 2.3. Inverse-rendering recovery. An example of geometry recovered from images using our method.

Previous methods [113, 135] approach the problem of using mesh-based energy functions on Eulerian surfaces with the idea of differentiable extraction of Lagrangian representations. We show these methods are not in accordance with the level-set theory [121] (§ 2.5). The discussion also yields a more general proof (§ 2.5.1) for differentiable surface extraction from level set functions with arbitrary gradient norms ($|\nabla\phi| \neq 1$). Our method is more formally connected to the theory and we validate it with experimental observations made on three diverse problem

settings, 1) Curvature-based deformation (§ 2.6.1), where we demonstrate more accurate surface smoothing and mean-curvature flow than previous methods [135, 183], 2) Inverse rendering of geometry (§ 2.6.2), where we show accurate recovery from multi-view images for high-genus shapes without object masks as in [113, 185] (Example in Fig. 1), and 3) User-defined shape editing (§ 2.6.3), where the implicit surface is deformed $60\times$ faster than previous work [183] which deforms only the explicit surface.

2.2 Related Work

Coordinate-based MLPs have become a popular choice as function approximators for signals like images, videos and shapes [30, 123, 143, 153, 158]. Our work focuses on using MLPs for approximating implicit geometry. Such representations are compact, continuous and differentiable. These advantages are well suited with gradient-descent based optimization and learning problems. Recent developments in generative modeling of shapes [26, 99, 123, 142], 3D consistent image synthesis [27, 181], 3D reconstruction [11, 154, 193] and inverse rendering [68, 117, 184, 185], all rely on representing geometry using MLPs. For a more detailed discussion on recent work regarding coordinate-based representations refer to the survey by Xie *et al.* [179], and for inverse rendering the survey by Tewari *et al.* [162].

There is also a rich literature on geometry processing [19, 21, 20, 151, 150] and inverse rendering [10, 46, 93, 112] for explicit surface representations. Yang *et al.* [183] introduce some of the geometry processing ideas to neural implicit surfaces, but the method could be inaccurate (§ 2.6.1) and slow (§ 2.6.3).

For inverse rendering applications, differentiable renderers for triangle meshes are used for gradient-based optimization. Physics-based renderers differentiate through a light-simulation process [13, 79, 115, 188] and provide accurate and unbiased gradients. Alternatively, differentiable rasterization [74, 85, 134] can be used for high-performance gradient computation, but only for single-bounce rendering models. However, optimizing triangle meshes with

gradient-descent is non-trivial. Careful design of the optimization method [46, 112] and error functions [93] is required for robust optimization.

To circumvent some of these issues, IDR [185] and DVR [113] use implicit surface representations like SDFs and occupancy functions [100] parameterized with MLPs. These methods mitigate some of the topological restrictions, but are not physics-based and are not in accordance with the level-set theory (§ 2.5.2). We propose an inverse rendering method which uses explicit differentiable renderers for parametrically defined implicit surfaces. The proposed method is not as sensitive to initialization as explicit methods [46, 112] are, does not require an object mask like implicit methods [113, 185] do, and maintains the ability to vary topology during optimization.

Our method uses the level-set theory [121] as the foundation for optimizing and deforming parametric implicit surfaces. Previous methods [6, 108, 174] for the applications discussed in this work apply only to non-parametric level sets. Perhaps the most related works to our method are MeshSDF [135] and RtS [33]. Compared to MeshSDF, our approach is more formally connected to the theory of level-sets (§ 2.4), applies to all parametric functions (unlike only SDFs for MeshSDF) (§ 2.5.1), and works for a more diverse set of optimization problems like shape editing (§ 2.6.1, 2.6.3) and inverse rendering (§ 2.6.2)—unlike experimental observations made by Remelli *et al.* [135] only on learning-based settings. Compared to RtS [33], we also show geometry processing applications along with theoretical parallels (§ 2.5) between parametric level-set methods like MeshSDF [135], DVR [113], IDR [185] and the classical theory [120]. Our inverse rendering method is also shown to work (§ 2.6.2) for a set of high-genus shapes with a genus-0 initialization, as opposed to only object-pose optimization or simplistic genus changes shown in [33].

2.3 Background

Consider a closed surface of arbitrary topology $\partial\Omega$ evolving with respect to time t. We define a Lagrangian representation of this surface with a finite set of k points in $\mathbb{R}^3$ as $\partial\Omega_L = \{\mathbf{x}_i \mid \mathbf{x}_i \sim \partial\Omega;\ \forall i \in \{1,2,3,\ldots,k\}\}$. This point-set can be viewed as a triangle mesh if an additional set of connections between the points is provided, and as a point cloud otherwise. Implicitly this surface can also be represented with a family of level-sets $\phi : \mathbb{R}^3 \mapsto \mathbb{R}$, the zero iso-contour of which represents the surface $\partial\Omega_E = \{\mathbf{x} \mid \phi(\mathbf{x}) = 0\}$. ϕ can be arbitrarily chosen, but a canonical choice is a signed-distance function (SDF) which satisfies:

$$\phi(\mathbf{x}) = \min_{\mathbf{x}_C \in \partial\Omega} \{||\mathbf{x} - \mathbf{x}_C||_2\}\ \text{sign}(\mathbf{x} - \mathbf{x}_C), \tag{2.1}$$

where $\mathbf{x}_C$ is the closest point on the surface to $\mathbf{x}$ and the sign of $\phi(\mathbf{x})$ denotes whether $\mathbf{x}$ is enclosed $(-)$ within the shape or not $(+)$.

Parameterizing ϕ

Analytically defining ϕ for simple and regular shapes is relatively straightforward [130], but is infeasible for most objects. Recent work on 3D reconstruction [113, 185] and generative shape modeling [30, 123] suggests parameterizing ϕ using a multi-layer perceptron (MLP) with θ as its parameters. The networks are optimized by minimizing an energy function comprised of a distance term [123] and a gradient term [49] enforcing $|\nabla\phi|$ to be 1. We use SIREN [143] as the parametric function of choice, although our method is agnostic to the network parameterization. The network acts approximately as an SDF at the rest state ($t = 0$), but may not retain the SDF property (Eq. 2.1) as the surface evolves. We denote this surface as $\partial\Omega_E(\theta) = \{\mathbf{x} \mid \Phi(\mathbf{x};\theta) = 0\}$, where Φ is parameterized with θ as the weights and biases of the network. For clarity, we use Φ for parametric level-sets and ϕ for non-parametric.

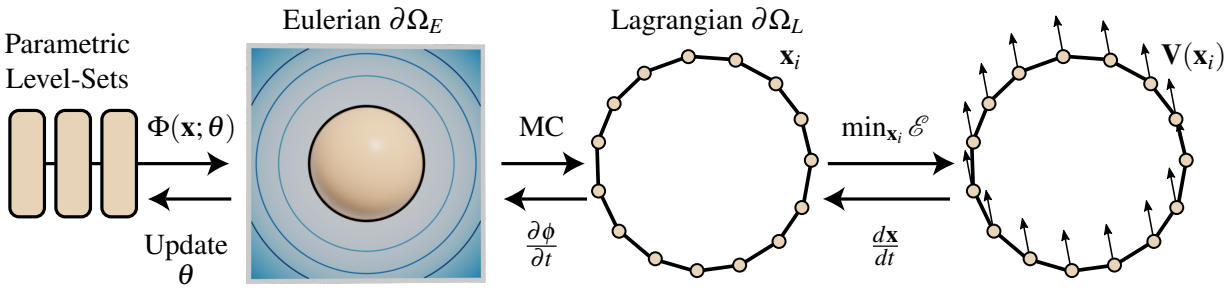


Figure 2.4. Method Overview. We present a level-set method to evolve neural representations of implicit surfaces. Using Marching Cubes (MC) [90], a Lagrangian surface $\partial\Omega_L$ is extracted from an Eulerian representation $\partial\Omega_E$ encoded in the network parameters θ. An energy function $\mathscr{E}$ is defined on $\partial\Omega_L$ which is minimized using gradient-descent. The gradients of the optimizer together act as a flow-field $\mathbf{V}$ on the surface points $\mathbf{x}$, which is used to evolve the non-parametric ϕ using the level-set equation. The values of ϕ on the surface act as references to update the parameters θ of the network.

2.4 Method

We begin the discussion by first characterizing the deformation of surfaces into Lagrangian (§ 2.4.1) and Eulerian (§ 2.4.2) settings. We show that gradient descent on energy functions defined for triangle meshes can be viewed as surface deformation under the dynamics of a flow field $\mathbf{V}$, which is *discretely* defined only on the surface points $\partial\Omega_L$. We can use this flow field to deform a *continuous* surface representation using the level-set equation. We extend the level-set equation to the case of parametric level-sets Φ which enables us to use loss functions defined on triangle meshes to deform MLP-defined level-sets.

2.4.1 Lagrangian Deformation

As mentioned earlier, in the Lagrangian setting, the surface is defined with a finite set of points $\partial\Omega_L$. A variety of methods in geometry processing and computer vision define an energy function $\mathscr{E}$ that is minimized to make instantaneous updates to $\partial\Omega_L$. Some recent examples include optimizing point-clouds for view-synthesis [3, 138] and triangle-meshes for inverse-rendering of geometry [46, 112]. The surface is updated using spatial gradients $\frac{\partial\mathscr{E}}{\partial\mathbf{x}}$, which is well studied in numerical analysis [156] and optimization methods [32]. Through the lens of

physics, these gradients induce an instantaneous flow field $\mathbf{V}(\mathbf{x})$, which can be used to evolve the surface by integrating the following ordinary differential equation (ODE):

$$\frac{d\mathbf{x}}{dt} = -\frac{\partial \mathscr{E}}{\partial \mathbf{x}} \rightarrow \mathbf{V}(\mathbf{x}). \qquad \triangleleft \text{ Lagrangian Deformation} \tag{2.2}$$

Numerically, this can be done using forward-Euler steps $\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \mathbf{V}^t(\mathbf{x})$. This is easy to accomplish if the connectivity of the points remains static. More sophisticated integration schemes can also be used [156]. Here, in case of optimization problems solved using gradient descent, Δt is equivalent to the learning rate. Several works in shape deformation [52] and inverse rendering [112] can be subsumed by this ODE with different definitions for flow $\mathbf{V}$ and time-step Δt. We next show that these readily available energy functions and optimization algorithms defined for explicit surfaces can also be used to optimize MLP-defined level-sets.

2.4.2 Eulerian Deformation

To avoid the topological complications associated with Lagrangian deformations, we can instead define a corresponding Eulerian deformation field. By definition, we know for points $\mathbf{x} \in \partial\Omega$, $\phi(\mathbf{x}) = 0$. Using implicit differentiation:

$$\begin{aligned} &\frac{d\phi(\mathbf{x})}{dt} = \frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial t} = \frac{\partial \phi}{\partial t} + \nabla\phi \cdot \mathbf{V} = 0 \qquad \triangleleft \text{ From 2.2} \\ \iff &\frac{\partial \phi}{\partial t} = -\nabla\phi \cdot \mathbf{V}. \qquad \triangleleft \text{ Eulerian Deformation} \end{aligned} \tag{2.3}$$

This partial differential equation (PDE) is sometimes referred to as the level-set equation [121], the material derivative [15] or the G-equation [95]. We extend this PDE to obtain an evolution method for parametric level-sets Φ. First, for each time step t, we extract a Lagrangian surface representation $\partial\Omega_L^t$ from Φ using MC [90]. Depending on the task at hand, an energy function $\mathscr{E}$ (*e.g.*, photometric error for inverse rendering) is defined on $\partial\Omega_L^t$. Assuming $\mathscr{E}$ is differentiable, we compute $\frac{\partial \mathscr{E}}{\partial \mathbf{x}} = -\mathbf{V}^t(\mathbf{x})$ for each vertex $\mathbf{x}_i \in \partial\Omega_L$. With the flow field $\mathbf{V}^t$, we update the

level-set function as we would in the non-parametric case using forward-Euler steps:

$$\phi^{t+1} = \Phi^t - \Delta t \nabla \Phi^t \cdot \mathbf{V}^t. \qquad \triangleleft \text{ From (2.3)} \tag{2.4}$$

The time step Δt here is a parameter which is dependent on the dynamics of the flow-field. If $\mathbf{V}$ is highly non-linear, taking smaller steps (*i.e.*, Δt is small) is required, while for a simple field, larger values of Δt should suffice. For each time step, we take the values of non-parametric ϕ^{t+1} as the reference and update the parameters of Φ accordingly. This is achieved by minimizing the following objective:

$$\min_{\theta} J(\theta) = \frac{1}{|\partial\Omega_L|} \sum_{\mathbf{x} \in \partial\Omega_L} ||\phi^{t+1}(\mathbf{x}) - \Phi(\mathbf{x};\theta)||^2 \,, \tag{2.5}$$

using gradient descent. Since the surface updates are small for each time step, the number of descent steps required is in the order of 10^2. This makes the method convenient for obtaining neural representations of deformed variants of the initial geometry. After each optimization routine, we again extract the Lagrangian surface using MC [90], which is subsequently fed into a mesh-based energy-minimization problem. An overview of our method is shown in Figure 2.4. For each of the applications we show in § 2.6, we define $\mathbf{V}^t$ using a corresponding energy function, and minimize J for each time step.

2.5 Theoretical Comparisons

The level-set method discussed in § 2.4.2 subsumes two related works, 1) Differentiable iso-surface extraction (MeshSDF) [135], and 2) Differentiable rendering of implicit surfaces (DVR/IDR) [113, 185]. We first show that MeshSDF minimizes the level-set objective J defined in (2.5) with a single gradient-descent step. But the surface does not propagate in agreement with the level-set equation, as outlined in § 2.5.1. We then show that these two seemingly disparate works (MeshSDF and DVR) are closely related in § 2.5.2. We end the discussion with

an explanation for how DVR deviates from the level-set equation.

2.5.1 Differentiable Iso-Surface Extraction

Result 2.1. *Differentiable Iso-Surface Extraction [135] takes a single gradient-descent step to minimize the level-set objective function J (Equation 2.5).*

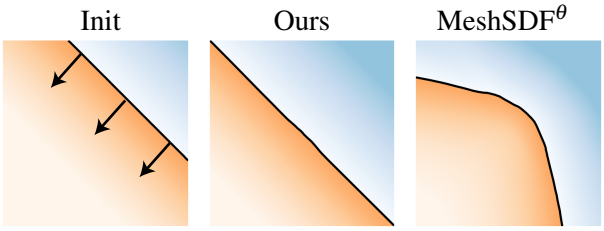


Figure 2.5. MeshSDF$^\theta$ does not follow the level-set equation. (*Left*) A planar surface defined implicitly with an MLP is influenced by a flow field in the direction of its normal. The motion attained using differentiable iso-surface extraction (*Right*) is inconsistent with the field. Our method (*Center*) propagates the front as expected.

Proof. MeshSDF [135] defines a loss-function $\mathscr{L}$ on a triangle mesh extracted using Marching Cubes [90] from an SDF parameterized with an MLP. They use an MLP $\Phi(\mathbf{x};\theta,\mathbf{z})$ conditioned on a latent-code $\mathbf{z}$ characterizing the shape. Using $\mathscr{L}$ they update the latent-code $\mathbf{z}$, which is different from our goal of updating θ for an unconditional Φ. To clarify this distinction, we use MeshSDF$^\theta$ to denote our variant which updates θ. To update the parameters of the MLP, we compute the following gradient using the chain-rule:

$$\frac{\partial \mathscr{L}}{\partial \theta} = \sum_{\mathbf{x}\in\partial\Omega_L} \frac{\partial \mathscr{L}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \Phi} \frac{\partial \Phi}{\partial \theta}, \tag{2.6}$$

where $\mathbf{x}$ are the vertices on the mesh and Φ is an SDF. The first and the third gradient terms on the right are computed using automatic differentiation. The second term $\frac{\partial \mathbf{x}}{\partial \Phi}$ can be approximated as the inverted surface normal $-\mathbf{n}(\mathbf{x}) = -\nabla_{\mathbf{x}}\Phi$ [135], when Φ is an SDF. In the spirit of Lagrangian deformation (Equation 2.2), $-\frac{\partial \mathscr{L}}{\partial \mathbf{x}}$ acts as an instantaneous flow field $\mathbf{V}$ on the vertices $\mathbf{x}$. The

parameters of the MLP are then updated as:

$$\theta \leftarrow \theta - \lambda \frac{\partial \mathscr{L}}{\partial \theta} = \theta - \lambda \sum_{\mathbf{x} \in \partial \Omega_L} \mathbf{V} \cdot \nabla \Phi \frac{\partial \Phi}{\partial \theta}, \tag{2.7}$$

where λ is the learning rate. Alternatively, we can also update θ using the objective function defined in Equation 2.5 using gradient descent:

$$\begin{aligned} \theta \leftarrow \theta - \lambda \frac{\partial J}{\partial \theta} &= \theta - \lambda \sum_{\mathbf{x} \in \partial \Omega_L} 2(\phi^{t+1}(\mathbf{x}) - \Phi(\mathbf{x}; \theta)) \left(-\frac{\partial \Phi}{\partial \theta} \right) \\ &= \theta - \varepsilon \sum_{\mathbf{x} \in \partial \Omega_L} \mathbf{V} \cdot \nabla \Phi \frac{\partial \Phi}{\partial \theta}, \quad \triangleleft \text{ From (2.4)} \end{aligned} \tag{2.8}$$

where ε is a constant. The last equivalency is valid when $\phi^{t+1}(\mathbf{x}) - \Phi(\mathbf{x}; \theta) = -\Delta t \mathbf{V} \cdot \nabla \Phi$ (Eq. 2.4), which is true only for the first gradient descent step. Subsequently, the value of $\Phi(\mathbf{x}; \theta)$ changes as the parameters get updated. Comparing (2.7) and (2.8), we conclude that the optimization in MeshSDF$^{\theta}$ has the effect of taking a single gradient-descent step to minimize J. Note that while the proof by Remelli *et al.* [135] assumes Φ is an SDF, the second update equation (2.8) does not. It is valid for all level-set functions, with no restrictions on the values of the gradient-norm ($|\nabla \Phi|$) and is also valid for occupancy functions [100]. □

However, by taking a single step to update θ, the surface does not propagate in agreement with the level-set equation (2.3). This is problematic for applications which require the surface to move as intended by the flow-field. An example application is shown in § 2.6.1. We also illustrate this with a toy example in Figure 2.5 where a planar surface propagates in the direction of its normal. A more formal discussion is in Result 2.2.

Result 2.2. *Dfferentiable iso-surface extraction [135] does not propagate the surface-front as dictated by the flow field.*

Proof. We show this with an example flow field. Consider the surface-front propagating with a constant speed β in the direction of the normal. The corresponding Eulerian deformation is

constant across the surface:

$$\frac{\partial \phi}{\partial t} = -|\nabla \phi|\beta = -\beta. \qquad \triangleleft \text{ Assuming } \phi \text{ is an SDF} \tag{2.9}$$

With the same flow field, we can estimate the instantaneous change in Φ (parametric) for MeshSDF$^{\theta}$ as:

$$\frac{\partial \Phi}{\partial t} = \frac{\partial \Phi}{\partial \theta}\frac{\partial \theta}{\partial t} = \frac{\partial \Phi}{\partial \theta}\sum_{\mathbf{x}\in\partial\Omega_L} -\beta\frac{\partial \Phi}{\partial \theta} = \frac{\partial \Phi}{\partial \theta}B. \quad \triangleleft \text{ From (2.7) and (2.9)} \tag{2.10}$$

The term B on the right is constant for every point $\mathbf{x}$ on the surface. The gradient $\frac{\partial \Phi}{\partial \theta}$ is dependent on the position where it is evaluated and hence front-propagation $\frac{\partial \Phi}{\partial t}$ is not constant. On the contrary, we minimize the objective function defined in (2.5) which ensures that the surface propagation is constant as in (2.9). □ □

For the implications of Result 2.1 and 2.2, and experimental comparisons, we defer the discussion to § 2.6.1.

2.5.2 Differentiable Surface Rendering

An alternate way of extracting a Lagrangian surface $\partial\Omega_L$ from Φ is by computing ray-surface intersections using ray-marching or sphere-tracing [56]. If ray-marching is differentiable [64, 83], one can backpropagate gradients from error functions defined on $\partial\Omega_L$ to the parameters θ defining the implicit surface. Recent developments in inverse rendering [100, 185] rely on this idea. The explicit surface extracted using ray-marching differs from the one obtained using marching-cubes in two ways, 1) Ray-marching does not extract the connectivity (*e.g.*, triangle faces) among the intersection points. This restricts the usage of loss functions which rely on attributes like the edge-length or differential operators like the Laplacian. 2) The intersection points depend on the camera attributes. The resolution of the image plane affects the density of points and the viewing direction determines the visibility. As a

result, $\partial\Omega_L$ obtained using ray-marching could be sparser than the one obtained marching cubes. Furthermore, as we show in Result 2.3, by using differentiable ray-marching the parameters get updated exactly as when differentiable iso-surface extraction is used—although with a less favorable Lagrangian representation (sparser and no connectivity). This is in addition to the computational disadvantage associated with ray-marching. We also formally show in Result 2.4 that surface evolution with differentiable ray-marching is in disagreement with level-set theory for tangential flows.

Result 2.3. *Surface evolution using differentiable ray-marching of parametric implicit surfaces [100, 185] is the same as using differentiable iso-surface extraction [135] when the viewing direction* $\mathbf{v}_u$ *is parallel to the normal* $\mathbf{n}$ *at the intersection point* $\mathbf{x}_u$*. The parameters* θ *for the level-set function* Φ *are updated as:*

$$\theta \leftarrow \theta - \lambda \sum_{\mathbf{x}_u} \mathbf{V} \cdot \nabla\Phi \frac{\partial \Phi}{\partial \theta}, \tag{2.11}$$

where $\mathbf{x}_u$ are the visible points and $\mathbf{V}$ is the flow field. Comparing (2.11) and (2.7), the gradient-descent step is the same. As in the case of MeshSDF$^\theta$, here the surface is evolved with a single step in the parameter space. We provide a more detailed proof for (2.11) in Section 2.7.1.

Result 2.4. *Differentiable ray-marching of parametric implicit surfaces [100, 185] disagrees with the level-set equation for tangential components* $\mathbf{V}^\perp$ *of the flow field* $\mathbf{V}$*. The change in parameters* $\Delta\theta$ *is:*

$$\Delta\theta = \lambda \sum_{\mathbf{x}_u} \pm |\mathbf{V}^\perp| \tan(\arccos(\nabla\Phi \cdot \mathbf{v}_u)) \frac{\partial \Phi}{\partial \theta} \neq 0, \tag{2.12}$$

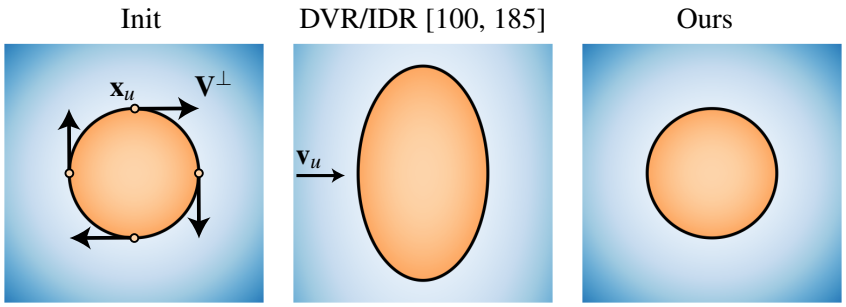


Figure 2.6. Tangential flow fields may deform surfaces when DVR/IDR is used for surface extraction. (*Left*) A parametric Eulerian circle undergoes tangential deformation $\mathbf{V}^\perp$ at surface points $\mathbf{x}_u$. (*Middle*) Using differentiable surface rendering, the surface deforms incorrectly. (*Right*) Our method agrees with the level-set equation and the resultant deformation is the identity.

which could be non-zero. A detailed proof is in Section 2.7.1. Referring to Eq. 2.3, for tangential flows the surface should not undergo any deformation, i.e. $\frac{\partial \phi}{\partial t} = 0$. However, since θ gets updated as in (2.12), the surface *does* deform. We instead minimize the objective function J (Eq. 2.5) which is 0 for a tangential field. We show an example of tangential deformation in Figure 2.6. Experimental comparisons in § 2.6.2 validate that deforming Φ accurately is critical for applications like inverse rendering.

2.6 Applications

We focus on validating the proposed theory with computer graphics models in three different settings, 1) Curvature-based deformations (§ 2.6.1), which can be used to smooth/sharpen features and apply curvature defined flows on implicit surfaces, 2) Inverse rendering of geometry (§ 2.6.2), where a differentiable renderer for explicit geometry can be used to evolve implicit surfaces, and 3) User-defined deformations (§ 2.6.3), where a user can specify alterations for a given object.

2.6.1 Curvature-based Deformation

To apply surface smoothing on parametric implicit surfaces, we first define a corresponding explicit force field. For the extracted Lagrangian surface $\partial\Omega_L$ at each time

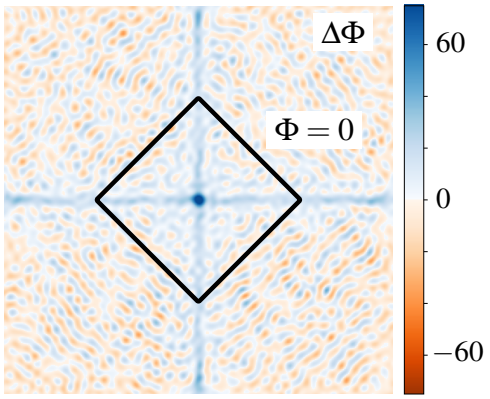


Figure 2.7. The laplacian $\Delta\Phi$ of an MLP-defined level-set function is noisy. We show the mean-curvature values for a parametric level-set function of a square. Large values are observed for a zero-curvature surface.

step, we minimize the Dirichlet-energy functional on the surface. In the continuous setting, this is defined as:

$$\mathscr{E}(\partial\Omega) = \int_{\partial\Omega} ||\nabla \mathbf{x}||^2 \, d\mathbf{x}. \tag{2.13}$$

Minimizing $\mathscr{E}$ can be shown [19] to induce the following explicit flow-field on the surface:

$$\frac{\partial \mathbf{x}}{\partial t} = \mathbf{V}(\mathbf{x}) = \lambda \Delta \mathbf{x} = -2\lambda\, \kappa(\mathbf{x})\mathbf{n}(\mathbf{x}), \tag{2.14}$$

where λ is a scalar diffusion coefficient, Δ is the Laplace-Beltrami operator and $\mathbf{n}$ is the normal. κ is the mean-curvature, which for an implicit surface is defined as the divergence of the normalized gradient of ϕ (*i.e.*, $\nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}$) [120]. It is equivalent to computing the laplacian $\Delta\Phi$ of the MLP using automatic differentiation. In Figure 2.7 we show that such an estimation of the laplacian is noisy—significant magnitudes are observed even for surfaces with zero curvature.[1] Instead of using (2.14) as it is, which requires estimating $\Delta\Phi$, we approximate $\mathbf{V} = \lambda\Delta\mathbf{x} \approx \lambda\mathbf{L}\mathbf{x}$; where $\mathbf{L}$ is the discrete Laplacian we can compute using the Lagrangian surface. Note that this is feasible only because of the hybrid (Eulerian+Lagrangian) nature of our method. We use

[1] This might be due to the unconstrained Lipschitz constants of MLPs [82].

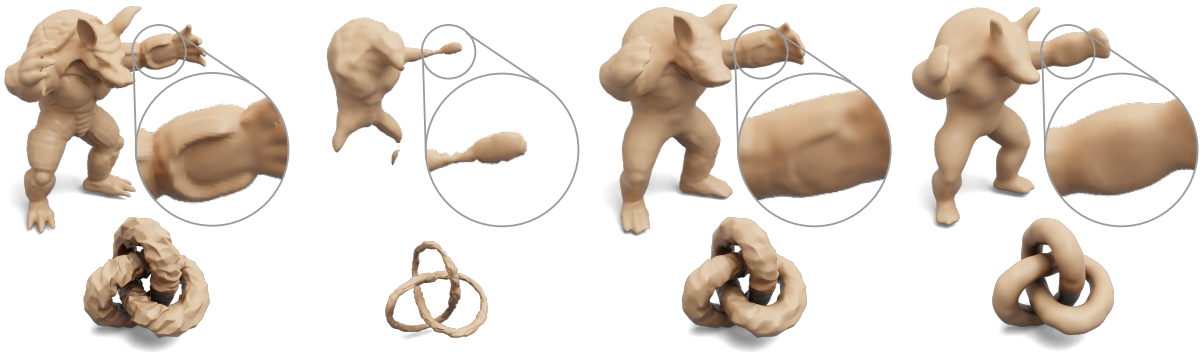


Figure 2.8. Surface smoothing on parametric level-sets. We apply surface smoothing on an MLP-defined implicit surface by minimizing Dirichlet energy on the corresponding explicit surface. We use a discrete Laplacian to define a flow-field on the surface; NFGP [183] uses its continuous counterpart and preserves too many high-frequency details. MeshSDF$^\theta$ [135] fails to smoothen the surface.

the flow-field to update Φ using the method outlined in § 2.4.2. Figure 2.8 shows qualitative comparisons for smoothing applied on two surfaces. We show a comparison with a method which applies deformation using the continuous Laplacian ($\Delta\Phi$) (NFGP) [183]. When $\mathscr{E}$ is minimized using MeshSDF$^\theta$ [135], the deformation is not curvature based and high-frequency features are retained during the evolution. More results and implementation details are in Section 2.7.2.

Equation (2.14) is referred to as mean-curvature flow [39]. Since our method deforms the surface in accordance with the flow-field, we can use (2.14) to apply mean-curvature flow

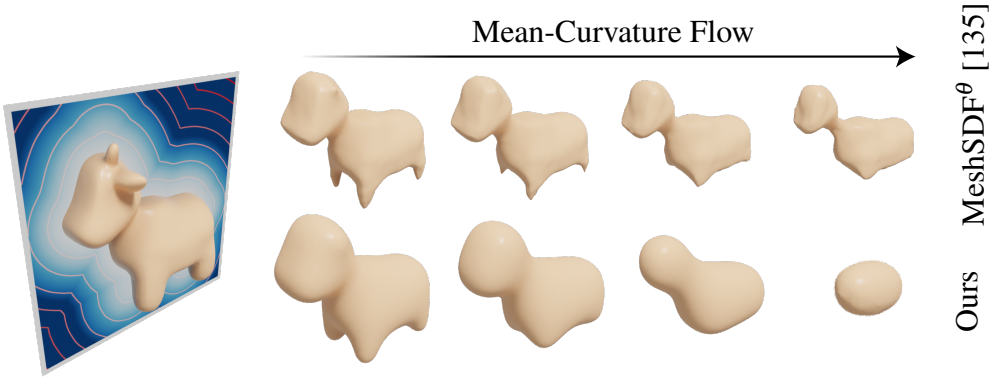


Figure 2.9. We use an explicit mean-curvature flow-field to deform a parametrically defined implicit surface. When the same flow-field is used with MeshSDF, the deformation is not curvature-based. A genus-0 surface morphs into a sphere using our method while MeshSDF retains high-curvature regions.

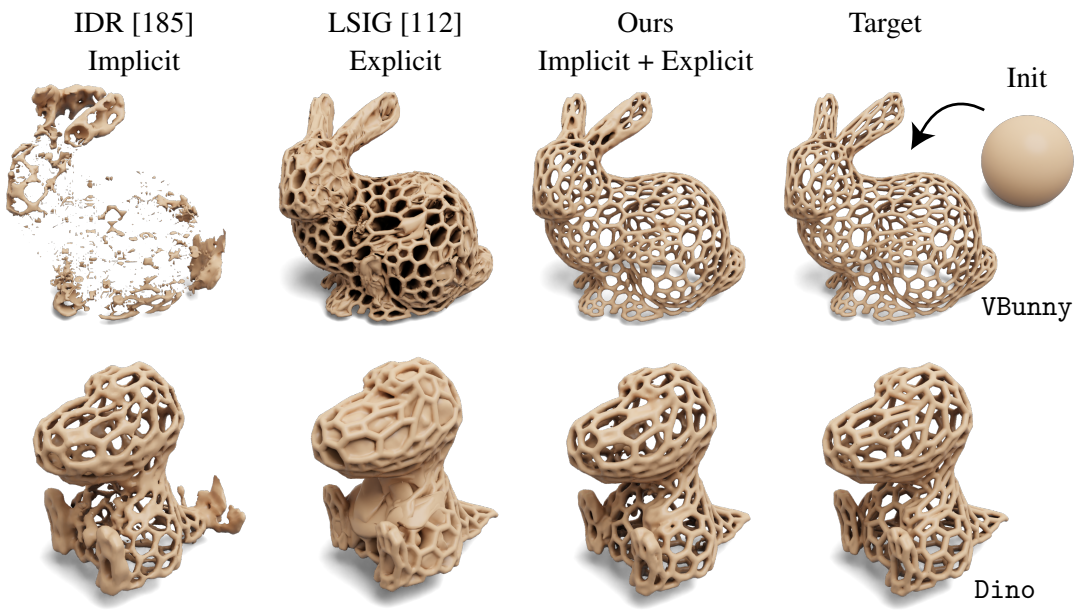


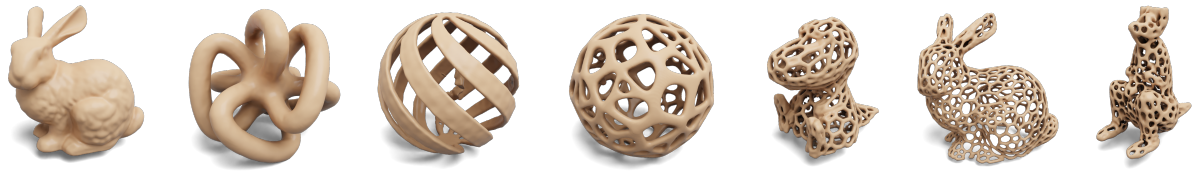
Figure 2.10. Inverse rendering of high-genus shapes. A spherical surface is used for initialization. IDR [185], which uses differentiable rendering of implicit surfaces does not recover fine-details. LSIG [112] uses a triangle-mesh and cannot change the topology post initialization. Using an explicit differentiable renderer to optimize implicit geometry, we can change the topology during optimization and recover fine-details. IDR requires an object mask and a neural renderer.

on a parametrically defined ϕ. Yang *et al.* [183] minimize an objective function which is handcrafted for a specific level-of-smoothness. Applying curvature-based flow is infeasible with their method since it would require a new optimization objective for each level-of-smoothness. As MeshSDF$^{\theta}$ [135] does not evolve the surface according to the level-set equation, the flow obtained with it is incorrect. We show an example flow on a genus-0 surface in Figure 2.9.

2.6.2 Inverse Rendering of Geometry

We propose an inverse-rendering method which uses a differentiable renderer designed for triangle meshes to optimize geometry defined using parametric level-sets. As in the case of recent methods [10, 113, 185] which recover geometry and reflectance from a set of multi-view images, we use an analysis-by-synthesis approach. A photometric error comparing the captured and the rendered images is minimized using gradient descent. The gradients of the error function are used to define an explicit flow-field. A corresponding Eulerian deformation field is obtained

Table 2.1. Quantitative evaluation on inverse rendering of geometry. An initial sphere is optimized to a diverse set of shapes from multi-view images. Chamfer distance is reported for geometric consistency and PSNR is reported to evaluate the visual appearance of optimized geometry. IDR [185] w/o Mask does not converge for any of the shapes (marked –). Methods marked with † require object masks. LSIG [112] works well for a genus-0 shape (`Bunny`) but struggles with high-genus shapes. Best numbers are in orange. Shapes recovered using our method are shown on the top.



Method	Bunny	Genus6	Rind	VSphere	Dino	VBunny	Kangaroo	
IDR - w/o Mask [185]	–	–	–	–	–	–	–	
†IDR - Phong [185]	9.67	3.71	10.24	16.93	3.71	14.80	2.77	
†IDR - Neural [185]	9.84	1.35	0.21	0.16	2.07	9.11	5.43	Chamfer
LSIG [112]	0.06	2.85	3.94	4.78	2.09	4.66	1.80	$\times 10^{-3}$
Ours	0.18	0.12	5.56	3.71	1.25	0.10	1.62	↓ better
†IDR - Phong [185]	21.52	18.84	15.89	18.28	20.01	17.27	21.67	
†IDR - Neural [185]	23.10	28.70	26.24	25.70	22.49	16.62	21.74	PSNR
LSIG [112]	38.51	25.67	22.81	24.06	25.11	21.77	25.52	dB
Ours	38.86	32.94	28.46	30.50	28.16	29.62	26.48	↑ better

to evolve the level-set function Φ. As a result we can take large steps in inverse rendering of geometry with unconstrained topology and guarantees on mesh quality. The optimization is also more robust and does not require an object-mask as in the case of IDR/DVR [113, 185].

Following the approach of Nicolet *et al.* [112], we focus on geometry recovery from synthetic scenes with known reflectance. As in the case of [93], we formulate the forward rendering model with a collocated camera and a point-light. We use a diffuse-Phong BRDF with known albedo and a single-bounce rendering function. Although we choose Nvdiffrast [74] as the differentiable rasterizer in our method, in theory it can be swapped with any other differentiable renderer. Starting from an initial estimate Φ, for each time-step t we first extract the triangle mesh $\partial\Omega_L^t$ and minimize the following objective:

$$\min_{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_k \in \partial\Omega_L^t} \mathscr{E} = \sum_j ||I(j) - R(j, \partial\Omega_L^t, \{\mathbf{n}_i\})||^2, \tag{2.15}$$

where R is the rendering function, $I(j)$ is the ground-truth radiance at pixel j and $\{\mathbf{n}_i\}$ is the set of all vertex normals. We use the gradients of $\mathscr{E}$ to define the flow-field as $\mathbf{V}^t(\mathbf{x}_i) = -\frac{\partial \mathscr{E}}{\partial \mathbf{x}_i} - \lambda \mathbf{L}\mathbf{x}_i$, where $\mathbf{L}\mathbf{x}_i$ is used for smooth evolution. Ablation is given later in this chapter. Taking a single descent step for θ is sufficient since the evolution does not need to follow a specific trajectory (unlike in § 2.6.1).

We evaluate the recovery on a diverse set of shapes, each of which is rendered from 100 random viewpoints. We use IDR [185] and LSIG [112] as baselines. We test IDR in three settings, 1) w/o Mask, 2) With known Phong shading, and 3) Using the Neural renderer in [185]. Quantitative comparisons are in Table 2.1 and qualitative comparisons are in Fig. 2.10. For a genus-0 shape (`Bunny`) LSIG [112] is able to recover accurate geometry, but the optimized meshes can have self intersections as shown on the left. It struggles with high-genus shapes

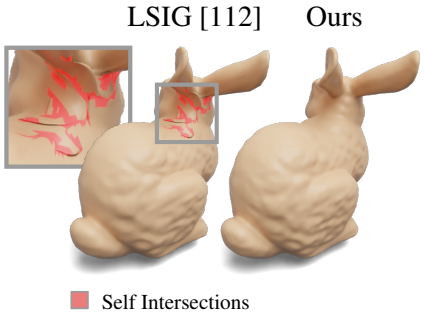


Figure 2.11. Optimizing triangle-meshes can introduce self intersections. LSIG [112] works well for a genus-0 shape in terms of the reported metrics but can have optimized meshes with self-intersecting faces. We highlight such faces for the `Bunny` object.

as the mesh connectivity remains static throughout the optimization routine. Even with correct topology at initialization, the recovery is erroneous, as we show later in this chapter. Comparisons with IDR [185] are in Figure 2.12.

2.6.3 User-defined Shape Editing

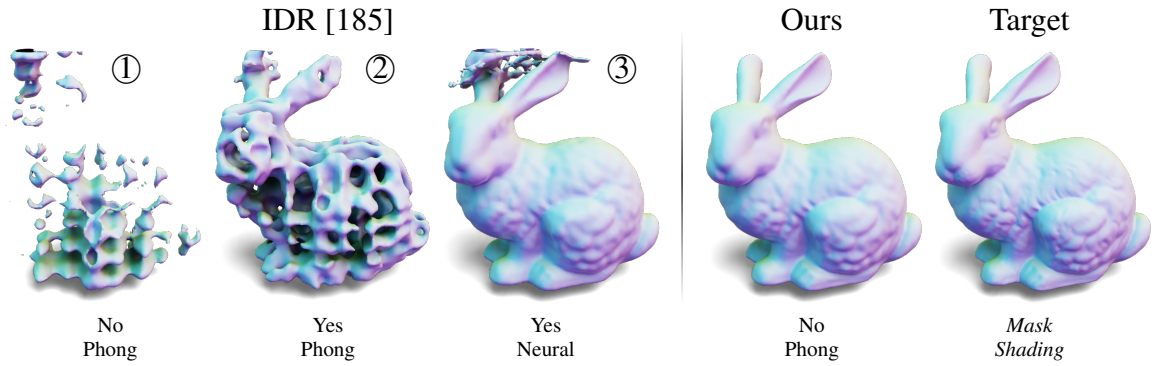


Figure 2.12. Qualitative comparison for inverse rendering using implicit representations. We evaluate IDR in three different settings. ① Without object-mask supervision it fails to converge to a reasonable geometry. ② With a known-reflectance model (Phong) the silhouette of the object is recovered but without any details. ③ It requires a rendering network (unknown reflectance) and an object-mask for good convergence. The proposed method recovers accurate geometry without a mask or a rendering network.

In this section, we demonstrate deformation operations on parametric level-sets using constraints defined by a user. The problem setup is in line with the extensive literature [21, 150, 151, 187] on shape editing for triangle meshes. At $t = 0$, we first extract an explicit mesh from a neural implicit surface using MC [90]. A user can specify handle regions on this surface to either rotate, translate or freeze parts of the shape along with their target locations. This generates a sparse deformation flow-field on the surface which we *densify* by minimizing a thin-shell energy function that penalizes local stretching and bending on the surface [161]. Estimating the flow-field requires solving a linear system $-k_S\Delta\mathbf{V} + k_B\Delta^2\mathbf{V} = 0$, where Δ and Δ^2 are the laplacian and bi-laplacian operators. k_S and k_B are weighting terms for stretching and bending respectively. Additional constraints which adhere to the user specifications are also added to the linear system [20]. With the obtained flow-field, we update the parameters of the level-set function such that the surface propagation is as intended. We also use gradient regularization as in [49]. We discuss the method with more details in Section 2.7.4. We show example deformations in Fig. 2.13. Note that the method by Yang *et al.* [183] achieves similar results, but is slow (takes $\sim$ 10 hours per shape) and it deforms only the explicit surface and not

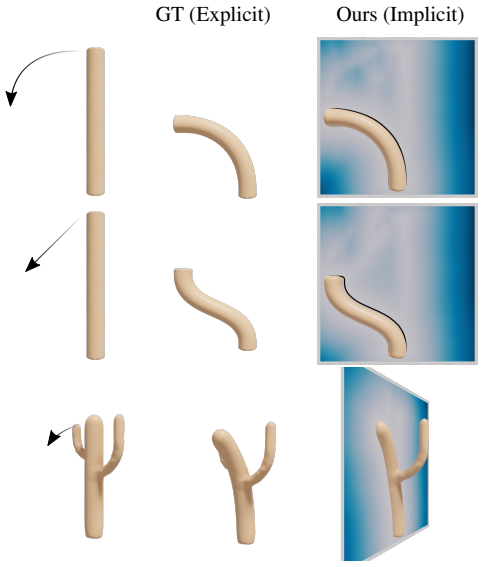


Figure 2.13. User-defined editing on parametric level-sets.

the implicit function. Our method deforms the implicit surface with a 60× speedup (∼ 10 min per shape).

2.7 Additional Derivations and Results

2.7.1 Differentiable Surface Rendering

We first draw comparisons between Differentiable Volumetric Rendering[2] [113], Implicit Differentiable Renderer (IDR) [185] and Differentiable Iso-Surface Extraction in Result 3. Next, in Result 4, we show that methods in [113, 185] deviate from the level-set theory for tangential flow fields.

Result 2.3. *Surface evolution using differentiable ray-marching of parametric implicit surfaces [100, 185] is the same as using differentiable iso-surface extraction [135] when the viewing direction* $\mathbf{v}_u$ *is parallel to the normal* $\mathbf{n}$ *at the intersection point* $\mathbf{x}_u$*. The parameters* θ *for the level-set function* Φ *are updated as:*

[2]Here, volumetric rendering is a misnomer. It's really surface rendering for geometry defined with volume/occupancy.

$$\theta \leftarrow \theta - \lambda \sum_{\mathbf{x}_u} \mathbf{V} \cdot \nabla\Phi \frac{\partial \Phi}{\partial \theta}.$$

Proof. Inverse rendering methods in [113, 185] find a surface-intersection point by marching a along ray that is spawned from the camera centre $\mathbf{c}$, in the direction $\mathbf{v}_u$, for each pixel u. The intersection point $\mathbf{x}_u$ is analytically defined as:

$$\mathbf{x}_u = \mathbf{c} + d_u \mathbf{v}_u, \tag{2.16}$$

where d_u is the depth for $\mathbf{x}_u$. The distance d_u in [113] is estimated from the camera centre. For IDR [185], the distance is computed from a point $\mathbf{y}$ close to the surface which modifies (2.16) to $\mathbf{x}_u = \mathbf{y}_u + d_u \mathbf{v}_u$ ($\mathbf{y}_u$ is denoted as $\mathbf{x}_0$ in [185]). In both the methods, a rendering loss function $\mathscr{L}$ (photometric error) is computed for pixels $u \in U$. To update the geometry parameters θ, the gradient $\frac{\partial \mathscr{L}}{\partial \theta}$ is computed:

$$\frac{\partial \mathscr{L}}{\partial \theta} = \sum_{u \in U} \frac{\partial \mathscr{L}}{\partial \mathbf{x}_u} \frac{\partial \mathbf{x}_u}{\partial \theta} = \sum_{u \in U} \frac{\partial \mathscr{L}}{\partial \mathbf{x}_u} \frac{\partial \mathbf{x}_u}{\partial d_u} \frac{\partial d_u}{\partial \theta} = \sum_{u \in U} \frac{\partial \mathscr{L}}{\partial \mathbf{x}_u} \cdot \mathbf{v}_u \frac{\partial d_u}{\partial \theta} \tag{2.17}$$

To compute $\frac{\partial d_u}{\partial \theta}$, we slightly deviate from the derivations in [113, 185] for clarity. For points $\mathbf{x}_u$ on the implicit surface, we know $\Phi(\mathbf{x}_u) = 0$. Using implicit differentiation:

$$\begin{aligned}
& \frac{\partial \Phi}{\partial \theta} + \frac{\partial \Phi}{\partial \mathbf{x}_u} \frac{\partial \mathbf{x}_u}{\partial \theta} = 0 \\
\iff & \frac{\partial \Phi}{\partial \theta} + \frac{\partial \Phi}{\partial \mathbf{x}_u} \frac{\partial \mathbf{x}_u}{\partial d_u} \frac{\partial d_u}{\partial \theta} = 0 \\
\iff & \frac{\partial \Phi}{\partial \theta} + \frac{\partial \Phi}{\partial \mathbf{x}_u} \cdot \mathbf{v}_u \frac{\partial d_u}{\partial \theta} = 0 \qquad \triangleleft \text{ From (2.16)} \\
\iff & \frac{\partial d_u}{\partial \theta} = -\frac{1}{\frac{\partial \Phi}{\partial \mathbf{x}_u} \cdot \mathbf{v}_u} \frac{\partial \Phi}{\partial \theta}
\end{aligned} \tag{2.18}$$

From (2.17) and (2.18) we have:

$$\frac{\partial \mathscr{L}}{\partial \theta} = -\sum_{u \in U} \frac{\partial \mathscr{L}}{\partial \mathbf{x}_u} \cdot \frac{\mathbf{v}_u}{\frac{\partial \Phi}{\partial \mathbf{x}_u} \cdot \mathbf{v}_u} \frac{\partial \Phi}{\partial \theta} \approx -\sum_{u \in U} \frac{\partial \mathscr{L}}{\partial \mathbf{x}_u} \cdot \frac{\nabla \Phi}{|\nabla \Phi|^2} \frac{\partial \Phi}{\partial \theta} \text{ (when } \mathbf{v}_u \to \nabla \Phi) \tag{2.19}$$

Taking $-\frac{\partial \mathscr{L}}{\partial \mathbf{x}_u} = \mathbf{V}$, we can update the parameters θ using (2.19) and gradient descent as:

$$\theta \leftarrow \theta - \lambda \sum_{\mathbf{x}_u} \mathbf{V} \cdot \nabla \Phi \frac{\partial \Phi}{\partial \theta}. \tag{2.20}$$

Equation (2.11) shows that DVR/IDR and MeshSDF$^\theta$ are closely related, while Result 1 shows that these methods do not agree with the level-set theory. □ □

Result 2.4. *Differentiable ray-marching of parametric implicit surfaces [100, 185] disagrees with the level-set equation for tangential components* $\mathbf{V}^\perp$ *of the flow field* $\mathbf{V}$*. The change in parameters* $\Delta\theta$ *is:*

$$\Delta\theta = \lambda \sum_{\mathbf{x}_u} \pm |\mathbf{V}^\perp| \tan(\arccos(\nabla \Phi \cdot \mathbf{v}_u)) \frac{\partial \Phi}{\partial \theta}. \tag{2.21}$$

Proof. From (2.19), we know the change in parameters θ for methods in [113, 185] is:

$$\Delta\theta = \lambda \sum_{\mathbf{x}_u} \mathbf{V} \cdot \frac{\mathbf{v}_u}{\nabla \Phi \cdot \mathbf{v}_u} \frac{\partial \Phi}{\partial \theta}. \tag{2.22}$$

When the flow-field is tangential to the surface, $\mathbf{V} = \mathbf{V}^\perp$, and $\mathbf{V}^\perp \perp \nabla \Phi$. The change in parameters for this flow-field is:

$$\Delta\theta = \lambda \sum_{\mathbf{x}_u} \frac{\mathbf{V}^\perp \cdot \mathbf{v}_u}{\nabla \Phi \cdot \mathbf{v}_u} \frac{\partial \Phi}{\partial \theta}. \tag{2.23}$$

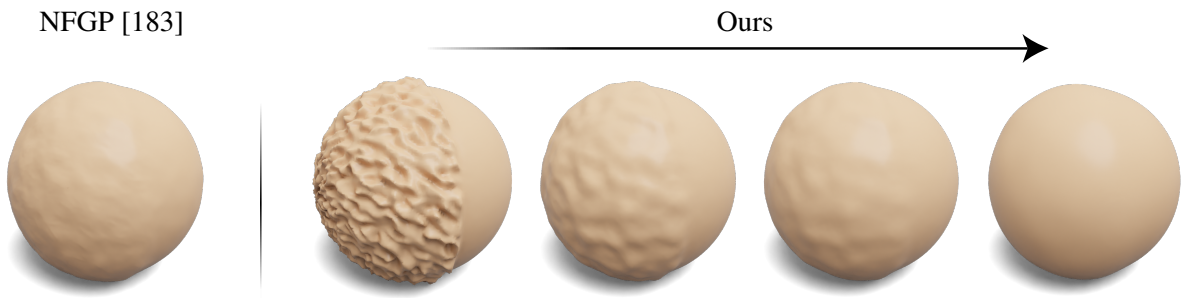


Figure 2.14. Surface smoothing evolution on a half-noisy sphere. (*Right*) Our method can evolve a noisy surface using an explicit diffusion flow-field. This allows smoothing on implicitly defined surfaces with increasing levels of smoothness. (*Left*) Yang *et al.*'s [183] method uses an optimization objective for a single level of smoothness.

When $\Phi : \mathbb{R}^2 \mapsto \mathbb{R}$ is a level-set function defined in 2D, we can modify (2.23) as:

$$\begin{aligned}
\Delta\theta &= \lambda \sum_{\mathbf{x}_u} \pm |\mathbf{V}^{\perp}| \frac{\sin\alpha}{\cos\alpha} \frac{\partial \Phi}{\partial \theta} \qquad \text{where, } \alpha = \arccos(\nabla\phi \cdot \mathbf{v}_u) \text{ and } \Phi \text{ is an SDF} \\
&= \lambda \sum_{\mathbf{x}_u} \pm |\mathbf{V}^{\perp}| \tan\alpha \, \frac{\partial \Phi}{\partial \theta} \\
&= \lambda \sum_{\mathbf{x}_u} \pm |\mathbf{V}^{\perp}| \tan(\arccos(\nabla\Phi \cdot \mathbf{v}_u)) \, \frac{\partial \Phi}{\partial \theta}. \qquad (2.24)
\end{aligned}$$

The parameters here could change depending on the angle subtended between the normal $\nabla\Phi$ and the viewing direction $\mathbf{v}_u$. This dependence on the viewing direction is a result of using differentiable ray-marching. When the function is deformed using the level-set method described in the main paper, the change in parameters $\Delta\theta$ does not depend on the viewing direction and is 0. For 3D level-set functions, the parameters could still change as the term $\mathbf{V}^{\perp} \cdot \mathbf{v}_u$ in (2.23) could be non-zero. □

2.7.2 Curvature-based Deformation

For each shape, we optimize a SIREN [143] MLP with 5 layers, each of which has a 512-sized vector output. We use the publicly-released code[3] by Yang *et al.* [183] for SDF queries and optimization of the network. The flow-field defined in Equation (14) is used for

[3] https://github.com/stevenygd/NFGP

smoothing. The learning rate is 10^{-6}, the time-delta Δt is 0.95 and an eikonal regularization term (enforcing $\nabla\Phi = 1$) [50] is used with the error function defined in Equation (5). The regularization term is weighed 10^{-3} times against the main objective. We use Adam optimizer [70] and use 200 gradient steps for each time step. The Lagrangian surface is extracted using Marching Cubes at $(120 \pm 3)^3$ resolution. We are attaching a script file with the main function (check out `smooth.py`). Figure 2.14 shows the surface evolution with respect to time on a half-noisy sphere (similar to [160]). In addition to the results in the main manuscript, we show qualitative comparisons with other methods in Figure 2.15.

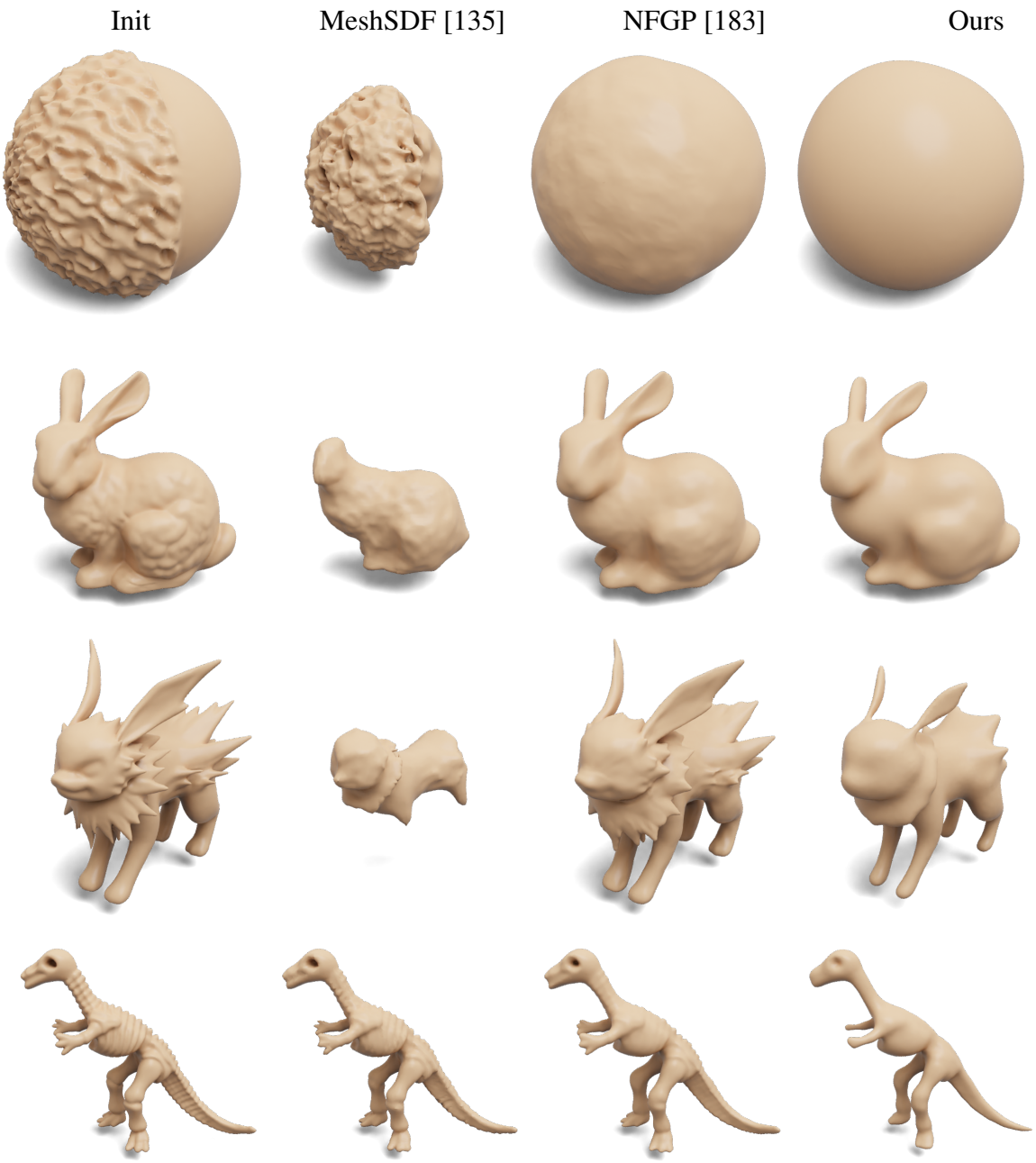


Figure 2.15. Qualitative comparisons for surface smoothing. We apply surface smoothing on four computer graphics models encoded as parametric level-sets. Best viewed when zoomed-in.

2.7.3 Inverse Rendering

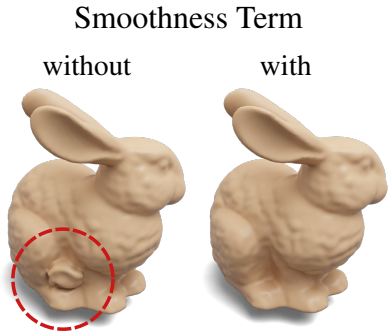


Figure 2.16. Ablation on smoothness regularization term in the flow-field for inverse rendering.

Our method uses the same spherical initialization for all the shapes. The network is composed of 4 layers, with 512 neurons in each layer. The learning-rate is 2×10^{-6}, the weight-decay factor is 0.1 and the time-delta Δt is 10^{-4}. Each object is confined in a 2^3 volume and is rendered with Nvdiffrast [74]. The weight for the smoothness regularization term linearly decreases from 10^{-3} to 0. An ablation is shown in Figure 2.16. We do not observe reliable improvements quantitatively with the regularizer, but do observer better convergence. The Phong shading model is used with 0.55 as the albedo value. We show an experiment with different topological initializations for LSIG [112] in Figure 2.17. All the qualitative comparisons for results in Table 1 (main) are in Figure 2.18. Sample code is in `inverse.py`.

2.7.4 User-defined Deformation

As discussed in the main paper, we use the thin-shell energy [161] loss to densify a sparsely defined flow-field from user inputs. It induces a flow field which is characterized using an Euler-Lagrange equation:

$$-k_s \Delta \mathbf{V} + k_b \Delta^2 \mathbf{V} = 0. \tag{2.25}$$

We know $\mathbf{V}$ for the regions on the surface where the user specifies the deformation and would like to estimate it for the entire surface such that there is minimum bending and stretching. Equation 2.25, however is a result of a linearized version of the thin-shell energy loss. In

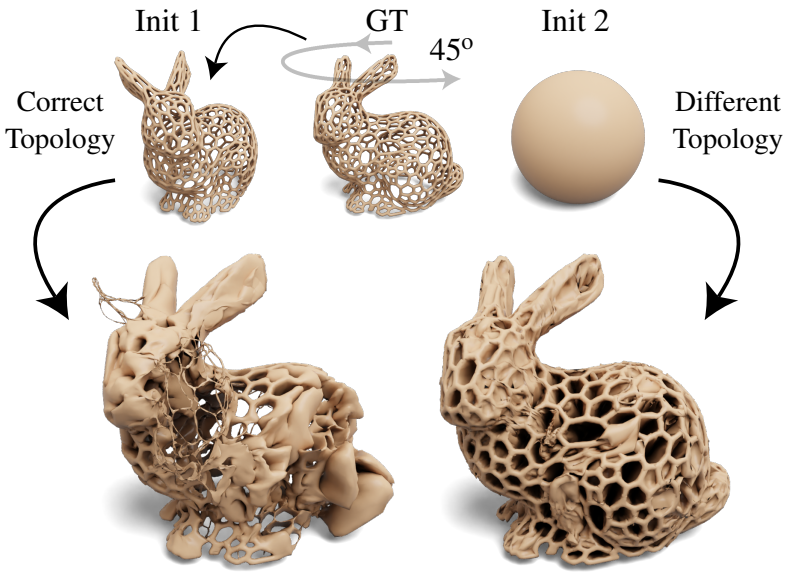


Figure 2.17. LSIG [112] converges to a poor estimate with correct topology at initialization. Optimizing a triangle mesh for inverse rendering problems is non-trivial even when the initialization is close to the ground truth and has correct topology. For `VBunny` we rotate the target geometry and use it as initialization. The shape optimized using LSIG [112] has correct topology but with poor mesh quality and reconstruction accuracy.

this case, the resultant deformations can be unpleasant. This is required for explicit surface deformation methods as they are expected to be conducive to real-time editing. On the contrary, our goal is to have plausible deformations which can be slightly more time consuming. For each editing operation ($\mathbf{x} \rightarrow \mathbf{x}'$), we break the deformation in T time steps as $\mathbf{x} \rightarrow \mathbf{x}^1 \rightarrow \mathbf{x}^2 \rightarrow \mathbf{x}^3 \ldots \mathbf{x}^T$. For each time-step t, we solve for (2.25) (using a sparse linear solver) with the following user constraints:

$$\mathbf{V}(\mathbf{x}_h^{t-1}) = \mathbf{x}_h^t - \mathbf{x}_h^{t-1}, \tag{2.26}$$

where $\mathbf{x}_h \in \mathscr{H}$ belongs to a set of handle vertices for which the user defines deformation. Having obtained the flow-field for all the points on the surface, we evolve the surface using an Euler step as $\mathbf{x}^t = \mathbf{x}^{t-1} + \mathbf{V}(\mathbf{x}^{t-1})$. Note the absence of a delta term (Δt) in the Euler step. This is because of how we define the flow-field; we know exactly where the surface is expected to be at a given time step (From (2.26)). This is different from gradient-based Euler integration where we take



Figure 2.18. Qualitative comparisons for Inverse Rendering of Geometry.

small steps in the direction of the flow-field. Since we know *exactly* where the surface is at time t (can assume $\Phi(\mathbf{x}) = 0$ for surface points), we can tweak the objective defined in Equation 5

(main):

$$\min_{\theta} J = \frac{1}{|\partial\Omega_L^t|} \sum_{\mathbf{x}\in\Omega_L^t} ||\Phi(\mathbf{x})||^2 + \beta \sum_{\mathbf{x}\in\Omega} (|\nabla\Phi(\mathbf{x})| - 1)^2, \tag{2.27}$$

where we also add Eikonal regularization. An example deformation is shown in Figure 2.19 including a comparison with NFGP [183]. Note that our goal with these experiments is to showcase the versatility of the level-set method and not propose a method which performs accurate and real-time deformations; we would still pose this application of user-editing as a proof of concept. Several explicit-surface based methods [19] exist which can probably generate better deformations.

Implementation Details

We use pre-trained neural implicit representations provided in the publicly-released code by Yang *et al.* [183]. The number of time steps T is 20. The learning rate is 2×10^{-6} and 750 gradient steps are taken to minimize Equation 2.27 using Adam [70]. The regularization weighting term $\beta = 10^{-4}$. We provide a code snippet in `deform.py`.

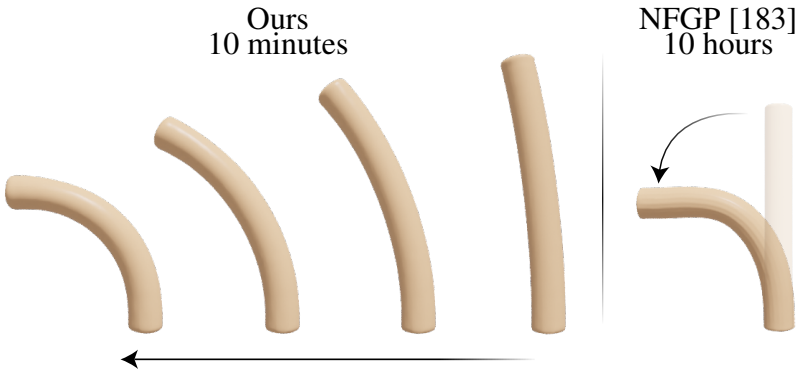


Figure 2.19. User-defined rotation on an implicitly defined cylinder. (*Left*) User editing using our method is faster (takes 10 min) than (*Right*) NFGP which takes 10 hr for the same editing operation.

2.8 Conclusion

Our work formulates a level-set evolution method for parametrically defined implicit surfaces. Unlike previous work, the proposed method is in agreement with the level-set theory. It does not require surface extraction to be differentiable and can be used to apply a variety of mesh-based algorithms to neural implicit surfaces. Although the surface deformation is as dictated by the flow field, the corresponding implicit function may not retain gradient characteristics during evolution, *e.g.*, an SDF may not remain an SDF. This could become a pertinent problem for algorithms like sphere tracing [56] which require reliable distance queries, and is an interesting avenue for future research. We hope that this work encourages further inquiry into recent work on geometry optimization and brings light to some of the elegant methods in computer vision and graphics developed in the pre-deep-learning era.

Chapter 2, in full, is a reprint of the material as it appears in "A Level Set Theory for Neural Implicit Evolution under Explicit Flows", Ishit Mehta, Manmohan Chandraker, Ravi Ramamoorthi, European Conference on Computer Vision (ECCV), 2022. The dissertation author was the primary investigator and author of this paper.

3 | A Theory of Topological Derivatives for Inverse Rendering of Geometry

Chapter 2 fixed the representation and showed how to move a neural implicit surface along a prescribed flow, but it assumed that the flow was already given. This chapter asks where that flow should come from when the target is an image, and in particular how the optimization can change the topology of a shape rather than only its boundary. It derives topological derivatives for rendering, a gradient signal that the deformation method of the previous chapter can consume, and in doing so exposes the question of the image objective that Chapter 4 takes up next.

3.1 Introduction

Recovering geometry from images is a central theme for several problems in vision and graphics, where a common approach is to derive the differential of the rendering functional. Depending on the type of surface representation used, such as Bézier paths [81], triangle meshes [37, 79] or level-sets [43], corresponding gradient flow equations are derived. These works formulate image differentials as shape derivatives, which we posit to be restrictive for inverse problems, since deformations induced by shape derivatives (SD) are limited to surface boundaries. This may lead to local minima when recovering geometry with high-genus topology. In Figure 3.1 we illustrate two inverse problems where shape derivatives do not suffice to recover the optimal shape. In such cases the optimization is required to make updates far away from the boundary by (a) nucleating additional volume in the exterior, and (b) perforating the interior of the shape. Our work theoretically characterizes such shape perturbations in the form of topological derivatives (TD) for inverse rendering.



Figure 3.1. Pathologies of shape derivatives. Previous work on inverse surface reconstruction relies on shape derivatives which are inadequate for instances when (a) the target geometry is far from the initialization, and when (b) target has a hole in the interior. We resolve these failure cases with topological derivatives.

We derive TDs following the definition proposed by Sokolowski and Zochowski [147]. In the case of planar curves, vanishing balls are introduced in their interior or exterior. In the limit, the TD is estimated as the difference in the image functionals for the perturbed and unperturbed shapes. The resulting gradient updates can prompt hole and phase nucleation in regions of high-error (see Figure 3.1). For closed surfaces in 3D, we introduce TDs with respect to conic perturbations through the interior of the visible shape. We observe that visibility terms in SDs are evaluated only on the apparent contours of the visible shape. In contrast, gradient flows using TDs can encourage visibility changes in the interior, resulting in a more robust and accurate recovery.

Informed by recent successes in inverse rendering [12, 97, 168, 170, 184], we use level-sets [121] for surface representation. We build on the extensive literature on *variational* level-sets [40, 65, 148, 190] for surface reconstruction. Our approach differs from the more recent methods on differentiable rasterization [81] and rendering [12, 60, 79, 168] that use differential calculus to derive geometry gradients. The variational framework lends us a common structure to analyze the motion of continuous surfaces in both 2D and 3D. It also provides a natural way to extend shape derivatives to the formulation for topological derivatives. Additionally, using this approach, we are able to draw theoretical insights across literature spanning 3D reconstruction [24, 43], topology optimization [147], differentiable vector graphics [81] and differentiable rendering [60, 74, 79, 134].

We structure this work as three theoretical results interspersed with empirical observations for closed curves in 2D (§ 3.4) and surfaces in 3D (§ 3.5). Previous work on differentiable

rasterization [81] proposes geometry gradients for standard vector graphics representations. In Result 3.1 (§ 3.4.1), we derive a corresponding shape derivative for level-sets in the variational setting. Motivated by the above limitations, we derive topological derivatives for hole and phase nucleation in Result 3.2 (§ 3.5.2). We show practical applications of image vectorization and text-to-vector graphics using an evolution equation that works with arbitrary loss functions. In § 3.5.1 we reason about the necessity of TDs in 3D. Result 3.3 (§ 3.5.2) discusses the construction of conic perturbations and the derivation for TD. We show improvements over previous methods that rely on SDs in terms of speed for visibility optimization (Figure 3.5), accuracy of recovery for complex topology (Figure 3.9) and also show an application of reconstructing shape ambigrams from a single image (Figure 3.10).

3.2 Related Work

In this section, we concisely situate our work with respect to literature on level-sets, differentiable rendering and topological derivatives. A more detailed discussion on literature relevant to our problem formulation, theory and experimental settings can be found within Sections 3.3, 3.4 and 3.5.

Level Sets

We use the level-set method [121] for shape evolution. In the case of 3D, level-sets have been extensively used for shape reconstruction from range data [174], stereo images [40], multi-view images [43] and RGB-D data [111]. In 2D, they have been used for image segmentation as active contour models [25, 28, 67]. These methods derive equations of motion for curves and surfaces as a set of partial differential equations that minimize an optimization objective. Solem and Overgaard [148] provide a geometric viewpoint to analyze these methods. The first variational approach to level-set optimization was proposed by Zhao *et al.* [190] for multi-phase surface motion. More recently, level-sets as signed-distance functions have become a popular choice for generative modeling of 3D shapes [30, 123] and inverse rendering [64, 83, 114, 185].

Differentiable Rendering

Differentiable renderers are designed to estimate the derivatives of a rendering integral with the primary focus on handling geometric discontinuities. One class of differentiable renderers handles primary-visibility discontinuity for triangle meshes [74, 85, 89, 134] with a blurring kernel — an approach that is fast but not physically accurate. Li *et al.* [79] propose edge sampling to correctly handle discontinuities and differentiate the full rendering equation [66], which was later extended for differentiable rasterization of vector graphics [81]. Subsequently, several works aim for numerical efficiency and accuracy [13, 60, 92, 188]. Work by Bangaru *et al.* [12] and Vicini *et al.* [168] is especially relevant to our work as they handle discontinuities for surfaces represented by signed-distance functions. All such works measure image sensitivities with respect to visibility changes on silhouette boundaries, while we focus on meaasuring them in the shape's interior.

Topological Derivatives

The standard definition of topological derivatives is given by Sokolowski and Zochowski [147]. Part of our notation comes from the asymptoptic analysis of TDs in [110]. They have been used in relatively low-dimensional problems in physics for electrical impedence tomography [57] and inverse scattering [24], or in structural engineering to discover optimal support structures [7]. In computer vision, the use of topological derivatives has been limited to a few problems in image processing [77]. The work by Burger *et al.* [23] is perhaps the closest to ours in terms of using the level-set method with SDs and TDs. Please refer to [116] for a recent survey on the usage of TDs.

3.3 Background and Problem Formulation

We consider the problem of inverse shape optimization from a set of images. Our focus is on recovering closed curves in 2D and closed surfaces in 3D. With Γ being the surface under

consideration, we take a variational optimization approach to minimize an image functional,

$$\min_{\Gamma} I(\Gamma) = \int_{\mathscr{I}} g(\mathbf{u})\, d\mathbf{u}. \tag{3.1}$$

The functional I is defined over an image plane $\mathscr{I}$ with $d\mathbf{u}$ as the pixel-area measure. The integrand g is evaluated at pixels $\mathbf{u}$ and can be any reasonable error function. For instance, $g = |\hat{L} - L|$ can evaluate the error between measured (L) and estimated radiance ($\hat{L}$).

We define Γ as a closed n-dimensional surface residing in $\mathbb{R}^{n+1}$, where $n = 1$ for planar curves and $n = 2$ for surfaces in 3D. We use a level-set function $\phi : \mathbb{R}^{n+1} \to \mathbb{R}$ to represent Γ as, $\Gamma := \{\mathbf{x} : \phi(\mathbf{x}) = 0\}$. We use the standard convention [121] to represent the interior of Γ as $\Omega_\Gamma = \{\mathbf{x} : \phi(\mathbf{x}) \leq 0\}$ and the exterior as $\mathbb{R}^{n+1} \setminus \Omega_\Gamma$.

To recover the optimal surface, we require a notion of gradient in the variational setting — for which we build on the framework of variational level-sets [148]. Consider the surface Γ to be a point on an m-dimensional manifold M of admissible surfaces. The functional I to be minimized can be considered a scalar function that maps $M \to \mathbb{R}$. We use the idea of *differentials* (and corresponding Gâteaux derivatives) from differential geometry [104]. We derive the gradient $\nabla_M I$ and optimize Γ with an initial value problem: $\dot{\Gamma}(t) = \nabla_M I(\Gamma(t))$, with Γ_0 as the surface at initialization. The optimal surface Γ^* can be then be recovered using a regular surface evolution of $t \mapsto \Gamma(t)$. By definition, the gradient flow ∇I is composed of only normal components as tangential flow fields keep Γ invariant [121]. We refer the reader to [148] for a more rigorous discussion on why ∇I is a descent direction and reduces I. Intuitively, ∇I is a scalar speed in the direction of the normal n at each point $\mathbf{x} \in \Gamma$ such that I is reduced. The corresponding evolution equation for the underlying level-set function is,

$$\frac{\partial \phi}{\partial t} = -\nabla_M I |\nabla \phi(t)|. \tag{3.2}$$

In Sections 3.4 and 3.5 we derive evolution equations of this form to minimize I in the case of

closed curves in 2D and surfaces in 3D respectively. Computing ∇I requires defining an inner product which operates on the tangent space $T_\Gamma M$ at Γ. For w and v normal velocity directions on $T_\Gamma M$, we can define $\langle \cdot, \cdot \rangle_\Gamma : T_\Gamma M \times T_\Gamma M \to \mathbb{R}$ as,

$$\langle w, v \rangle_\Gamma = \int_\Gamma w(\mathbf{x}) v(\mathbf{x}) \, d\sigma, \tag{3.3}$$

where $d\sigma$ is the surface measure. Consider $v \in T_\Gamma M$ as any normal velocity direction of the form $v = -\frac{\psi}{|\nabla \phi|}$. We can deform the surface Γ in this direction by updating the level-set function as $\phi^s = \phi + s\psi$, where s controls the level of deformation and ψ is the speed. The directional (Gâteaux) derivative of I in this direction v can then be defined as [148],

$$dI(\Gamma)v := \frac{d}{ds} I(\phi + s\psi)\Big|_{s=0}. \tag{3.4}$$

For *any* such normal velocities v in the tangent space, if we can reformulate (3.4) as,

$$dI(\Gamma)v = \frac{d}{ds} I(\phi + s\psi)\Big|_{s=0} \stackrel{?}{=} \langle w, v \rangle_\Gamma, \tag{3.5}$$

then w is the gradient $\nabla_M I$. We revisit this definition of Gâteaux derivative (3.5) and the evolution equation (3.2) in the subsequent sections to derive gradient flows using shape and topological derivatives.

3.4 Curves in 2D

We derive equations of motion for closed planar curves subject to a rendering functional. These equations take the form of *shape derivatives* (§ 3.4.1) and *topological derivatives* (§ 3.4.2) and can be used to recover 2D vector shapes with respect to an error function. With this formulation, we show shape evolution for image vectorization (Figure 3.3) and generation of vector graphics from text prompts (Figure 3.4).

3.4.1 Shape Derivatives

Result 3.1. *Let I be an image functional of a closed curve Γ encoded as the 0-isocontour of a level-set function ϕ in $\mathbb{R}^2$. If the interior of the shape corresponds to a foreground scene function g_F and the exterior to a background function g_B then the shape derivative ∇I is,*

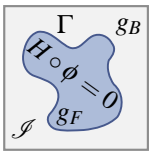
$$\nabla I = g_F - g_B.$$

We start by looking at the image integral (3.1) that integrates a scene function g over the image plane,

$$I(\Gamma) = \int_{\mathscr{I}} g(\mathbf{u})\, d\mathbf{u} = \int_{\mathbb{R}^2} g(\mathbf{x})\, d\mathbf{x}, \tag{3.6}$$

where g is any arbitrary function of color on the image plane. For 2D rasterization, the image plane coincides with $\mathbb{R}^2$ and hence we can change the integral domain as shown in (3.6). We also assume that g is band-limited to the support of the image *i.e.*, $g(\mathbf{u}) = 0$ for $\mathbf{u} \notin \mathscr{I}$. The closed curve Γ partitions the $\mathbb{R}^2$ plane into two regions as per Jordan's Curve Theorem [54]. We delineate the two regions with scene functions g_F and g_B for foreground and background respectively. We use a level-set function ϕ to synthesize a characteristic function $H \circ \phi$, where H is a standard heaviside function. By construction, our function evaluates to 0 in the foreground and 1 in the background. The characteristic function can be used to expand the integrand as,

$$I(\Gamma) = \int_{\mathbb{R}^2} g_F(\mathbf{x})[1 - H(\phi(\mathbf{x}))] + g_B(\mathbf{x}) H(\phi(\mathbf{x}))\, d\mathbf{x}.$$



We can now the derive the shape derivative ∇I using a perturbation of the form $\phi^s =$

$\phi + s\psi$ that corresponds to velocity v. The Gâteaux derivative of I in this direction v is,

$$\begin{aligned} dI(\Gamma)v &= \frac{d}{ds} I(\phi + s\psi)\Big|_{s=0} \qquad \triangleleft \text{From (3.4)} \\ &= \int_{\mathbb{R}^2} g_F \frac{d}{ds}[1 - H(\phi^s)] + g_B \frac{d}{ds} H(\phi^s)\, d\mathbf{x} \,\Big|_{s=0} \\ &= \int_{\mathbb{R}^2} -\delta(\phi)\psi g_F + \delta(\phi)\psi g_B \, d\mathbf{x} \\ &= \int_{\mathbb{R}^2} (g_F - g_B) \frac{-\psi}{|\nabla\phi|} \delta(\phi)|\nabla\phi| \, d\mathbf{x} \\ &= \int_{\Gamma} (g_F - g_B) v \, d\sigma = \langle g_F - g_B, v \rangle_{\Gamma}, \end{aligned} \tag{3.7}$$

where $d\sigma = \delta(\phi(\mathbf{x}))|\nabla\phi(\mathbf{x})|d\mathbf{x}$ is the surface measure [59, 148] and $v = \frac{-\psi}{|\nabla\phi|}$. Note the interchange of integral and differential operators due to Leibniz rule. Comparing the definitions in (3.5) and (3.7), we conclude that the shape derivative for functional I is,

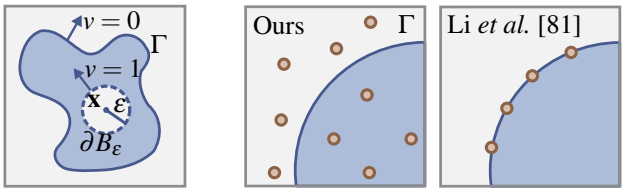
$$\nabla I = g_F - g_B. \tag{3.8}$$

The shape derivative in (3.8) can be used to minimize I using the level-set evolution equation (3.2). As per the domain of the integral in (3.7), this derivative is defined only on the curve Γ. Li *et al.* [81] arrive at a similar result using differential calculus and numerically evolve 2D shapes by explicitly sampling the edges. As shown in Figure 3.3, SDs can fail to prompt crucial topology changes for image vectorization.

3.4.2 Topological Derivatives

The topological derivative measures sensitivities with respect to infinitesimally small perturbations in a shape's volume. We derive the topological derivative in the context of inverse rendering to inform where to nucleate holes and phases in a shape in order to reduce a rendering functional.

Result 3.2. *Consider a closed curve Γ with Ω_Γ as its interior. For a point $\mathbf{x} \in \mathbb{R}^2$ the topological*



(a) (b)

Figure 3.2. (a) We derive topological derivatives for 2D rendering functionals. An infinitesimal-hole of perturbation B_ε with radius $\varepsilon \to 0$ is introduced at point $\mathbf{x}$. A Gâteaux derivative is computed in the direction of a normal velocity v such that $v = 1$ on ∂B_ε and 0 on the unperturbed surface Γ. (b) We find functional forms of shape and topological derivatives to be the same with the only difference being the domain. Numerically, this eliminates the need for explicitly sampling boundaries for shape evolution as in [81].

derivative with respect to hole and phase nucleation is,

$$D_\tau(\mathbf{x}) = g_F(\mathbf{x}) - g_B(\mathbf{x}).$$

We follow the standard definition of a topological derivative from Sokolowski and Zochowski [147]. We focus on hole nucleation in this section and defer the derivation for phase nucleation to Section 3.6.1. Let Ω_Γ denote the interior of a given curve Γ. For a point $\mathbf{u} \in \mathscr{I}$ on the image plane, we introduce a small circular hole $B_\varepsilon(\mathbf{u})$ of radius ε. The topological derivative of the image functional I at a point $\mathbf{x} \in \Omega_\Gamma$ can then be defined as a scalar function,

$$D_\tau(\mathbf{x}, \Omega_\Gamma) \coloneqq \lim_{\varepsilon \to 0} \frac{I(\Omega_\Gamma \setminus \overline{B_\varepsilon(\mathbf{x})}) - I(\Omega_\Gamma)}{V(B_\varepsilon)}, \tag{3.9}$$

where $I(\Omega_\Gamma)$ is the functional value with the unperturbed shape, $\overline{B_\varepsilon}$ is the closure set of points inside the perturbation and $V(B_\varepsilon) = \pi\varepsilon^2$ is the area of the circle. The corresponding asymptotic expansion of this definition [110, 147] is,

$$I(\Omega_\Gamma \setminus \overline{B_\varepsilon}) = I(\Omega_\Gamma) + V(B_\varepsilon) D_\tau(\mathbf{x}, \Omega_\Gamma) + o(V(B_\varepsilon)). \tag{3.10}$$

We compute the Gâteaux derivative of this expression in the direction of a normal velocity v

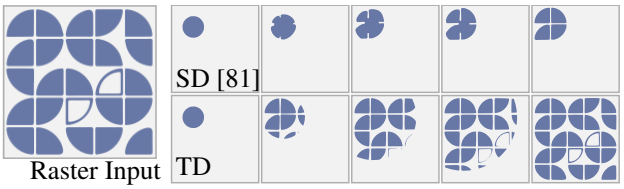


Figure 3.3. Reconstruction of vector images from raster input. Shape derivatives (analagous to curve gradients in [81]) evolve closed curves with visibility changes only on the boundaries. These gradients are sparse and the optimization can plateau at a local minimum (*top*). Topological derivatives can reconstruct a vector image (*bottom*) from a raster input (*left*) with adaptive topological changes.

defined as shown in Figure 3.2 (a). Intuitively, the defined velocity increases the size of the perturbation by a constant and keeps the curve Γ as it is. Formally, $v = 0$ on the unperturbed curve Γ and $v = 1$ at the boundary ∂B_ε of the perturbation hole, such that it smoothly goes to 0 outside a small neighborhood of $\mathbf{x}$. Assuming $\mathbf{x} \notin \Gamma$ and since velocity v is 0 on the unperturbed curve, the shape derivative (3.8) in the direction v is 0:

$$dI(\Omega_\Gamma)v = \int_\Gamma (g_F - g_B)v \, d\sigma = 0. \tag{3.11}$$

From (3.9), (3.10) and (3.11) we can rewrite the TD as:

$$\begin{aligned} D_\tau(\mathbf{x}, \Omega_\Gamma) &= \lim_{\varepsilon \to 0} \frac{1}{V'(B_\varepsilon)} dI(\Omega_\Gamma \setminus \overline{B_\varepsilon})v \\ &= \lim_{\varepsilon \to 0} \frac{1}{V'(B_\varepsilon)} \int_{\Gamma \cup \partial B_\varepsilon} (g_F - g_B)v \, d\sigma. \end{aligned} \tag{3.12}$$

Note that with the introduction of a hole, the perturbed shape is comprised of surface elements from the original domain Γ in addition to the boundary ∂B_ε of the hole — that is, $\hat{\Gamma} = \Gamma \cup \partial B_\varepsilon$ where $\hat{\Gamma}$ is the perturbed curve (see Figure 3.2 (a)). The denominator of (3.12) is a result of the fact that $dV(\varepsilon)v = V'(\varepsilon) = 2\pi\varepsilon$ as $v = 1$ on the hole's boundary. Further simplification yields:

$$
\begin{aligned}
D_\tau(\mathbf{x}, \Omega_\Gamma) &= \lim_{\varepsilon \to 0} \frac{1}{2\pi\varepsilon} \int_\Gamma (g_F - g_B) v \, d\sigma + \\
&\qquad \int_{\partial B_\varepsilon} (g_F - g_B) v \, d\sigma \\
&= \lim_{\varepsilon \to 0} \frac{1}{2\pi\varepsilon} \int_{\partial B_\varepsilon} (g_F - g_B) \, d\sigma \\
&= g_F(\mathbf{x}) - g_B(\mathbf{x}).
\end{aligned}
\tag{3.13}
$$

Equation 3.13 is the topological derivative of the image functional I at a point $\mathbf{x}$. Intuitively, at a given point in the interior of the shape, if the error with respect to the background color is lower than the foreground, formation of a hole can be prompted by increasing the value of $\phi(\mathbf{x})$. We note that D_τ is defined in $\mathbb{R}^2 \setminus \Gamma$. This is in stark contrast with the shape derivatives from Result 1 that encouraged visibility changes only on the curve. By analogy, we also derive topological derivatives for phase nucleation and arrive at the same result, *i.e.*$D_\tau(\mathbf{x}, \mathbb{R}^2 \setminus \Omega_\Gamma) = g_F(\mathbf{x}) - g_B(\mathbf{x})$, where $\mathbf{x}$ is a point in the exterior of the given shape. The full derivation is in Section 3.6.1.

Level-Set Evolution

We can now formulate the evolution equation in (3.2) using both shape and topological derivatives:

$$
\frac{\partial \phi}{\partial t} = -\left[\nabla I(\mathbf{x}) + D_\tau(\mathbf{x})\right] |\nabla \phi|. \tag{3.14}
$$

The SD ∇I is defined on Γ and is 0 elsewhere. The TD D_τ is defined on $\mathbb{R}^2 \setminus \Gamma$. As shown in Figure 3.3, we can use the evolution equation for image-based reconstruction of vector graphics from raster input. When only the first term ∇I is used, the optimization can plateau at a local minimum. The second term is critical for recovering the target geometry, particularly when the target has a differing geometric structure from the initialization. We show more results in this

[1]Text-prompt: "Frontal face of `animal`. Minimal line drawing. Trending on artstation. Plain white background. Black and white."

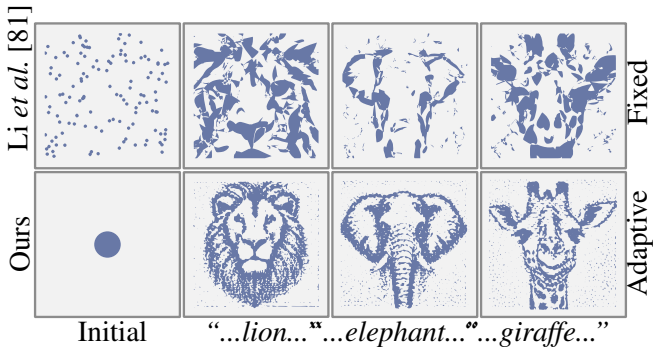


Figure 3.4. Adaptive generation of detailed vector graphics. Our formulation for surface evolution can be used to induce topological changes subject to any differentiable loss functions for images. Here we show an example of generating vector graphics from a text prompt using the score-distillation-sampling loss from [129]. (*top*) Existing methods use differentiable vectorization from [81] and are forced to work with fixed shape topologies. (*bottom*) Conversely, topological derivatives can be used to adaptively generate detailed vector graphics from a simple initialization (*bottom-left*). [1]

context along with the implementation details later in this chapter.

Eliminating Edge Sampling

Comparing Result 1 and Result 2, we find that the topological and shape derivatives are exactly the same except for the domain on which they are defined. This simplifies the evolution equation in (3.14) with a single term for the entirety of $\mathbb{R}^2$,

$$\frac{\partial \phi}{\partial t} = (g_B - g_F)|\nabla \phi|. \tag{3.15}$$

In practice, for numerical optimization, we find that this simplification eliminates the need for explicitly sampling boundaries for shape evolution as done in Li *et al.*'s [81] method for differentiable vector graphics (Figure 3.2 (b)). For complex shapes, with a large set of paths, this leads to faster optimization. We provide details later in this chapter.

Generative Vector Graphics

Using the chain-rule, the evolution equation can be adapted for problems beyond just reconstruction. Given an aribitrary differentiable loss function $\mathscr{L}$, we can evolve the shapes with

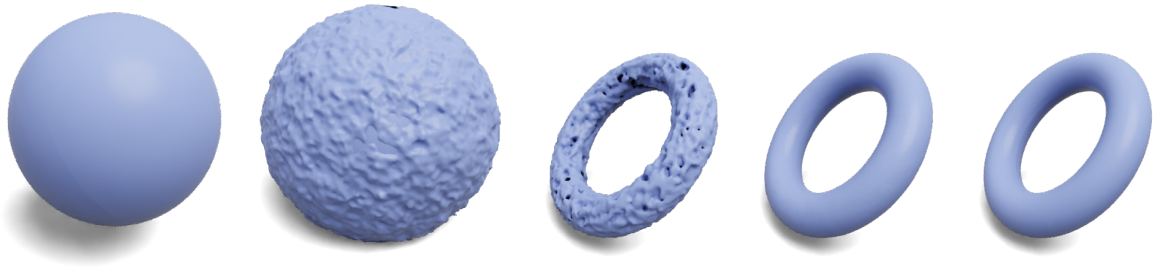


Figure 3.5. Visibility gradients drive topological changes. Given a set of multi-view images, we recover a genus-1 shape from a sphere. When shading gradients are used the optimization makes only local updates. Using visibility gradients the topology and the silhouettes of the recovery match the target. Both the terms are required for optimal recovery.

topological changes such that $\mathscr{L}$ is minimized by the following evolution:

$$\frac{\partial \phi}{\partial t} = \frac{\partial \mathscr{L}}{\partial I}(g_B - g_F)|\nabla \phi|. \tag{3.16}$$

We show an application of generating 2D vector graphics from text-prompts with a text-to-image diffusion model [136]. The loss function $\mathscr{L}$ is the score-distillation-sampling loss from [129]. Previous methods for text-to-svg use the differentiable rasterizer from [81]. Such methods require a fixed number of shapes starting from the intialization and cannot adaptively change the topology in a differentiable manner. Using topological derivatives, we show highly-detailed generation of vector graphics from a relatively simple initialization of a disk (Figure 3.4). More examples and implementation details are given later in this chapter.

3.5 Surfaces in 3D

Similar to the 2D case, our goal is to minimize the image functional I defined in (3.1). Comparing it to the rendering equation [66], I can depend on scene parameters such as the geometry, material and lighting in the scene. We focus on geometry optimization and take the material and lighting parameters as given. The function g is discontinuous in terms of the geometry parameters and hence naively differentiating I using automatic differentiation is

erroneous [79].

3.5.1 Background on Shading and Visibility

To address discontinuities with respect to visibility changes, Gargallo *et al.* [43] derive the shape derivative of I to evolve a level-set function ϕ as,

$$dI(\Gamma) = -\underbrace{\nabla g \cdot \frac{\mathbf{x}}{\mathbf{x}_z^3}\chi}_{\text{Shading}} + \underbrace{(g - g_B)\frac{\mathbf{x}\cdot\nabla(n\cdot\mathbf{x})}{\mathbf{x}_z^3}\delta(n\cdot\mathbf{x})\chi}_{\text{Visibility}}, \tag{3.17}$$

where χ denotes the visibility, n the normal and $\mathbf{x}_z$ the z coordinate. The first term here is similar to shading gradients in more recent differentiable renderers [60, 74, 79, 134], which can be used to make local updates in the interior of the visible surface. The second term is enabled only at the silhouette boundary as per $\delta(n\cdot\mathbf{x})$, where δ is a dirac distribution function. The difference $g - g_B$ moves the visible contours of the surface based on the foreground (g) and background (g_B) radiance. This visibility term is similar to shape derivatives in the 2D case (3.8), with additional factors to account for the distance from the camera, visibility and shape curvature.

Need for Topological Derivatives

For recovering geometry with complex topology, we find the visibility gradients to be critical in terms of recovering the overall structure. We empirically show this in Figure 3.5, where a genus-1 shape is optimized from a sphere using the two gradient forms. When only shading gradients are used, the optimization fails to evolve the topology; with visibility gradients in isolation we can retrieve the correct topology but not the details; when both are used together, we recover the target shape. Despite this significance, the optimization signal obtained from visibility gradients is quite sparse. As evident from (3.17), these gradients are evaluated only at the extremeties of the shape and cannot induce visibility changes in the interior of the shape. In flatland, this means that visibility gradients account for changes only at the extreme points of the convex set covering a shape (as shown in Figure 3.6 (b)). This makes the optimization

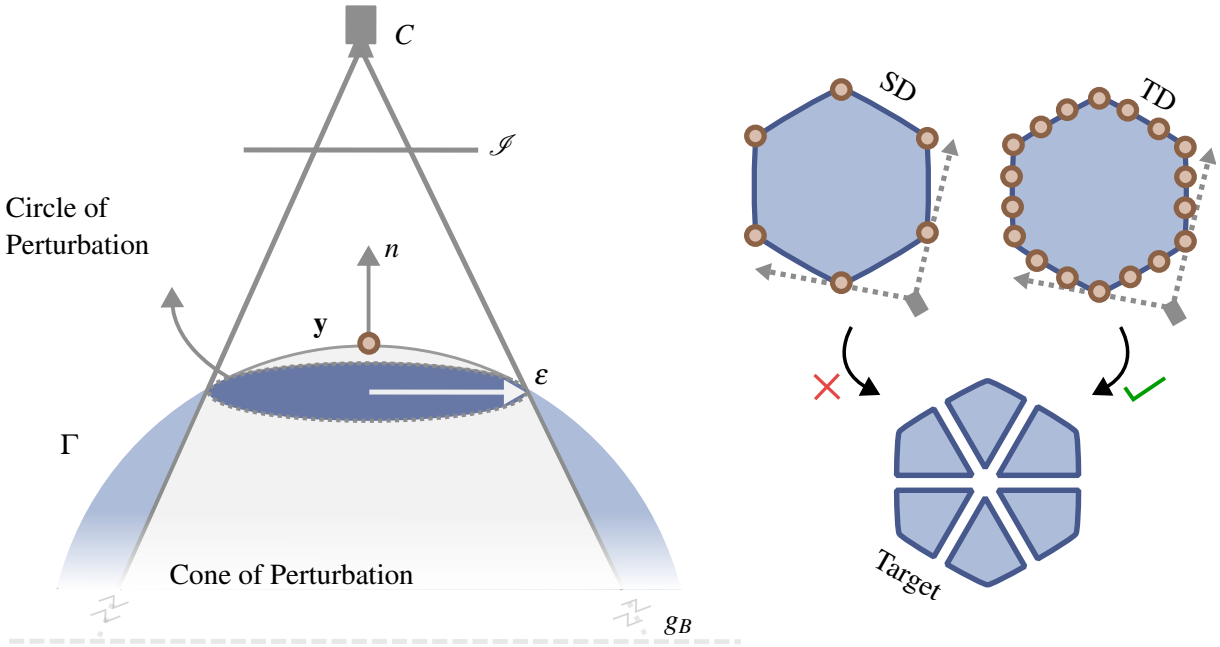


Figure 3.6. (a) We derive topological derivative at a point **y** on a surface Γ using a vanishing cone of perturbation originating at the camera C. (b) Visibility terms from shape derivatives (SD) are evaluated only at the apparent contours. In flatland, these points resemble the extreme points of the convex set of the shape. A target shape with the same convex booundary as the initial shape cannot be recovered using the visibility terms in SDs. TDs evaluate visibility changes on the entire surface and can resolve such ambiguities.

susceptible to unexpected local minima as topological changes are less likely in regions away from the surface. Our goal is to resolve this by enabling topological changes in the interior regions using gradients that measure visibility changes on the *entire* surface.

3.5.2 Topological Derivatives

Result 3.3. *Let I be an image functional of a closed and connected surface Γ in $\mathbb{R}^3$. The functional I integrates a scene function g for the surface and g_B for the background. Then the topological derivative of I at a point $\mathbf{y} \in \Gamma$ with respect to an infinitesimal conical perturbation from the origin through $\mathbf{y}$ is,*

$$D_\tau(\mathbf{y},\Gamma) = (g(\mathbf{y}) - g_B(\mathbf{y}))\frac{\kappa \mathbf{y}^t \mathbf{y}}{\mathbf{y}_z^3}.$$

In case of surfaces in $\mathbb{R}^3$, there is a large class of infinitesimal perturbations that change the configuration of the shape. In this work we focus on perturbations that influence the genus of the shape. We begin by considering a point $\mathbf{y}$ on a surface Γ at which want to estimate the topological derivative. Without loss of generality, we will assume that the surface Γ is placed in between the camera at the origin and a background scene. A curved circle of planar radius ε is placed on the surface with $\mathbf{y}$ as the center (Figure 3.6 (a)). We term the corresponding planar circle as the circle of perturbation. Starting from the origin, we can find an elliptic cone that inscribes this circle at an angle that is consistent with the normal at $\mathbf{y}$. We use this cone to define our perturbed shape as the difference of the enclosed volume of the original shape and the intersection of the original shape with the cone of perturbation. If Ω_Γ is the interior of Γ and Ω_ε of the perturbation cone, then $\Omega_{\hat{\Gamma}(\varepsilon)} = \Omega_\Gamma - \Omega_\Gamma \cap \Omega_\varepsilon$ is the volume enclosed by the perturbed surface $\hat{\Gamma}$. By construction, we know that as $\varepsilon \to 0$, the perturbed surface $\hat{\Gamma}(\varepsilon) \to \Gamma$. Beyond just the perturbation, we are interested in its resultant effect on the image. The image plane intersects with the cone of perturbation and inscribes an ellipse. Intuitively, as we increase the radius of the circle of perturbation, the area of the ellipse on the image plane increases and the

image accumulates more light from the background.

In the given construction, we now define the topological derivative of I at $\mathbf{y}$ as,

$$D_\tau(\mathbf{y},\Gamma) \coloneqq \lim_{\varepsilon\to 0} \frac{I(\hat{\Gamma}) - I(\Gamma)}{V(\varepsilon)}. \tag{3.18}$$

Note that this is a slightly different notion of the topological derivative as compared to the standard form [147] which introduces vanishing balls B_ε at the points of perturbation. Our deviation from this norm is out of necessity. The error functionals that we consider for inverse rendering operate only on the projection of the shape and hence any perturbations of the standard form in the shape's interior will have no effect on the error. The term $V(\varepsilon)$ in the standard form is the Lebesgue measure of the ball B_ε, although, in our case, we choose $V(\varepsilon) = \pi\varepsilon^2$ as the area of the planar circle. This choice will lend us a limit (3.18) that exists and is finite.

Similar to the 2D case (§ 3.4.2), we define a normal velocity v such that $v = 1$ on the newly formed visible contour around the circle of perturbation, and $v = 0$ on the unperturbed surface. Hence, the Gâteaux derivative of the image functional $I(\Gamma)$ in the direction v is,

$$\langle dI(\Gamma), v\rangle_\Gamma = \int_\Gamma dI(\Gamma) v \, d\boldsymbol{\sigma} = 0, \tag{3.19}$$

where dI is defined as in (3.17). Analogous to the case of TDs in 2D (3.10), by using asymptotic expansion we can rework the definition for 3D in (3.18) as,

$$D_\tau(\mathbf{y},\Gamma) = \lim_{\varepsilon\to 0} \frac{1}{V'(\varepsilon)} dI(\hat{\Gamma}) v. \tag{3.20}$$

Intuitively, as $\varepsilon \to 0$, this definition measures the rate of change in the image of a perturbed shape with respect to the rate of increase in the area of the circle of perturbation. Using the shape

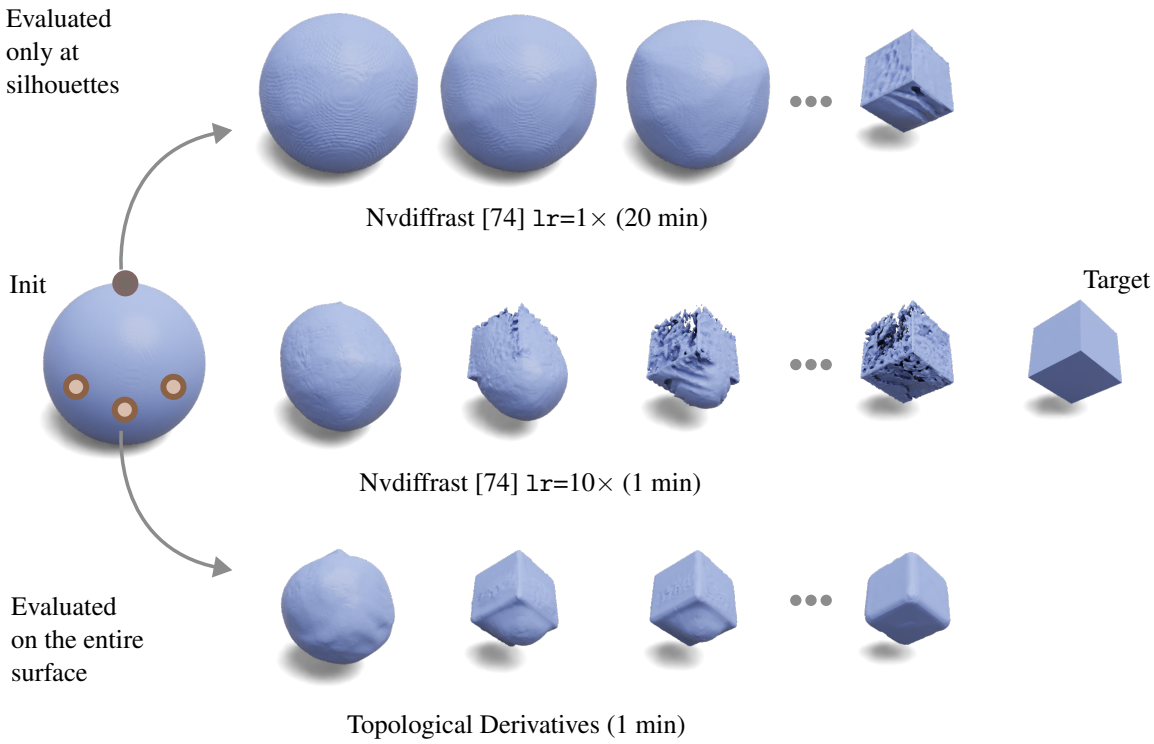


Figure 3.7. Large steps in visibility optimization. Standard differentiable renderers evaluate visibility gradients only at the silhouette. Topological derivatives estimate sensitivies with respect to visibility changes on the entire surface. This enables a more robust way to take large steps in inverse rendering. We compare topological derivatives (*bottom*) with visibility gradients (*top*) from Nvdiffrast [74] for multi-view reconstruction. (*middle*) We observe that silhouette gradients are unstable at higher learning rate (`lr`).

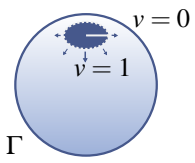


Figure 3.8. Conical perturbation in 3D. A newly visible contour forms around the circle of perturbation, where the normal velocity is $v = 1$, while $v = 0$ on the unperturbed surface.

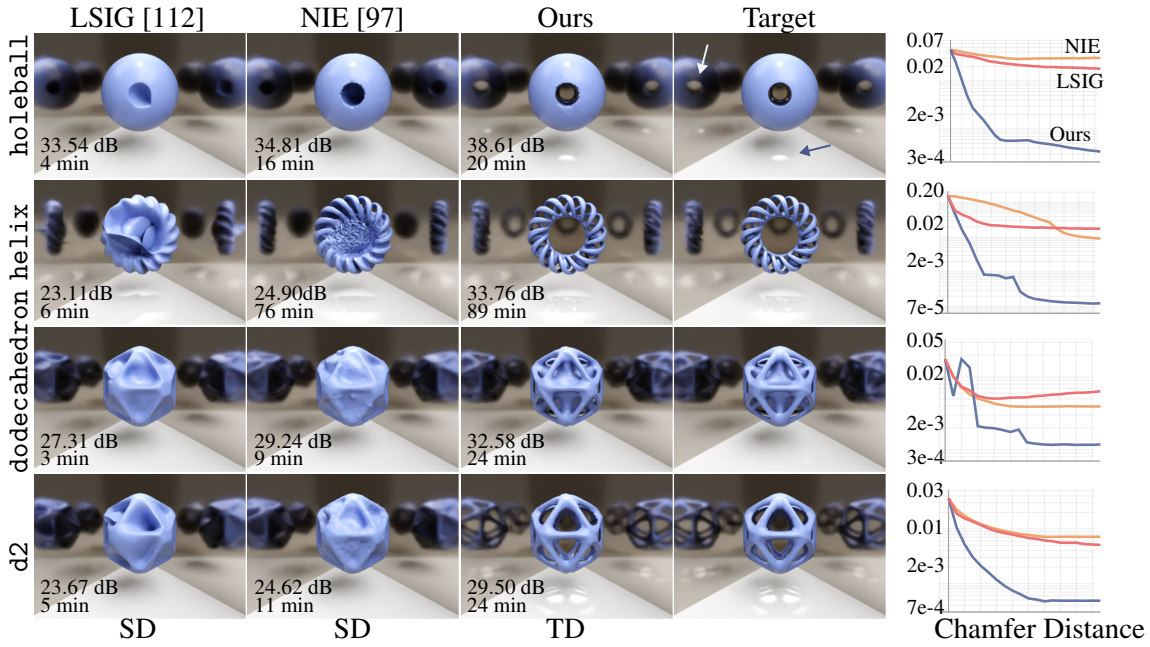


Figure 3.9. Inverse rendering of geometry with complex topology. Given a set of images, we cover complex shapes from a spherical initialization. We compare our method that uses topological derivatives to a differentiable renderer [74] that uses visibility gradients defined on the silhouette. We regularize the gradients from [74] with two recent methods. LSIG [74] uses a triangle mesh representation and can make topological changes using remeshing [18]. NIE [97] uses the same gradients with a level-set function. These approaches struggle with prompting visibility changes in the interior and can get stuck in a local minimum. Convergence plots of chamfer distance are on the right. We also provide PSNR and optimization time for each scene. Our method, as expected, requires a slight time-overhead over NIE to estimate the TD. We place the recovered shapes in front of two glossy surfaces to highlight the quality of recovery.

derivative defined in (3.17) we can replace the $dI(\hat{\Gamma})$ with,

$$D_\tau(\mathbf{y},\Gamma) = \lim_{\varepsilon\to 0}\frac{1}{V'(\varepsilon)}\int_{\hat{\Gamma}}\left[-\nabla g\cdot\frac{\mathbf{x}}{\mathbf{x}_z^3}\chi\right]v + \left[(g-g_B)\frac{\mathbf{x}\cdot\nabla(n\cdot\mathbf{x})}{\mathbf{x}_z^3}\delta(n\cdot\mathbf{x})\chi\right]v\,d\sigma. \tag{3.21}$$

We simplify this integral with two assumptions. First, in the small neighborhood of $\mathbf{y}$, as $\varepsilon\to 0$, the radiance function g is constant. This eliminates the first term in the integrand as $\nabla g = 0$. Next, we assume constant curvature. This results in $\mathbf{x}\cdot\nabla(n\cdot\mathbf{x}) = \kappa\mathbf{x}^t\mathbf{x}$ and curvature [2] κ can be taken out of the integral sign as,

$$D_\tau(\mathbf{y},\Gamma) = \lim_{\varepsilon\to 0}\frac{\kappa}{V'(\varepsilon)}\int_{\hat{\Gamma}}(g-g_B)\frac{\mathbf{x}^t\mathbf{x}}{\mathbf{x}_z^3}\delta(n\cdot\mathbf{x})\chi v d\sigma. \tag{3.22}$$

The term $\delta(n\cdot\mathbf{x})\chi$ constrains the domain for the integral to the set of apparent contours that are visible from the camera. The choice of v further restricts this domain to only the circle of perturbation (which we denote by $\partial\circ$) as by definition, $v = 0$ for all other contours:

$$\begin{aligned} D_\tau(\mathbf{y},\Gamma) &= \lim_{\varepsilon\to 0}\frac{\kappa}{V'(\varepsilon)}\int_{\partial\circ}(g-g_B)\frac{\mathbf{x}^t\mathbf{x}}{\mathbf{x}_z^3}\,d\sigma \\ &= (g(\mathbf{y})-g_B(\mathbf{y}))\frac{\kappa\mathbf{y}^t\mathbf{y}}{\mathbf{y}_z^3}. \end{aligned} \tag{3.23}$$

This is the topological derivative of functional I at a visible point $\mathbf{y}$ on the surface. Unlike the visibility gradients by Gargallo *et al.* [43] in (3.17) and other differentiable renderers [60, 74, 79, 134], the topological derivative in (3.23) can prompt visibility changes on the entire surface.

[2] More details on the relevance of κ are given later in this chapter.

Level-Set Evolution

We can finally use topological derivatives (3.23) and shape derivatives (3.17) together to evolve a level-set function ϕ using the evolution equation from (3.2):

$$\frac{\partial \phi}{\partial t} = -dI(\Gamma)|\nabla \phi| - D_\tau(\mathbf{x}, \Gamma)|\nabla \phi|. \tag{3.24}$$

Although, the shape derivative term ($dI(\Gamma)$) is valid only for primary visibility. We can lift this assumption by replacing the shape derivatives with gradients from a differentiable path tracer for triangle meshes [60, 79] as follows [97, 135],

$$\frac{\partial \phi}{\partial t} = -\frac{\partial I}{\partial \mathbf{x}} \cdot n - D_\tau(\mathbf{x}, \Gamma)|\nabla \phi|. \tag{3.25}$$

This replacement enables higher-order shading and visibility gradients while being practical in terms of availability of multiple mesh-based differentiable renderers that are 1) fast [74, 134], 2) have user-friendly APIs [60, 74, 79, 134], and 3) can handle complex light transport effects [60, 79].

Large Steps in Visibility Optimization

Unlike the visibility gradients in present differentiable renderers, the topological derivative from Result 3.3 is defined for the interior of the visible shape. For shape recovery from multi-view images, there is the immediate benefit that, for each point on the surface, we can obtain visibility sensitivities defined from several viewing angles. Conversely, with standard shape derivatives, visibility gradients are defined for points only when they appear at the silhouette. The resulting optimization signal is quite sparse in practice — and in cases with complex lighting and materials, it is noisy. We find that topological derivatives provide a more robust signal for visibility changes and enable faster optimization. In Figure 3.7 we show a comparison with visibility gradients from nvdiffrast [74] for inverse shape recovery from multi-view images.

Multi-View Reconstruction

We validate the proposed theory using synthetic 3D shapes of complex topology. Given a set of images with known material and environment lighting, we use the level-set evolution in (3.25) to minimize reprojection error. We show qualitative and quantitative results in Figure 3.9 with nvdiffrast [74]. Since [74] is defined for triangle meshes, we regularize the gradients for smoother optimization using LSIG [112] and enable topological changes with NIE [97]. Theoretically, our method becomes an extension of NIE in the form of an additional term in the level-set PDE (3.25) for topological derivatives. Starting from a genus-0 sphere, we find that shape derivatives are insufficient to prompt visibility changes in the shape's interior. The `holeball` example, which requires puncturing three axial holes through the sphere, is a prime illustration of this problem. The regions of hole nucleation are within the silhouette of the shape and hence other methods require shading gradients to prompt the visibility changes — which, as discussed earlier in Figure 3.5, are unreliable for topology evolution. We find that by using topological derivatives we can successfully recover the shown examples as they can evaluate visibility changes in such regions. We provide more examples of shape recovery later in this chapter.

Secondary Visibility

We extend our formulation for TD to a rendering integral with secondary light bounces. We project a point $\mathbf{u}$ on the image plane to a shading point in the scene by $\mathbf{x} = \boldsymbol{\pi}^{-1}(\mathbf{u})$ (see the inline figure). We can integrate a scene function over the hemispherical domain around this shading point $\mathbf{x}$ and relay the differentials back to $\mathbf{u}$. We follow a similar construction as in the case of primary visibility (Result 3.3) and estimate TD at a point $\mathbf{x}' \in \Gamma$ acting as an occluder for the light path $\mathbf{u} \to \mathbf{x} \to \mathbf{x}'$. Let $g(\mathbf{x}')$ be the scene function along this path with the unperturbed surface and $g_B(\mathbf{x}')$ the scene function after the conical perturbation. The TD of the shading

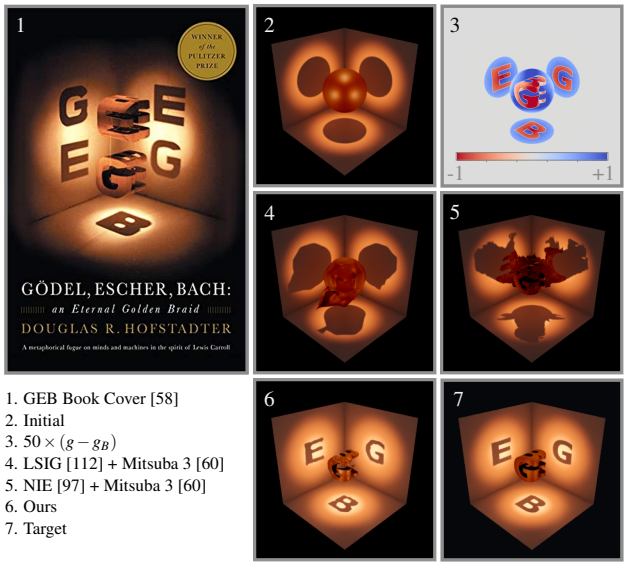


Figure 3.10. Shape ambigram from a single image. We reproduce the famous shape ambigram from the cover of *GEB* by Hofstader (1). Given a single image (7), we aim to recover a shape with complex shadow projections from a relatively simple shape (2). A differentiable path tracer like Mitsuba 3 [60] uses a form of shape derivative and struggles with visibility changes necessary in the interior regions. We regularize the gradients from [60] using two recent methods. One which works with triangle meshes [112] (4) and the other with level-sets [97] (5). Our method uses topological derivatives for secondary visibility and can successfully recover the shadows (6). In (3) we visualize the term $g - g_B$ at intialization.

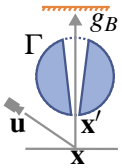


Figure 3.11. Secondary visibility. A point $\mathbf{u}$ on the image plane maps to a shading point $\mathbf{x} = \pi^{-1}(\mathbf{u})$; the scene function is integrated over the hemisphere around $\mathbf{x}$, with an occluder $\mathbf{x}'$ along the light path $\mathbf{u} \to \mathbf{x} \to \mathbf{x}'$.

functional around $\mathbf{x}$ with respect to a perturbation at $\mathbf{x}'$ is,

$$D_\tau(\mathbf{x}', \Gamma) = (g(\mathbf{x}') - g_B(\mathbf{x}')) \frac{\kappa(\mathbf{x}' - \mathbf{x})^t (\mathbf{x}' - \mathbf{x})}{(\mathbf{x}' - \mathbf{x})_z^3}. \tag{3.26}$$

We provide the full derivation later in this chapter. A surface evolution equation including a TD of this form can perforate through shapes along the secondary segments of light paths. To demonstrate the feasibility of this approach we propose a task of recovering 3D shape ambigrams from a single image. Inspired by the popular cover of *Gödel, Escher, Bach* [58], we construct a scene with an ambigram of letters `G`,`E` and `B` placed inside a box with shadow projections of the letters on different planes. As shown in Figure 3.10, differentiable path tracers such as Mitsuba 3 [60] cannot recover this shape with a genus-0 intialization. By adding a TD term to (3.25), we can deform the sphere such that more light can be accumulated in the regions with shadows.

3.6 Additional Derivations and Results

3.6.1 Topological Derivatives in 2D

We first outline the derivation for TD with respect to nucleating phases in the exterior of closed 2D curves.

Result 3.4. *Let Γ be a closed curve with Ω_Γ as its interior. For a point $\mathbf{x} \in \mathbb{R}^2 \setminus \Omega_\Gamma$ the topological derivative with respect to phase nucleation is,*

$$D_\tau(\mathbf{x}) = g_F(\mathbf{x}) - g_B(\mathbf{x}).$$

We follow the same derivation structure as in the case of hole nucleation. The main difference comes from how we define the notion of a perturbed shape. Consider a point $\mathbf{u} \in \mathscr{I}$ in the image plane, such that the corresponding point on the 2D plane $\mathbf{x} \in \mathbb{R}^2$ lies in the exterior of the given curve. We introduce a circular disk B_ε of radius ε centered at $\mathbf{x}$. The topological

derivative with respect to this perturbation is then,

$$D_\tau(\mathbf{x},\Omega_\Gamma) := \lim_{\varepsilon\to 0}\frac{I(\Omega_\Gamma\cup\overline{B_\varepsilon(\mathbf{x})})-I(\Omega_\Gamma)}{V(B_\varepsilon)}. \tag{3.27}$$

Note that the perturbed image functional is defined for the union of the original shape and the circular disk. The asymptotic expansion of (3.9) is,

$$I(\Omega_\Gamma\cup\overline{B_\varepsilon}) = I(\Omega_\Gamma)+V(B_\varepsilon)D_\tau(\mathbf{x},\Omega_\Gamma)+o(V(B_\varepsilon)). \tag{3.28}$$

We now construct a similar normal velocity as in the case of TD for hole nucleation. Let v be a continuous normal velocity such that $v=1$ on the boundary ∂B_ε of the added phase and $v=0$ on the original curve Γ. Intuitively, this function expands the radius of the circular disk by 1. Our goal is to measure the change in the functional I as we expand the nucleated phase and $\varepsilon\to 0$. We take the Gâteaux derivative of (3.10) in the direction v. By definition, we know:

$$dI(\Omega_\Gamma)v = 0. \tag{3.29}$$

From (3.10) and (3.29), we can redefine the TD as,

$$\begin{aligned} D_\tau(\mathbf{x},\Omega_\Gamma) &= \lim_{\varepsilon\to 0}\frac{1}{V'(B_\varepsilon)}dI(\Omega_\Gamma\cup\overline{B_\varepsilon})v \\ &= \lim_{\varepsilon\to 0}\frac{1}{V'(B_\varepsilon)}\int_{\Gamma\cup\partial B_\varepsilon}(g_F-g_B)v\,d\sigma \\ &= \lim_{\varepsilon\to 0}\frac{1}{V'(B_\varepsilon)}\int_{\partial B_\varepsilon}(g_F-g_B)\,d\sigma \\ &= g_F(\mathbf{x})-g_B(\mathbf{x}). \end{aligned} \tag{3.30}$$

The last step in the derivation is implied from: as $\varepsilon\to 0$, g_F and g_B correspond to $g_F(\mathbf{x})$ and $g_B(\mathbf{x})$, and $\int_{\partial B_\varepsilon}(g_F-g_B)\,d\sigma = 2\pi\varepsilon\times(g_F(\mathbf{x})-g_B(\mathbf{x}))$.

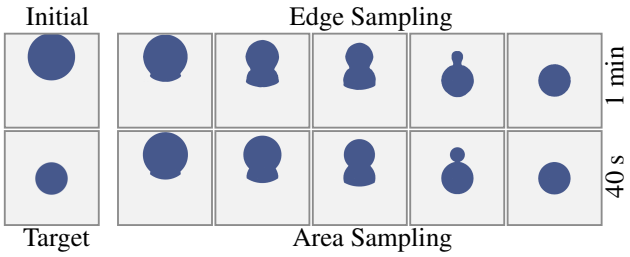


Figure 3.12. Area sampling v/s edge sampling. We can evolve curves using only area sampling (*bottom*) and get similar curve evolution as from a differentiable rasterizer [81] (*top*) which uses edge sampling. In practice, this leads to reduced optimization runtimes.

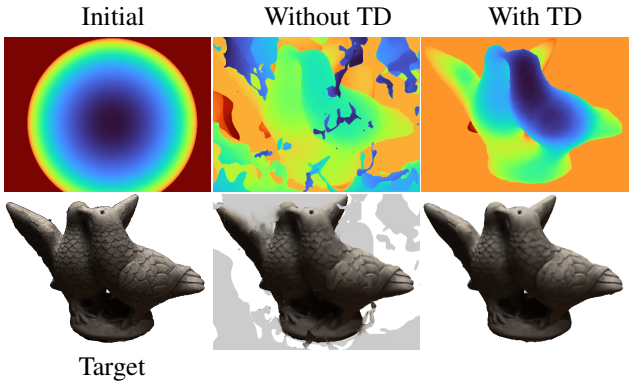


Figure 3.13. Preliminary results on 3D shape recovery with real data. We test our proposed theory on topological derivatives for multi-view 3D reconstruction using real data. Given 64 images for `scan 106` from DTU dataset [61], we reconstruct shape and reflectance starting from spherical initialization. On the top we show the depth map for the recovery and the textured mesh is shown on the bottom. We note that this result is still a proof-of-concept that demonstrates the potential of using topological derivatives in the presence of irregularities in real-world data.

Eliminating Edge Sampling

We show that the SD and TD for a closed curve is the same in their functional form. A level-set PDE which uses only shape derivatives has the evolution dynamics defined only on the curve. For numerical optimization, this requires explicit sampling of the boundaries as in [81]. Using the TD, the level-set PDE is defined everywhere on the image plane and we can evolve curves using area sampling. In practice, for complex shapes, this will lead to faster optimization. In Figure 3.12, we compare shape optimization using edge and area sampling. The area sampling example uses a single sample per pixel and only compares colors with the target at the centre of the pixel.

Generative Vector Graphics

We provide additional results for generating vector graphics using SDS [129] in Figure 3.14.

3.6.2 Initial Demonstration on Real Data

We test the feasibility of using topological derivatives with real data. Given a set of images, with a known background for each image, we reconstruct the object of interest and its reflectance. We use a neural reflectance function $f_\theta : \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^3$, which takes the position, normal and viewing direction as input and estimates the outgoing radiance. θ is the set of parameters for the neural network. Figure 3.13 shows the recovery using our method with an ablation on TD. Our initial results indicate that TD is useful for taking large steps to recover a coarse shape and resolve some of the shape/radiance ambiguities. Using only shape derivatives, we find the optimization to be biased towards optimizing reflectance parameters, which results in floating artifacts. We note that these results are preliminary and act as a proof of concept for using TD in the presence of non-idealities such as sensor noise, error in camera positions and distortion.

Curvature κ

The TD for 3D depends on the curvature of the perturbed surface, which is not the same as the curvature at a point $\mathbf{x}$ on the unperturbed surface. It depends on how we define the cone of perturbation. In practice, we find that we are able to achieve good recovery by assuming the curvature of the unperturbed shape to be constant across the *entire* surface. To understand this relevance of κ during optimization, we revisit the level-set PDE used for surface evolution:

$$\frac{\partial \phi}{\partial t} = -dI(\Gamma)|\nabla \phi| - \lambda D_\tau(\mathbf{x}, \Gamma)|\nabla \phi|. \tag{3.31}$$

Note that there is an additional factor λ in this PDE in comparison to the equation (24) in the main paper. We use λ as a weighting term to balance shape and topological derivatives during



Figure 3.14. Generating complex vector graphics using score-distillation [129] We show additional results for the task of reconstructing vector images using a non-standard reconstruction loss. We use score-distillation-sampling loss from [129] with a stable diffusion model [137]. The shape initialization is the same as shown in Figure 4 in the main paper. In the order of top to bottom and left to right, the following text-prompts are used for generation: `mouse`, `horse`, `penguin`, `eagle`, `mantis`, `pufferfish`, `cow`, `meerkat`, `rabbit`, `fennic fox`, `owl`, `shoebill`, `frog`, `koala`, `tiger`. We augment the prompts with "Frontal face of `<tag>`. Minimal line drawing. Trending on artstation. Plain white background. Black and white."

optimization. By assuming constant curvature, we can absorb κ into the weighting term λ.

Secondary Visibility

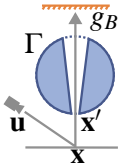


Figure 3.15. Secondary visibility. A point $\mathbf{u}$ on the image plane maps to a shading point $\mathbf{x} = \pi^{-1}(\mathbf{u})$; the scene function is integrated over the hemisphere around $\mathbf{x}$, with an occluder $\mathbf{x}'$ along the light path $\mathbf{u} \to \mathbf{x} \to \mathbf{x}'$.

Although our notation for topological derivatives follows the standard definition [147], for brevity, we assume that the TD is conditional to a given camera position and it's viewing direction. For primary visbility this is implicitly defined using the cone of perturbation where the apex is the camera's origin. This relationship between the camera position and the topological derivative is different in the case of secondary visibility. Here, the apex is at the shading point $\mathbf{x}$ which is the first intersection point for a ray originating from the camera. For this case, the definition of the scene functions g and g_B also differ. For primary visibility, the function g represents the radiance error along a ray originating at the camera center, whereas for secondary visibility, g corresponds to the radiance error for a light path that begins with $\mathbf{u} \to \mathbf{x} \to \mathbf{x}'$. Similarly the function g_B defines the error when the first occluder does not exist in case of primary visibility — and for secondary visibility, the error for path starting with $\mathbf{u} \to \mathbf{x} \to \mathbf{x}'$ but without the occluder corresponding to the secondary intersection. Apart from the differences in these definitions, the derivation for the TD for secondary visibility follows similarly from primary visibility. We assume a virtual camera centered at the shading point x and from the TD definition in Result 3 from the main manuscript, we arrive at this result,

$$D_\tau(\mathbf{x}', \Gamma) = (g(\mathbf{x}') - g_B(\mathbf{x}')) \frac{\kappa(\mathbf{x}' - \mathbf{x})^t (\mathbf{x}' - \mathbf{x})}{(\mathbf{x}' - \mathbf{x})_z^3}. \tag{3.32}$$

The change of terms from $\mathbf{x}$ to $\mathbf{x}' - \mathbf{x}$ is due to the repositioning of the camera from the origin to $\mathbf{x}$. To be more specific, $g(\mathbf{x}')$ can be expanded to $g(\mathbf{u} \to \mathbf{x} \to \mathbf{x}')$. In Figure 3.16 we show the

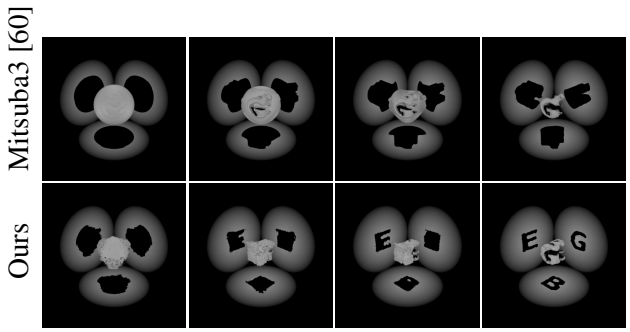


Figure 3.16. Reconstructing shape ambigrams We aid the GEB illustration (Figure 9) shown in the manuscript with mid-optimization renderings for NIE [97] + Mitsuba 3 [60] and our method. The recovery starts from a spherical initialization and progresses as shown here (from left to right).

optimization progression for the results shown in Figure 9 in the main manuscript.

3.7 Implementation Details

Image Vectorization

The image vectorization experiment uses a 128×128 2D grid to discretely reprent the level-set function ϕ. At initialization, ϕ is a signed distance function (SDF). During optimization, we do not enforce ϕ to be an SDF. The raster input is a 128×128 image upsampled to 512×512. At each iteration, we extract the 0-isocontours from the level-set function. The point-set for these contours are then used as closed bezier paths that we rasterize at 512×512 resolution. The scene functions $g_F(\mathbf{x})$ and $g_B(\mathbf{x})$ are defined as $|F(\mathbf{x}) - c(\mathbf{x})|$ and $|B(\mathbf{x}) - c(\mathbf{x})|$ respectively. $F(\mathbf{x})$ and $B(\mathbf{x})$ correspond to foreground and background colors at $\mathbf{x}$, and $c(\mathbf{x})$ is the color in the target raster image. We use a single sample per pixel and evaluate these functions at the center of the pixel. We use Adam as the optimizer and the learning rate for both shape derivative and topological derivative methods is 0.01.

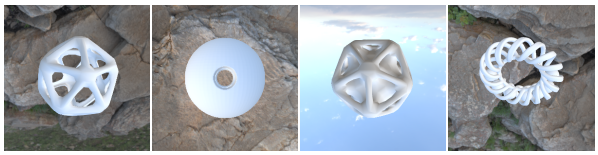


Figure 3.17. Samples from the set of images used for multi-view reconstruction. We use the `kloppenheim 06` known environment map by Greg Zaal [186].

Generating Vector Graphics

We use a 128×128 level-set grid. The score-distillation-sampling loss is from [129]:

$$\nabla \mathscr{L} = \mathbb{E}_{t,\varepsilon} \left[w(t)(\hat{\varepsilon}(z_t : y, t) - \varepsilon) \frac{\partial z}{\partial x} \right], \tag{3.33}$$

where t is the diffusion time step, w is a weighting term, z_t is the noisy latent variable, y is text-embeddings, $\hat{\varepsilon}$ is the estimated noise and ε is the ground-truth noise added to the latent variable. We use this gradient with our level-set PDE to optimize closed bezier curves given a text-prompt. The method using Li *et al.*'s [81] differentiable rasterizer is initalized with 128 randomly initialized disks with 12 control points. Our method is initialized with an SDF of a disk. The images are rasterized at 512×512 resolution. We use the learning rate scheduler from [159]. Our method is run for 2000 iterations and Li *et al.*'s method is run for 400 iterations. We find that the quality of output degradges significantly after 400 iterations with Li *et al.*'s method.

Multi-View 3D Reconstruction

Each example is recovered from a set of 100 images of 256×256 resolution, except for `helix`, which is rendered at 512×512. We use a level-set function ϕ defined on a 3D grid of $128 \times 128 \times 128$ resolution. Each shape is assumed to have a constant diffused BRDF. We use a known environment map [186] and use low-frequency illumination using spherical harmonics [132]. We finetune recovered meshes for each example after 1200 iterations and the total number of iterations is 2000. We employ a Laplacian regularizer for smoother recovery. At each step, we extract a mesh using Marching Cubes [91] at 128^3 resolution. We use nvdiffrast to

estimate the shading and edge visibility gradients [74]. The scene functions g and g_B correspond to L2 error between target - foreground and target - background. We use a single sample at the centre of each pixel to estimate the topological derivative. The weighting term as discussed in (3.31) is 0.002, which we reduce to 0 after 1200 iterations.

Secondary Visibility

We use a single image of 512×512 resolution to recover the shape. We use Mitsuba 3 [60] to estimate the shading and edge visibility gradients. To estimate the topological derivatives, we use a custom integrator for a two-bounce rendering model. Three spotlights are placed in the scene, one opposite to each of three orthogonal planes shown in Figure 3.16. We use emitter importance sampling for the integrator. The shape is extracted using marching cubes [91] at each iteration at 128^3 resolution. We use $\lambda = 0.001$.

Initial Experiments with Real Data

We use 64 images for `scan 106` from the DTU dataset [61]. Each image is downsampled to $300 \times 400 \times 3$. For a more controlled setup, we segment out the background and replace it with a constant color. We use a 3 layered, 256 neurons-wide MLP with ReLU activations as the neural BRDF. The input position to the network is mapped to fourier features [158]. We fine-tune the extracted mesh after 2000 iterations and the total number of iterations we use is 4000. The total optimization time is $\sim$ 1hr.

3.8 Discussion

As shown in the experiments (Figures 3.4, 3.9 and 3.10), topological derivatives (TDs) can be used to prompt visibility changes away from shape contours. The form of these derivatives resemble occupancy and segmentation loss functions [114, 185] that are frequently employed in shape reconstruction methods. Roughly put, topological derivative can be interpreted as a *soft* notion of occupancy loss and can be used when a segmentation mask of the final geometry is unavailable.

In comparison to volumetric-rendering based methods [103] the advantage of using TDs is efficient sampling and the ability to use more complex materials and light transport effects. As evident from the synthetic nature of the experiments, however, there is still a gap between the reconstruction quality achieved using TDs and volumetric methods. We anticipate a few significant challenges, that if resolved could reduce the gap. *First*, balancing the shape and topological derivative terms in Eq. 3.25 for shape evolution in 3D is not straightforward. TDs are not always necessary, especially when the optimization has reached a point where the topology of the shape is the same as that of the target. *Second*, the assumption of constant curvature (in Eq. 3.23) could lead to unexpected behavior and its effect needs a thorough inquiry. *Third*, we do not derive TDs for phase nucleation in 3D. In 2D, a sample in the pixel space corresponds to a single sample on the canvas. As a result, the term $(g - g_B)$ in Equation 3.13 relates to a single point in the shape's exterior. For 3D, in empty space, a pixel-sample corresponds to a ray in the direction of that sample. This makes the problem of phase nucleation significantly more challenging.

Finally, joint optimization of geometry and other parameters such color, material properties and lighting is not trivial and poses additional challenges. As a preliminary step, below, we show an experiment with the task of recovering both geometry and color for 2D vectorization.



Figure 3.18. A preliminary experiment jointly recovering geometry and color for 2D vectorization.

Chapter 3, in full, is a reprint of the material as it appears in "A Theory of Topological Derivatives for Inverse Rendering of Geometry", Ishit Mehta, Manmohan Chandraker, Ravi Ramamoorthi, International Conference on Computer Vision (ICCV), 2023. The dissertation author was the primary investigator and author of this paper.

4 | Locally Orderless Images for Optimization in Differentiable Rendering

The previous two chapters built a representation and the gradients that drive it. Both assume that comparing a rendered image with a reference yields a useful gradient in the first place. For many scene parameters it does not, because a pixel-wise loss reacts only where the two images already overlap. This chapter turns to the objective itself and replaces the pixel-wise comparison with one based on locally orderless images, which widens the basin of attraction for the optimizations used throughout this dissertation. With the representation, the gradients, and the objective now in place, Chapter 5 draws the three together.

4.1 Introduction

Much of the recent work addresses inverse rendering with analysis-by-synthesis — start with an initial guess of scene parameters, render an image using a differentiable renderer [74, 79, 89, 60, 134], compare it with a given photograph, estimate gradients, and update the parameters iteratively. While the success of gradient-based optimization in machine learning validates this approach, significant challenges remain. Despite having algorithms that accurately estimate gradients for the physics of image formation, optimization hurdles such as local minima, noisy loss landscapes, and the search for good initialization and parameterization all still persist [101].

For many inverse problems, the ideal optimization trajectory requires long-range motion of pixels and image features. Consider the example shown in Fig. 4.1, where a shiny ball is lit by an unknown light source that we wish to recover from a given image. A primary effect of optimizing the light position here is on the motion of the specular highlight on the ball. Since

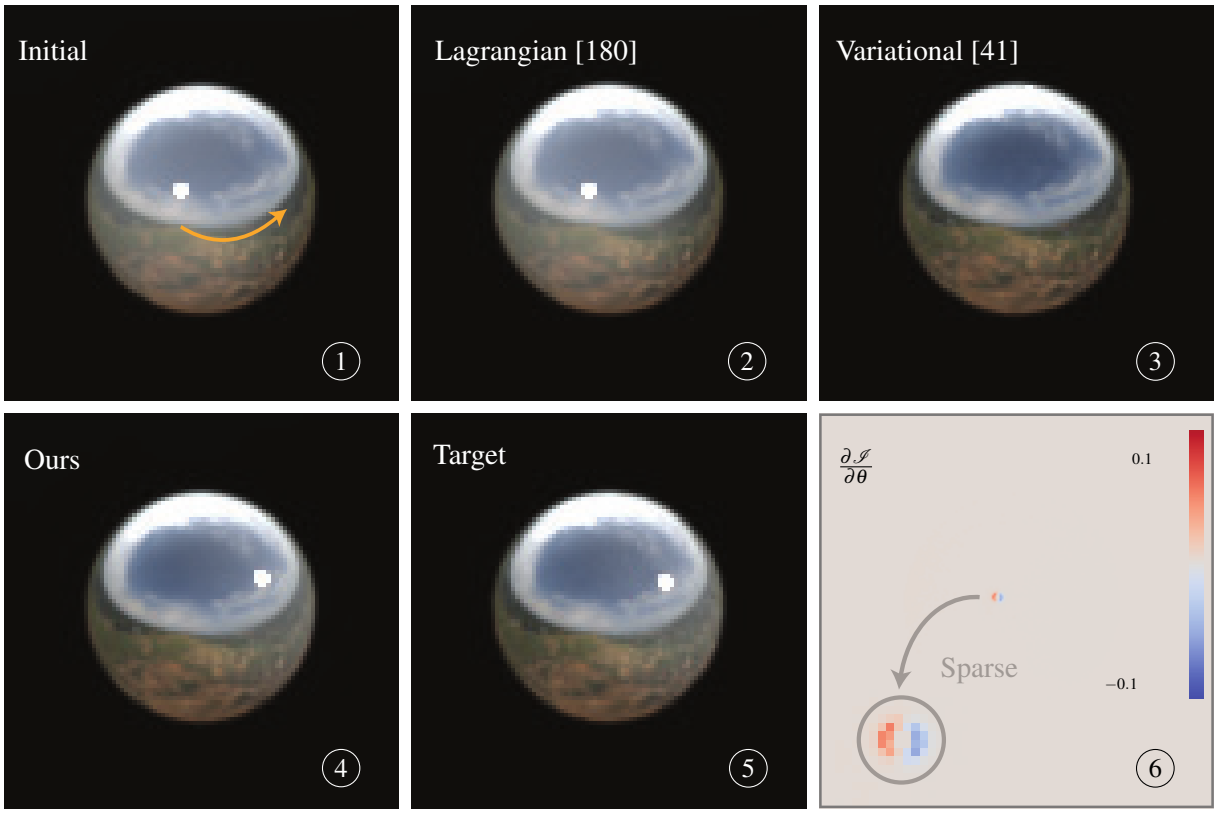


Figure 4.1. Image gradients are sparse with respect to optimization parameters that induce motion in the image space ⑥. We show an inverse problem with the goal of recovering the position (θ) of a distant light source from a synthetic image of a shiny ball, *i.e.*① to ⑤. Existing methods compute proxy gradients such as: Lagrangian derivatives [180], which track only primary-ray intersections, or variational derivatives [41], which can be prone to local minima. Our method uses standard RGB gradients and uses an inverse rendering objective that matches locally orderless images.

image gradients are sparse (Fig. 4.1 ⑥) for such parameters, the optimization landscape is rife with local minima and plateau regions [98, 41]. Existing methods address the gradient-sparsity problem with proxy gradients such as topological derivatives [98], lagrangian derivatives [180], and variational derivatives [41] — all of which are either expensive to compute [41, 180], only work with implicit geometry [98], or are restricted to primary light-transport effects [98, 180]. We propose a complementary approach that is compatible with RGB gradients from a differentiable renderer [80, 60, 74] — which we demonstrate to be expressive for a variety of inverse problems that have multiple locally-optimal solutions (as in Fig. 4.1).

Our method builds on scale-space matching [176], where signals are matched at multiple resolutions to measure similarity. In differentiable rendering, multi-scale matching has received limited attention, with a few works [41, 180] noting its unreliability. In this work, we observe that this unreliability stems from the standard approach of using multi-resolution images or Gaussian Pyramids [1], which average out local appearance and geometric details. Instead, we repurpose the inverse rendering objective to match local histograms that preserve the *full* distribution of intensity values within a neighborhood. Originally proposed by Griffin [48] to address imprecision in radiance measurements, and later extended by Koenderink and van Doorn [72] for reasoning about image topology — we postulate that the idea of viewing images as a family of histograms might be relevant in the seemingly unrelated field of inverse rendering.

Formally, we render images in three distinct scale spaces [72]: 1) the inner (σ) scale, describing the effective resolution of the image (Fig. 4.2 *top*), 2) the tonal (β) scale, which quantifies the imprecision in measured radiance at a location ($\mathbf{x}$), and 3) the extent (α) scale, which relates to the spatial region over which a histogram is computed (Fig. 4.2 *bottom*). The resulting representation, $\mathscr{P}(\mathbf{x}, \alpha, \beta, \sigma)$, is considered *locally orderless* [72] as the histograms eliminate any spatial variations in intensity, preserving only their distribution. We find that matching images within this locally orderless structure extends gradient support (Fig. 4.3) and is effective in solving inverse rendering problems that require long-range motion of image features (Fig. 4.7). Our method is straightforward to implement, requires no modifications to a

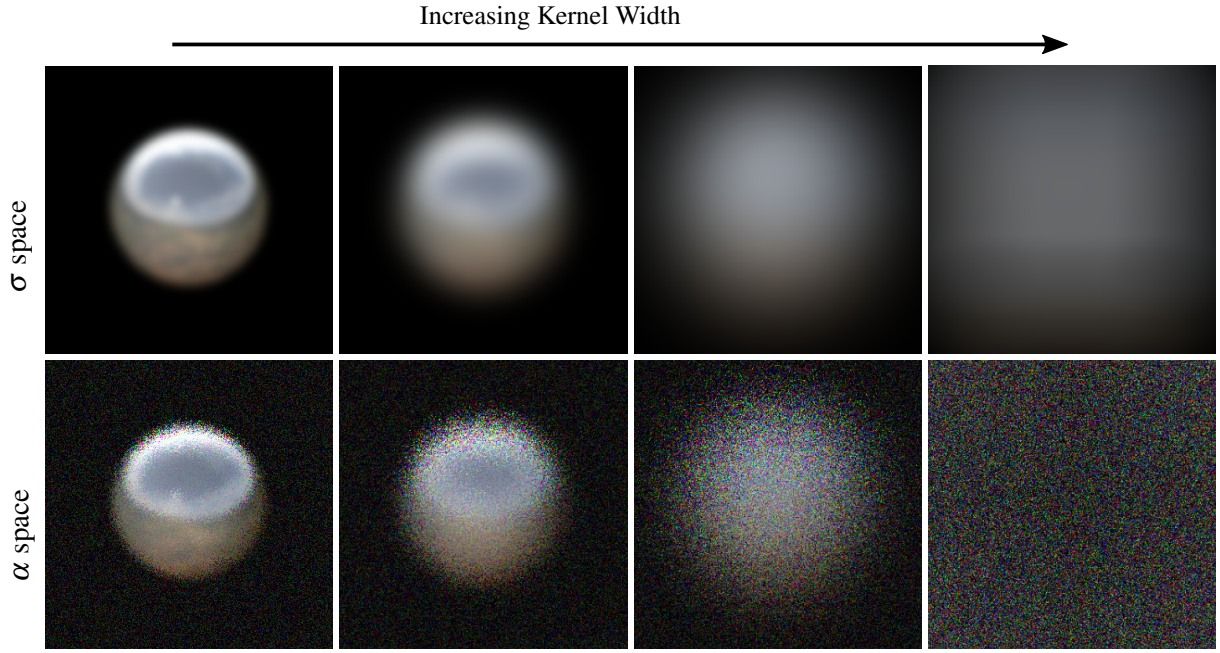


Figure 4.2. Inner and Extent scale spaces. The image representation $\mathscr{P}(\mathbf{x}, k, \alpha, \beta, \sigma)$, is composed of three distinct histogram-valued scale spaces. The σ-space (*top*) controls the effective resolution of the image and the α-space (*bottom*) defines the spatial extent of histogram integration. The rendered images shown here are intensities sampled as $\mathscr{I}(\mathbf{x}) = k, k \sim \mathscr{P}(\mathbf{x}, k, \theta, \alpha, \beta, \sigma)$ for given kernel parameters α and σ, and bin width β. We recover inverse rendering parameters θ by matching these locally orderless structures for rendered and reference images.

differentiable renderer, is robust under noise, and works with arbitrary geometry representations and complex light-transport effects.

Overall, our method achieves good local minima for diverse optimization problems arising in differentiable rendering (§ 4.4). Results on inverse vectorization (§ 4.4.1), path tracing (§ 4.4.2) and rasterization (§ 4.4.4) validate that our method can reliably use standard RGB gradients, while methods that require computing additional gradients may fail. Lastly, we show that our method is also compatible with variational optimization on scenes with complex-light transport effects (§ 4.4.3) and real data (Fig. 4.10).

4.2 Related Work

The goal of differentiable rendering is to compute derivatives of the rendering integral [66] with respect to scene parameters. Estimating the rendering derivative has a long history in vision and graphics [8, 133, 173], and proves useful for accurate shape estimation [109], inverse scattering [45], material acquisition [69, 22], and various vision tasks [14]. While early works focus on deriving gradients for specific applications, more recent works by Loper and Black [89], Li *et al.* [79], Jakob *et al.* [60], Laine *et al.* [74], and Ravi *et al.* [134] propose general-purpose differentiable renderers. These renderers come with different tradeoffs, however we focus specifically on maximizing the utility of the estimated gradients for solving inverse problems.

In contrast to neural networks, where parameter gradients can have global support [47], rendering gradients are more local, especially for parameters that define visibility in the scene. Optimization using gradient descent is difficult in this case, as shown in [98, 180, 41]. Existing methods extend gradient support either by using alternate geometry gradients [98, 180] or through the notion of *differentiating through blurring* — either by blurring discontinuities [86, 184, 172, 42] or blurring the parameter space [41] as a form of stochastic approximation [38] of the gradient. Our approach operates entirely in image space and uses scale-space matching [176]. While this makes our method conceptually orthogonal to other methods, we show that they are

complementary (§ 4.4.3).

Scale-space matching remains relatively underexplored in differentiable rendering. While Li *et al.* [79] use Gaussian Pyramids to avoid local minima when optimizing geometry primitives, and Vicini *et al.* [168] apply them for recovering signed distance functions, Xing *et al.* [180] show that using multi-resolution images is unreliable. In § 4.3, we illustrate why using image pyramids may fail. Our approach uses three distinct scale spaces implemented as Locally Orderless Images (LOIs), as proposed by Koenderink and van Doorn [72]. Unlike Gaussian pyramids that only match mean intensities within neighborhoods, our method matches entire distributions, leading to better recovery. LOIs have proven effective for various vision tasks [166], including image retrieval [78], object tracking [119], and non-linear filtering [165, 164].

4.3 Method

Consider an image $\mathscr{I}(\mathbf{x};\theta) : \mathbb{R}^2 \mapsto \mathbb{R}$ as a set of direct or indirect observations of parameters θ that model the geometry and the appearance of a scene. Starting from an unknown set of parameters θ, the goal is to recover an optimal set such that the rendered image matches identically to a given reference. We use a differentiable renderer to compute derivatives $\frac{\partial \mathscr{I}}{\partial \theta}$ and minimize an error function $\mathscr{E}$ through gradient descent. As shown in the example in Fig. 4.1, image gradients can be sparse and ineffective for inverse problems [98, 180, 41, 79].

As an illustration, consider the 1D toy example shown in Fig. 4.3 A reference image shows a disk for which we wish to recover its position θ along the horizontal axis. For any initialization, the image gradients are sparsely localized at the disk's silhouette. In the very likely case of no overlap between the initial and the target disks, $\frac{\partial \mathscr{E}}{\partial \theta} = 0$, providing no signal to update the initial choice of θ. Even with some overlap between the two disks, $\frac{\partial \mathscr{E}}{\partial \theta}$ is non-zero but still sparse, making the optimization sensitive to noise.

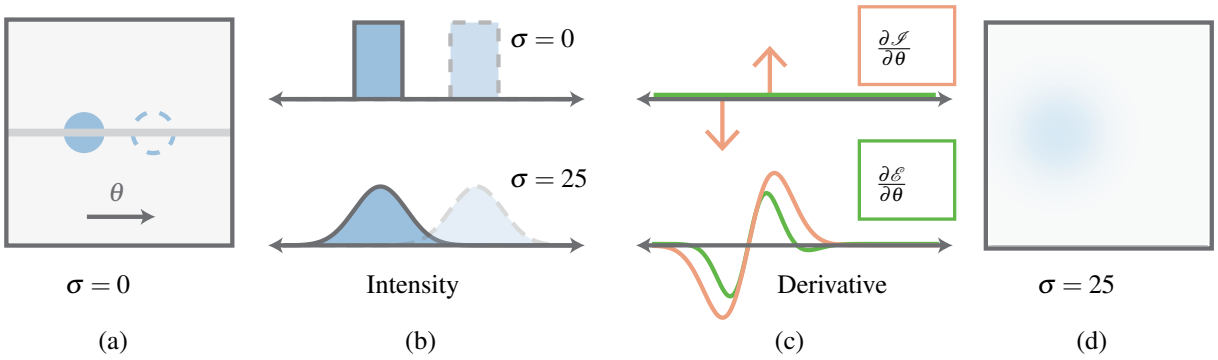


Figure 4.3. Scale-space matching extends gradient support. Given an image (a) of a disk we recover its position θ on the horizontal axis. At stationary resolution ($\sigma = 0$), the initial and target (dotted) disks do not overlap, as shown in the corresponding 1D signals in (b). The image gradient $\frac{\partial \mathscr{I}}{\partial \theta}$ is sparse (orange) and is non-zero only at the boundaries of the disk (c-*top*). The error gradient $\frac{\partial \mathscr{E}}{\partial \theta}$ is zero everywhere (green) and the optimization is stuck in a local minimum. When matching at coarser scales (d), the gradients are no longer sparse (c-*bottom*), leading to optimal recovery.

Inner Scale

To increase gradient support, we render images at different resolutions [176] and match them in scale space [175]. We apply a linear filter with a progressively increasing kernel width $\sigma \in \mathbb{R}_+$ to images rendered at their stationary resolution ($\sigma = 0$):

$$\mathscr{I}(\mathbf{x}; \theta, \sigma) = (G * I)(\mathbf{x}; \sigma), \tag{4.1}$$

where $*$ is the convolution operator and G is the Gaussian aperture function that defines the *inner scale*:

$$G(\mathbf{x}; \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\mathbf{x} \cdot \mathbf{x}}{2\sigma^2}\right), \quad \sigma > 0. \tag{4.2}$$

Let us revisit the example in Fig. 4.3. At coarser scales ($\sigma > 0$), we find that the image gradients have an extended support with a strong enough signal for convergence as the disks turn into larger overlapping blobs. This example reveals a key insight: while gradients are inherently local, blurring images with a scale-space kernel increases their support and simplifies

the problem of alignment in image space. Note that this idea of *gradient diffusion* is similar to variational optimization [41, 152, 155], where a blur kernel is alternatively used in the *parameter space* to achieve a similar effect of extending gradient support and enabling long-range matching in image space.

Scale-space matching performs well for simple, controlled problems, but it remains unreliable in more realistic scenarios. As shown in recent work [180, 41], for inverse problems with complex geometry and light-transport effects, blurring and multi-scale strategies are not robust. We identify two primary reasons for this. *First*, blurring has an averaging effect that changes the local appearance — this makes matching image features more difficult. *Second*, blurring suppresses high-frequency details. With minor tweaks to the previous toy example, such as including image noise or multiple objects (Fig. 4.4 and 4.5), we find that inner-scale matching still suffers from local minima problems. Our method uses two additional scale parameters to achieve optimal recovery.

Tonal Scale

We use scale-imprecision space [48] and relax the assumption of using images as real-valued functions. Instead, at each pixel location, we model a radiance distribution that captures the uncertainty in the measurements. Even in ideal settings, this is reasonable, as pixels integrate light over a non-zero area and measurements have finite precision [128]. In the context of inverse rendering, images can be noisy due to Monte Carlo integration [66] or external factors such as sensors and lenses. Explicitly modeling uncertainty in estimated parameters can lead to better recovery, as also observed in [192]. Our method models the uncertainty in radiance estimates using 1D kernels with bandwidth $\beta \in \mathbb{R}_+$. The result is a family of spatially-varying probability distributions $\mathscr{P}$ defined on the image plane:

$$\mathscr{P}(\mathbf{x}, k; \theta, \sigma, \beta) = \frac{1}{\sqrt{2\pi\beta^2}} \exp\left(-\frac{(k - \mathscr{I}(\mathbf{x}; \theta, \sigma))^2}{2\beta^2}\right), \tag{4.3}$$

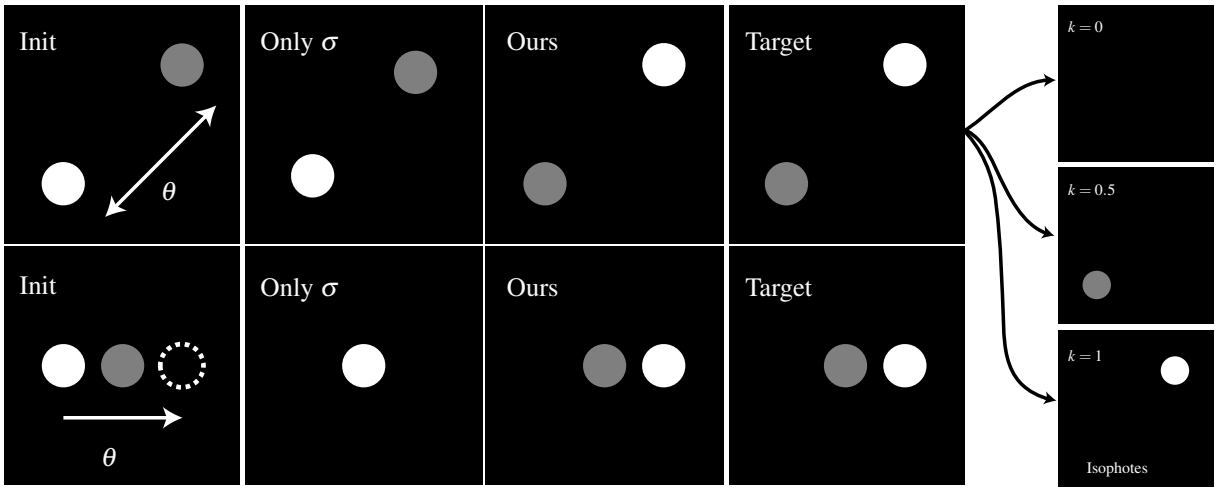


Figure 4.4. Tonal Separation. Shown are two (a-*top* and a-*bottom*) 1D inverse problems where we recover disk positions (θ) from images (*left*). Image matching within σ-space measures only the errors in the mean of the intensity distributions at each scale. In inverse settings that involve multiple objects with different appearances, this approach is likely to get stuck in a local minimum (a-*center-left*). The α-space integration kernels are intensity-aware and treat images as sets of distinct equal-intensity isophotes (b). When images are matched in all three scale spaces, the optimization is less prone to getting stuck in local minima (a-*center-right*).

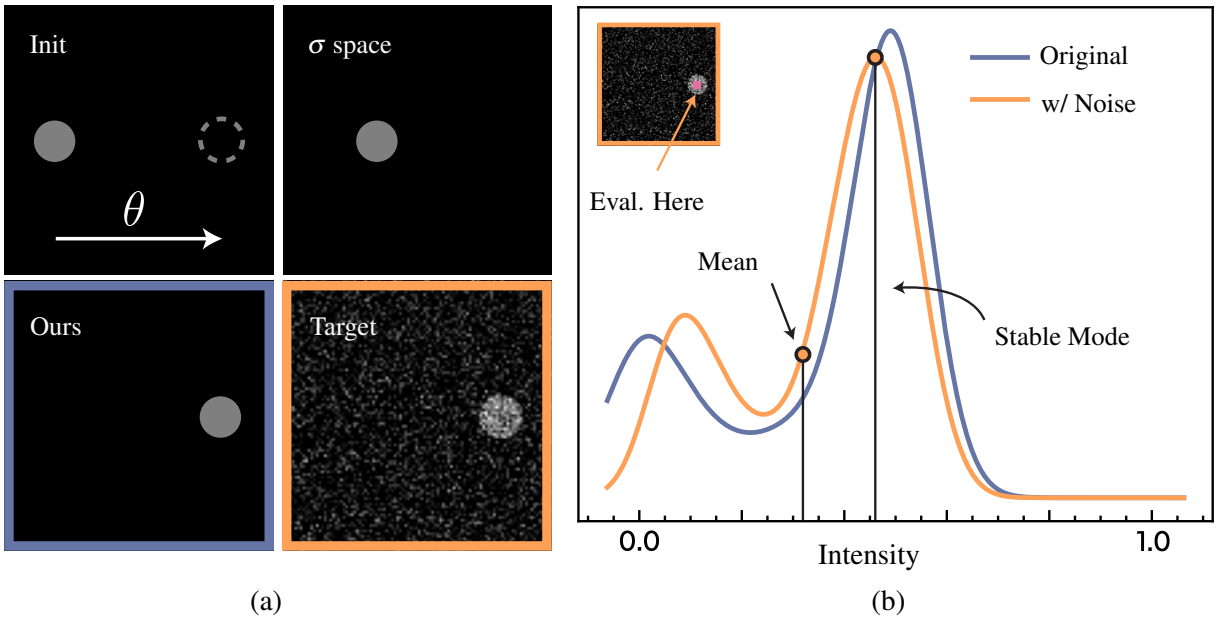


Figure 4.5. Histogram matching is less sensitive to noise. To recover the position (θ) of a circular disk from a noisy reference image (a-*bottom-right*), methods that match images only at their stationary resolution or in σ-scale space fail — as they overlook imprecision and uncertainty in radiance measurements. Our method uses a tonal parameter (β) to account for intensity uncertainty and an extent scale-space to preserve the distribution modes at coarser scales (b), leading to optimal recovery of θ.

where β is the scale in intensity domain and controls the tonal resolution, and $k \in [0,1]$ is intensity. For a given k, $\mathscr{P}(\mathbf{x},k)$ is a soft *isophote* that measures the probability of intensity being k at $\mathbf{x}$. This approach provides the benefit of *tonal separation*, where each isophote is treated as a distinct image, leading to improved separation both in visual features and image gradients. The inverse problem in Fig. 4.4 illustrates the benefit of using a tonal scale. In practice, we use β as the width of the bins used for discretizing $\mathscr{P}$ as histograms.

Extent Scale

To capture spatial relationships between the local distributions $\mathscr{P}$, we can extend this representation further using the idea of locally orderless images [72]. Similar to integrating image intensity over a spatial extent as in σ scale space, we use an aperture function A that

integrates the locally-defined histograms as follows:

$$\mathscr{H}(\mathbf{x},k;\theta,\sigma,\beta,\alpha) = \int_{\mathbb{R}^2} A(\mathbf{x}-\mathbf{y};\alpha)\mathscr{P}(\mathbf{y},k;\theta,\sigma,\beta)\, d\mathbf{y}, \tag{4.4}$$

where A is similar to the kernel defined in Eq. 4.2. Unlike conventional scale-space blurring which operates directly on radiance values, Eq. 4.4 blurs histogram contributions. This distinction enables the method to preserve the effective *modes* of the radiance distribution across scales — thereby retaining both the appearance and geometric characteristics of the image even at coarser resolutions. Consequently, our method is well-suited to handle perturbations in the reference images as the modes of the intensity distribution remain stable under noise [165]. See Fig. 4.5 for an illustrative example.

Histogram Matching

Equipped with three separate scale spaces, defined using the parameters α, β and σ, we can now pose the inverse rendering objective as minimizing the difference between the distributions of rendered ($\mathscr{H}'$) and reference ($\mathscr{H}^{\text{gt}}$) images. Matching histograms and computing related distance metrics is a well-studied task in graphics and optimal transport literature [125, 127]. We use the Wasserstein distance which has a closed form for 1D distributions. At given scales α, β and σ, the distance between two histograms is estimated as the summation of point-wise errors between their cumulative distribution functions (cdf):

$$\begin{aligned} \mathscr{E}(\theta,\alpha,\beta,\sigma) = \int_{\mathbb{R}^2}\int_0^1 [&\text{cdf}_{\mathscr{H}'}(\mathbf{x},k;\theta,\alpha,\beta,\sigma) \\ &-\text{cdf}_{\mathscr{H}^{\text{gt}}}(\mathbf{x},k;\alpha,\beta,\sigma)]^{1/p}\; dk d\mathbf{x}, \\ \text{where,}\quad \text{cdf}_{\mathscr{H}}(\mathbf{x},k;\cdot) &= \int_0^k \mathscr{H}(\mathbf{x},j;\cdot)\, dj \end{aligned} \tag{4.5}$$

We compute the total error as the summation over all the scales, *i.e.* $\mathscr{E}_{\text{total}}(\theta) = \sum_{\alpha,\beta,\sigma} \mathscr{E}(\theta,\alpha,\beta,\sigma)$. We use $p = 1$ in our experiments.

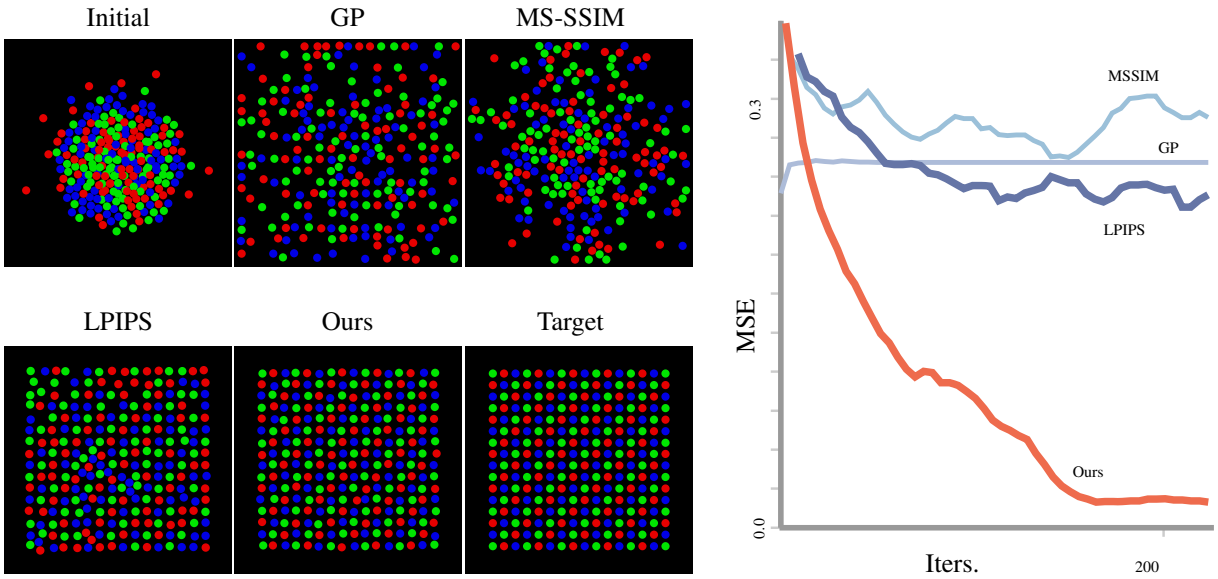


Figure 4.6. Comparisons with multi-scale methods. Given an image we optimize positions of 256 disks starting from random intialization. We use diffvg [80] to compute RGB gradients. Our method recovers the closest optimal arrangement. The error plot on the right shows mean-squared error (MSE) between rendered and target images *vs.*iterations.

4.4 Results

We evaluate our method using three differentiable renderers: diffvg [80] for vector graphics, a path tracer [60], and a rasterizer [74]. Our experiments are designed for three goals: First, we compare our method against other multi-scale approaches like Gaussian Pyramids [1] and MS-SSIM [171] (§ 4.4.1). Second, we evaluate how parameter-space blurring (PRDPT [41]) performs compared to scale-space matching (§ 4.4.2). Third, we test the reliability of RGB gradients versus proxy gradients like RGBXY [180]. We also show how our method complements parameter-space blurring through two example scenes (§ 4.4.3), including one with real data.

4.4.1 Differentiable Vectorization

Multi-scale feature matching is widely used in vision and graphics for various tasks. Three prominent techniques include: (a) the Gaussian pyramid [1], which approximates the σ-scale space and matches the mean of intensity distributions across scales; (b) multi-scale

Table 4.1. Quantitative comparisons for 2D. We recover positions of $n = [4, 16, 32, 64, 256]$ disks with *known* color from a reference 128×128 image. Image gradients are computed using diffvg [80] and sparsely defined along the silhouettes of the disks [98]. Compared to other scale-space approaches, we find our method to be best suited for this task.

n Disks	4		16		32		64		256	
Method	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
GP [1]	25.05	0.97	19.83	0.91	16.83	0.83	15.34	0.76	9.60	0.34
MS-SSIM [171]	27.44	0.98	21.41	0.94	17.93	0.87	15.42	0.78	9.83	0.35
LPIPS [189]	27.79	0.98	20.94	0.93	17.88	0.86	17.04	0.82	15.2	0.76
Ours	30.40	0.99	33.55	0.99	27.43	0.97	28.23	0.96	21.57	0.90

structural similarity (MS-SSIM) [171], which matches both the mean and covariance of the distributions; and (c) LPIPS [189], which uses a hierarchy of features from a deep neural network to measure perceptual similarity. Our approach, using three distinct scale spaces, is similar to these methods but is specifically designed for inverse rendering problems.

We design a test benchmark to evaluate multi-scale methods. Similar to previous experiments, the objective is to recover the positions of 2D disks with known colors from a given image. This is a representative task for inverse rendering problems that require long-range feature matching with sparse gradient information. The benchmark includes five problem sets, with difficulty determined by the number of disks in each image. Note that this a much more difficult task than optimizing color, since appearance gradients are not as sparse as geometry gradients [98]. Optimization parameters are randomly initialized, while target positions form a uniform grid. Differentiable vectorization [80] is used to compute image gradients. For other multi-scale approaches, we measure the L2 error between features at different scales, while our method uses Wasserstein distance (Eq. 4.5) to compare histograms. The reference images are 128×128 at stationary resolution and we choose $\alpha = [1, 5, 15], \sigma = [1, 5, 15, 45]$ and $\beta = 0.125$ as the scale space parameters. For consistency, we perform 10 runs, each with a different initialization, and report mean PSNR and SSIM in Table 4.1. The optimized images for $n = 256$ are shown in Fig. 4.6 along with the corresponding error plot.

Table 4.2. We evaluate our method and PRDPT [41], which employs parameter-space blurring, across five synthetic scenes of varying difficulty. We also present results using other multi-scale baselines. Our method recovers the optimal parameter configuration for all scenes, despite relying on sparsely defined RGB gradients.

	Shadow		*Shadow Mini*		*Caustics and Lights*		*Sort*		*Envlight*	
	PSNR	MAE	PSNR	MAE	PSNR	MAE	PSNR	MAE	PSNR	MAE
PRDPT [41]	31.77	0.0035	27.74	6.0147	17.09	4.8801	22.25	0.9201	31.87	0.7543
Ours	54.33	0.0034	54.73	0.0033	32.58	0.3475	25.23	0.4855	36.17	0.0775
Mitsuba [60] (GP)	24.44	1.1349	24.06	2.3491	21.02	0.7317	23.55	0.5189	28.73	0.8072
Mitsuba [60] (MSSSIM)	24.35	0.9320	24.34	0.9061	10.30	86.2286	18.91	0.7742	30.26	0.6703

4.4.2 Differentiable Path Tracing

Following the evaluation framework in [41], we test our method on five scenes. The goal here is to evaluate methods that use the notion of scale either in image space or parameter space (*i.e.*variational optimization). We use Mitsuba 3 [60] as the default renderer with the `prb_reparam` integrator [169]. Quantitative results evaluating visual appearance (PSNR) and parameter recovery (MAE) are in Table 4.2. We use the same kernel parameters, $\alpha = [1,5,15,45], \beta = 0.125$, and $\sigma = [1,5]$ for all the scenes. All images are rendered at 256×192 resolution. We use Adam with a learning rate of 10^{-2} for all methods.

Shadow

We use the `Shadow` scene from [41]. Given an image, we recover the position of a sphere that is not visible in the view frustum but projects a shadow onto a visible plane. This is a representative example for when RGB gradients perform poorly [41]. Using gradients from Mitsuba 3 [60], our method recovers the position of the sphere. Qualitative results for our method and variational optimization are shown in Fig. 4.7. While variational optimization is able to recover the target shadow, other multi-scale methods get stuck in local minima.

Shadow Mini

We modify the previous scene with two changes: (a) an initialization that is further away from the target, and (b) changing the scale of the sphere and hence the area of the shadow. Variational optimization struggles in such scenes. It achieves blurring in parameter space through

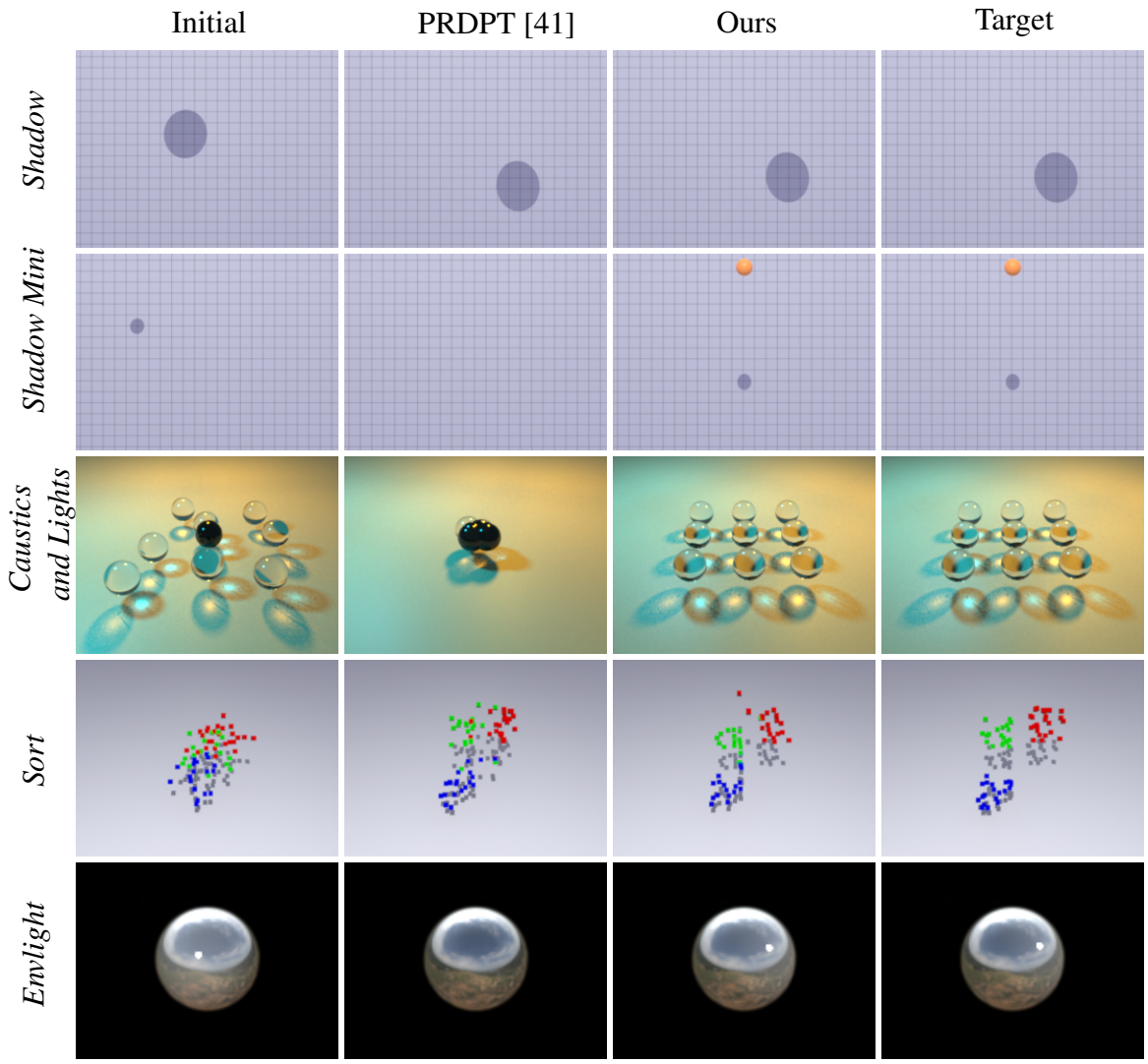
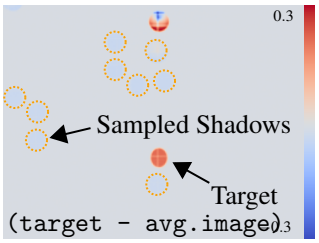


Figure 4.7. Quantitative comparisons with parameter-space blurring. We show five separate scenes (one in each row) with varying degrees of difficulty. Starting from a random initialization (left-most column), the goal is to recover the positions of different primitives so that the rendered image matches the reference (right-most column). Compared to parameter-space blurring (PRDPT [41]), we find our method that uses image-space local histogram matching is more suitable for these tasks. For quantitative comparisons, refer to Table 4.2.



Why does PRDPT [41] fail?

Figure 4.8. Why PRDPT fails on Shadow Mini. The inset plots the difference between the target image and the variational (average) estimate over 10 samples. The sampled shadows do not overlap with the target shadow, so variational optimization is driven away from the correct sphere position.

multiple samples that correspond to images with perturbed positions of the sphere. Ideally, for this approach to work, the shadows in the sampled images require some overlap with the target in the image space to update the parameters reliably. Since that is unlikely to happen (see inset) with the small shadow footprint in the reference, the optimization converges to the sphere being pushed out of the scene. The inset figure plots the difference between the target image and the variational estimate (avg. image) using 10 samples. As mentioned, shadows from the variational samples do not overlap with the target shadow. We test with different kernel parameters and sampling rates (τ in [41]), all of which perform similarly on this scene. Since our method operates directly in image space, it reliably recovers the reference independent of the initial position of the shadows. Other multi-scale methods converge at different local minima.

Caustics and Lights

This scene requires optimization in two separate domains: (a) recovering the positions of glass spheres, and (b) adjusting lights that are not directly visible in the view frustum. We normalize the range of both domains within $[-5,5]$. This task is challenging because, at initialization, the caustic patterns do not overlap with the reference in either appearance or position. We observe that matching histograms is particularly suited for this task due to the presence of Monte Carlo noise in forward rendering and high-frequency caustics patterns.

Sort

In the `Sort` scene [41], the objective is to recover the positions of 65 differently colored primitives from an image. This involves optimizing the x and z translation coordinates of these primitives, resulting in a 130-dimensional optimization problem. Since optimization in this scene is highly sensitive to the choice of initialization, we perform 20 different runs with varying random seeds and report the average results in Table 4.2. Comparisons for one of the runs are shown in Fig. 4.7. Since stochastic gradients have high variance for high dimensional problems [17], variational optimization is not ideal for such scenes.

Envlight

In this scene, we recover the position of a distant light source from an image of a reflective sphere rendered within a high-frequency environment. The light source is not directly visible, casting only a small specular highlight on the sphere, creating an optimization landscape with multiple local minima. Similar to the `Shadow Mini` scene, PRDPT [41] is stuck in a local minimum, as one of the sampled images need the highlight to be exactly aligned with the target. Our method, despite using sparse RGB gradients, successfully recovers the optimal configuration. Using image pyramids fails at this task as the high-frequencies and modes of intensity distribution are lost at coarser scales.

4.4.3 Variational Optimization

In the following two scenes, we demonstrate cases where RGB gradients alone are not reliable, and show how combining our method with variational derivatives [41] can achieve optimal recovery.

Kaleidoscope

As shown in [178], RGB gradients are unstable and inefficient for scenes involving complex light transport. Consider the virtual kaleidoscope scene shown in Fig. 4.9. Our goal is to recover both the kaleidoscope configuration and the positions of light-reflecting particles

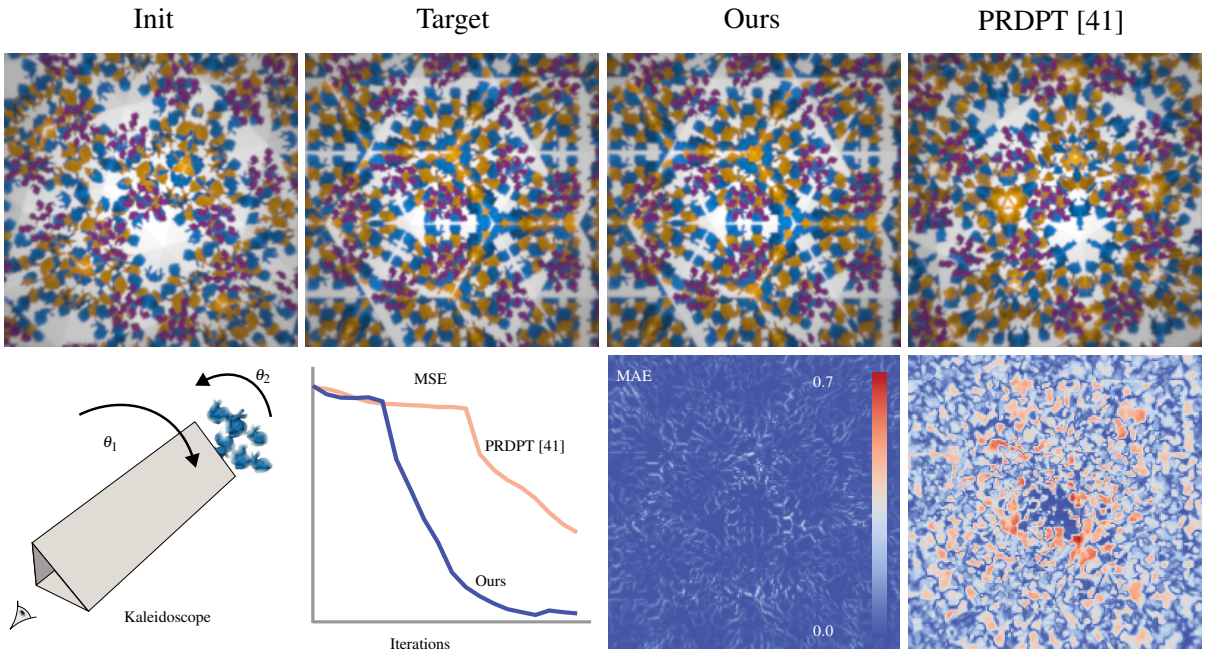


Figure 4.9. Variational Optimization with Locally Orderless Images. Our method works well with other optimization techniques like variational optimization. Given an image of a pattern inside a kaleidoscope, we recover the configuration of the kaleidoscope (θ_1) and the particles (θ_2) reflecting light through it. Using variational derivatives and matching images at their stationary resolution leads to sub-optimal recovery. By matching local image histograms (Ours), we successfully recover the target configuration.

Table 4.3. Quantitative results with differentiable rasterization. We evaluate our method using gradients (∇) from Nvdiffrast along with other multi-scale methods on test scenes provided in [180]. We show that our method using RGB gradients can reliably recover optimal solutions for low-dimensional problems (such as Translation and Rotation) and is ideal for higher dimensional problems (such as Camera Pose or Material recovery). Each image is rendered at 128×128. We use the scale parameters $\alpha = [1, 5, 15, 45], \sigma = [0, 5]$ and $\beta = 0.03125$.

	Parameters	Translation		Rotation		Shape		Trans. + Rot.		Camera Pose		Material		Env. Map	
	Num. views	6		6		1		4		1		6		6	
Method	∇ type	PSNR	MAE	PSNR	MAE	PSNR	MAE	PSNR	MAE	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Xing *et al.* [180]	RGBXY	63.62	0.00	26.66	0.45	30.75	0.30	30.23	0.68	23.50	0.92	33.01	0.96	32.43	0.91
Ours	RGB	51.77	0.00	30.73	0.33	28.22	0.54	36.23	0.35	27.22	0.96	49.07	1.00	55.22	1.00
Nvdiffrast [74] (GP [1])	RGB	31.16	0.18	22.40	0.49	25.55	0.59	19.53	2.11	14.82	0.73	42.48	0.99	47.79	0.99
Nvdiffrast [74] (MS-SSIM [171])	RGB	23.51	0.60	17.61	0.88	11.20	0.83	17.74	2.37	09.01	0.33	11.19	0.72	23.30	0.73

from a single image. The complex light paths within the kaleidoscope result in high-variance RGB gradients that cannot be reliably used for gradient-based optimization. Moreover, small parameter perturbations can dramatically change the kaleidoscope patterns, leading to high variance even in variational derivatives and preventing optimal recovery. We find that our approach of combining multi-scale histogram matching with parameter-space blurring is ideal for this task and successfully recovers the optimal configuration. We use the same scale-space parameters as in § 4.4.2 and render images at 200×200 resolution.

Real Scene

Inverse rendering with real photographs presents additional challenges, such as noise from sensors and lenses, and calibration errors. We test our method on a real scene, where we aim to recover the position of a glass orb on a known planar surface from a given photograph. The scene is initialized with calibrated light sources, camera parameters, and known material (borosilicate glass), but the position of the orb is unknown (Fig. 4.10 ①). Similar to the previous scene, we use variational optimization. We observe that using matching stationary images ($\sigma = 0, \alpha = 0$) is sensitive to noise and fails in this task (Fig. 4.10 ⑤). We attribute this failure to the same alignment issue as in `Shadown Mini`, where one of the samples in the variational estimate (Fig. 4.10 ④ shows the mean of all samples) needs to overlap with the target position of the orb. Our local-histogram matching approach does not require perfect alignment between the

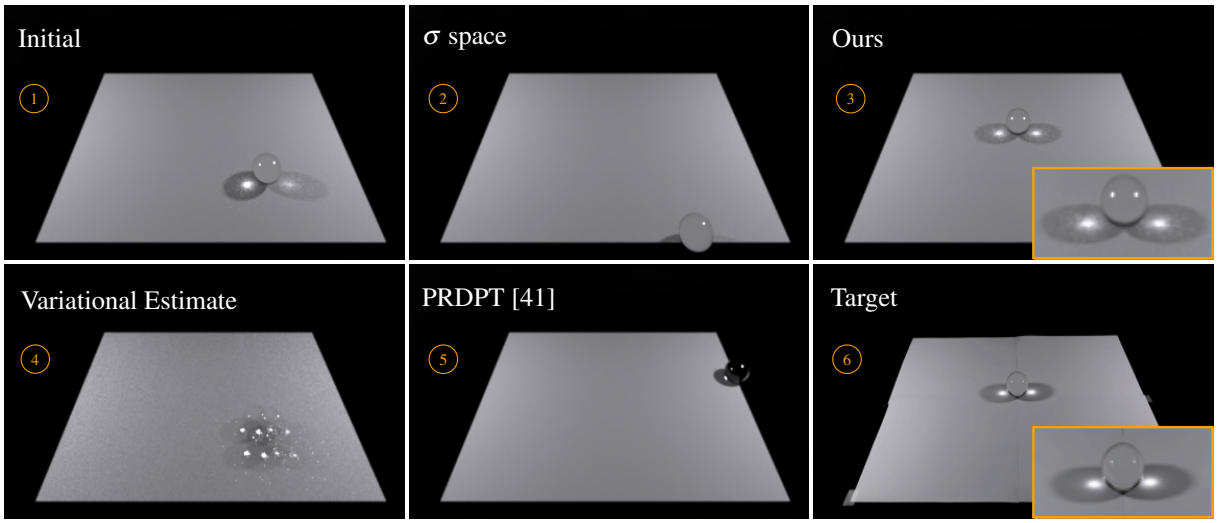


Figure 4.10. Using a real photograph. We show a preliminary experiment with real data that validates our method's robustness against sensor noise and calibration errors. Starting with known material, camera, and lighting, and a randomly positioned glass orb ①, we recover the optimal translation from a real photograph ⑥ using variational optimization. The variational estimate with 8 samples is shown in ④. Matching at stationary resolution ⑤ or only in the σ space ② fails in this task. Our method achieves optimal recovery starting from any initial position on the surface.

variational estimate and the reference images, allowing it to reliably recover the target position of the orb from any initialization on the planar surface (Fig. 4.10 ③). As an ablation on the α and β scale spaces, we also test matching only in the σ space with the same kernel parameters, which fails in this task (Fig. 4.10 ②).

4.4.4 Differentiable Rasterization

We evaluate our method on scenes provided by Xing *et al.* [180]. The tested scenes vary in complexity, requiring optimization for both geometric and appearance parameters. Quantitative results are in Table 4.3, using metrics to assess both visual appearance (PSNR) and parameter recovery accuracy (MAE). For low-dimensional parameter spaces (e.g., `Translation` and `Rotation`), our method using *only* RGB derivatives performs comparably to Xing *et al.*'s [180] approach, which utilizes both RGB and Lagrangian derivatives. For high dimensional parameter spaces, such as optimizing the full camera-pose matrix or texture map, RGB gradients combined

with local histogram matching performs better. We use the same scale parameters as in § 4.4.1.

4.5 Choosing kernel parameters

The parameter β defines the tonal scale, σ defines the image resolution, and α defines the spatial extent over which histograms are integrated. Since β implicitly defines the bin-width as $1/\beta$, it represents the effective tonal resolution. Intuitively, the number of bins limits the number of modes in the distribution. Since our goal is to preserve these modes at coarser scales, we choose β appropriately. In the limit as $\beta \to 0$, the *full* intensity distribution is maintained. For all of our experiments, we find that the maximum number of local histogram modes is 6, and choosing $\beta \geq 1/32$ (*i.e.*32 bins) works reasonably well. The parameter α is critical for long-range motion of image features, as it defines the extent of gradient support. Intuitively, low α values can be used for more local alignment in image space, while high values enable longer-range alignment. For complex problems involving both local and global alignment, we use multiple scales.

Ablating σ and α

Since β does not extend gradient support, using it alone without blurring histogram contributions via α is functionally equivalent to matching images at a stationary resolution. Figure 4.12 illustrates this with an optimization problem where using β in isolation leads to poor local minima.

Texture Recovery

We recover the texture of a planar surface starting from a random initialization. Contrary to what one might expect, high β values do not degrade the quality of recovery on this task. The low tonal resolution resulting from high β values does not affect the resolution of the underlying texture. As shown in Figure 4.15, the choice of β has a marginal effect on recovery quality. We demonstrate the advantages of using α space compared to σ space on this task in Figure 4.11.

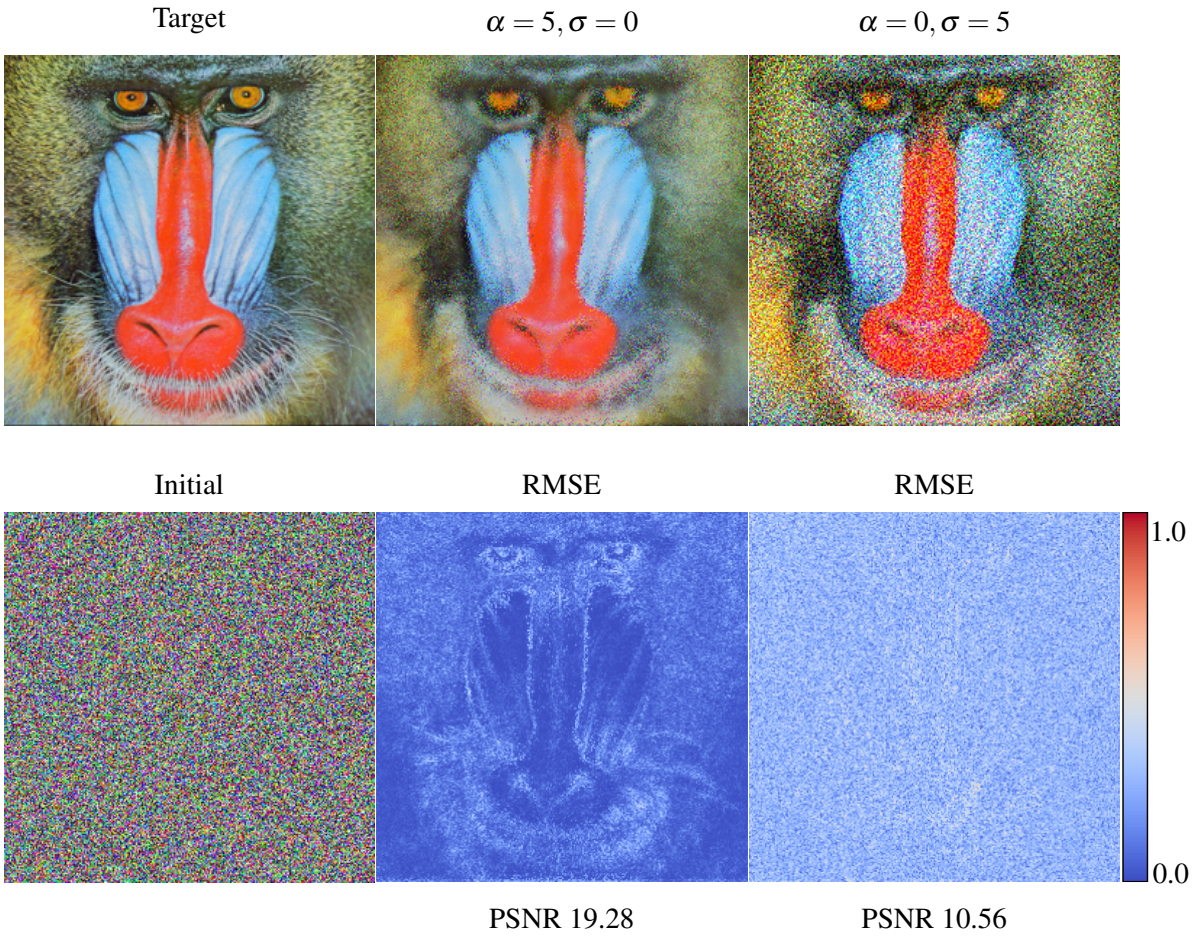


Figure 4.11. Texture Recovery. Starting from a random initialization, we recover a reference texture using $\beta = 1/8$. We compare our histogram-space blurring approach ($\alpha = 5, \sigma = 0$) with standard σ-space blurring ($\alpha = 0, \sigma = 5$). The σ-space matching yields suboptimal results since it only measures errors in the means of the distributions.

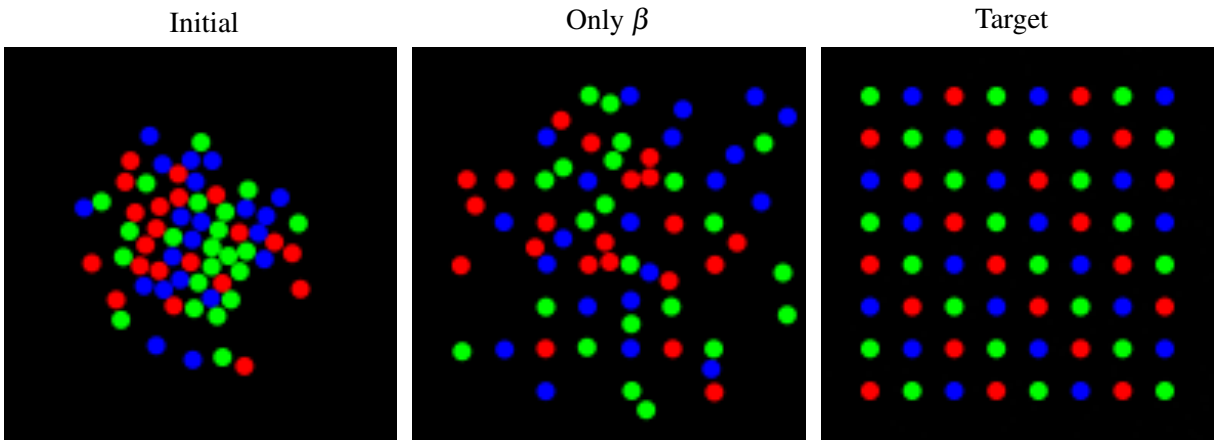


Figure 4.12. Ablation on α, σ. The parameter β alone does not influence the support of RGB gradients. When used in isolation without the α and σ spaces, the optimization converges to poor local minima. We demonstrate this by attempting to recover disk positions from random initializations using only β scale space.

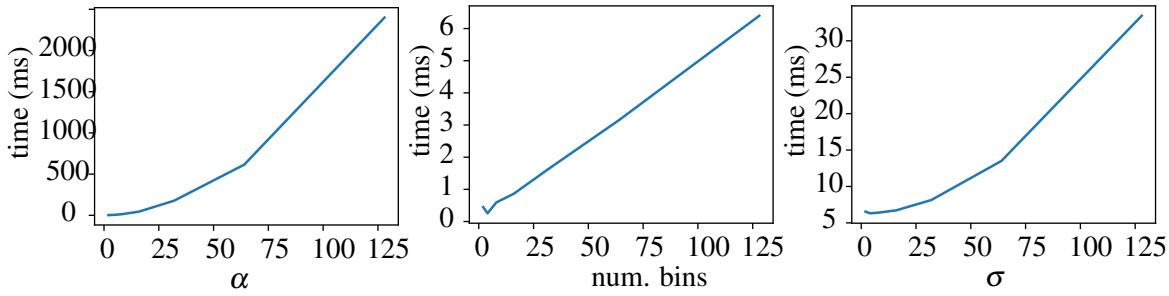


Figure 4.13. Runtime Analysis. We measure the effects of kernel-width parameters (α, β, and σ) on the total runtime of computing LOIs. The runtime increases linearly with kernel width, with histogram-space blurring (α kernels) being the slowest operation.

4.6 Runtime

We measure the effect of the LOI parameters on wall-time for feedforward operations. Please refer to Figure 4.13.

4.7 Implementation Details

Differentiable Vectorization

For this set of experiments, we use 128×128 images and recover the positions of n disks, making it an $n \times 2$ dimensional optimization problem. The disk positions are initialized by sampling from a normal distribution $\mathcal{N}(0,1)$. We compute image gradients using diffvg [80] and implement our method in PyTorch. We use a learning rate of 0.1 and the Adam optimizer with default parameters. For the kernel parameters, we set $\beta = 0.125$, $\alpha = [1,5,15]$ and $\sigma = [1,5,15,45]$. The parameter β also determines the histogram bin width, resulting in 8 bins for these experiments. Each disk has a width of 4 pixels. To prevent disk overlap, we resolve collisions at each optimization step. We provide error plots and visual comparisons for randomly selected runs in Figure 4.18.

Differentiable Path Tracing

For path tracing experiments, we render images at 256×192 resolution using Mitsuba 3 [60] with 64 samples per pixel for computing gradients. We use the `prb_reparam` integrator [169] with a maximum path length of 4. For optimization, we use Adam with default parameters and a learning rate of 0.01 for all parameters, except for light position parameters, for which we use a learning rate of 1. We provide additional visual comparisons and convergence plots beyond those in the main manuscript in Figure 4.17.

Differentiable Rasterization

We evaluate our method on the rasterization benchmark from [180]. All images are rendered at 128×128 resolution, using the same rendering and optimization parameters as [180].

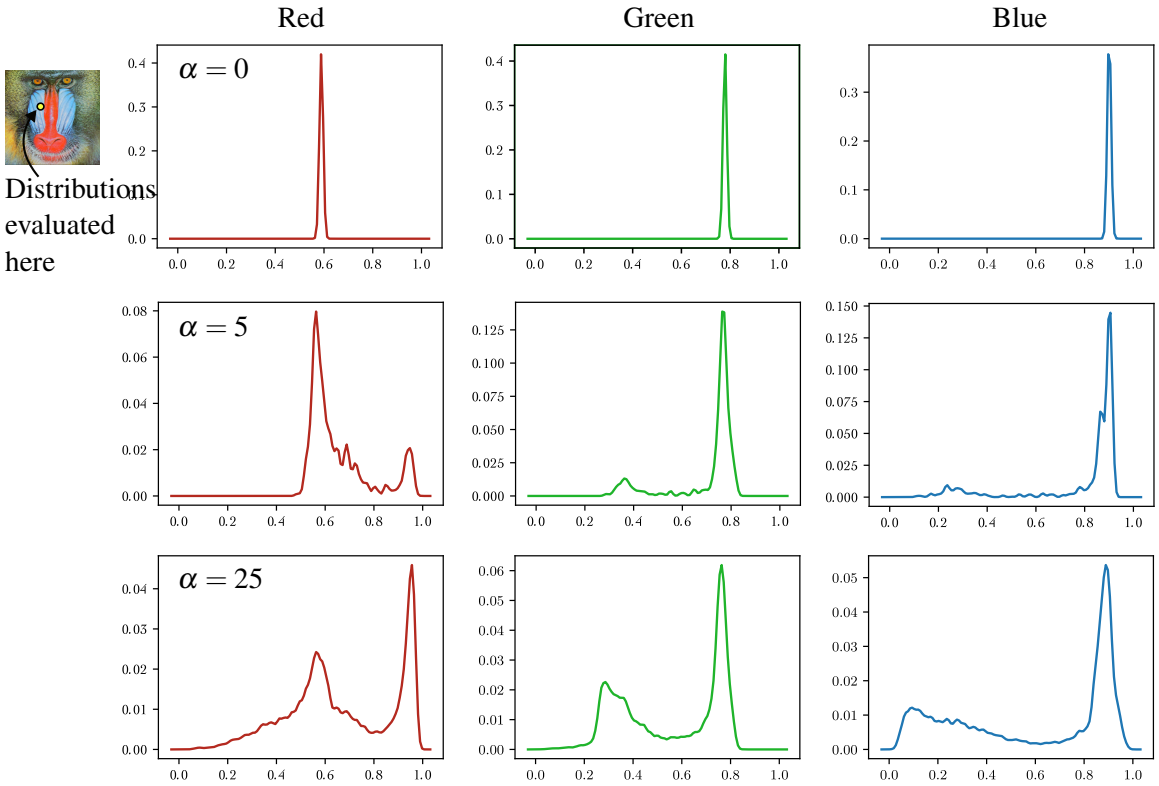


Figure 4.14. Histogram visualization. We visualize the histogram contributions across the three color channels at different α kernel widths. While standard Gaussian Pyramids only preserve the mean of each distribution, our method preserves their modes.

Visual comparisons between our method and standard scale space matching are shown in Figure 4.16. We use five random initializations for each task and report average error metrics in Table 2 in the main manuscript.

4.8 Discussion

Ablations

Our method relies on two critical components: the α and β scale spaces. Using only the σ space reduces our method to standard Gaussian Pyramids (GPs), as shown in our quantitative evaluations (Tables 4.1, 4.2, and 4.3). While using only the β space preserves tonal variations, it fails to extend the spatial influence of gradients, leading to poor performance across our experiments . Since the α space is dependent on β (Eq. 4.4), it cannot be used in isolation.

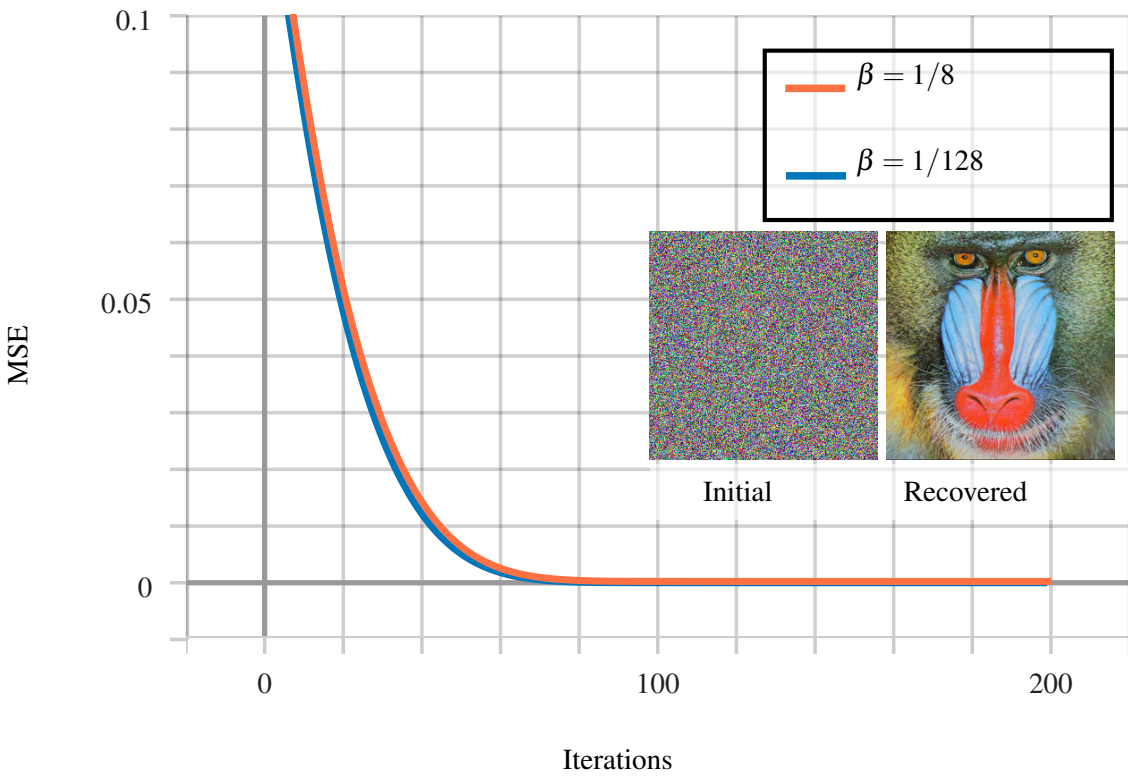


Figure 4.15. Our method remains robust to different choices of the β-kernel width when recovering a textured plane. We demonstrate this by comparing image error plots across two different kernel widths, both of which converge to good local minima.

Limitations

As our method operates exclusively in image space, parameters that do not influence primary or secondary effects still yield zero gradients in the LOI representation. For example, in the `mug` scene (Fig. 4.19) from [41], optimizing the mug's rotation when its handle is not visible in the initial view is infeasible. In such cases, our method is most effective when combined with variational optimization approaches, where parameter-space blurring can reveal occluded features like the handle through sampling.

Conclusion

In this work, we revisit the idea of locally orderless images (LOIs) in the context of differentiable rendering. One of the key contributions of this work is bridging these previously disconnected areas in computer vision and graphics. Through various experiments, we show that

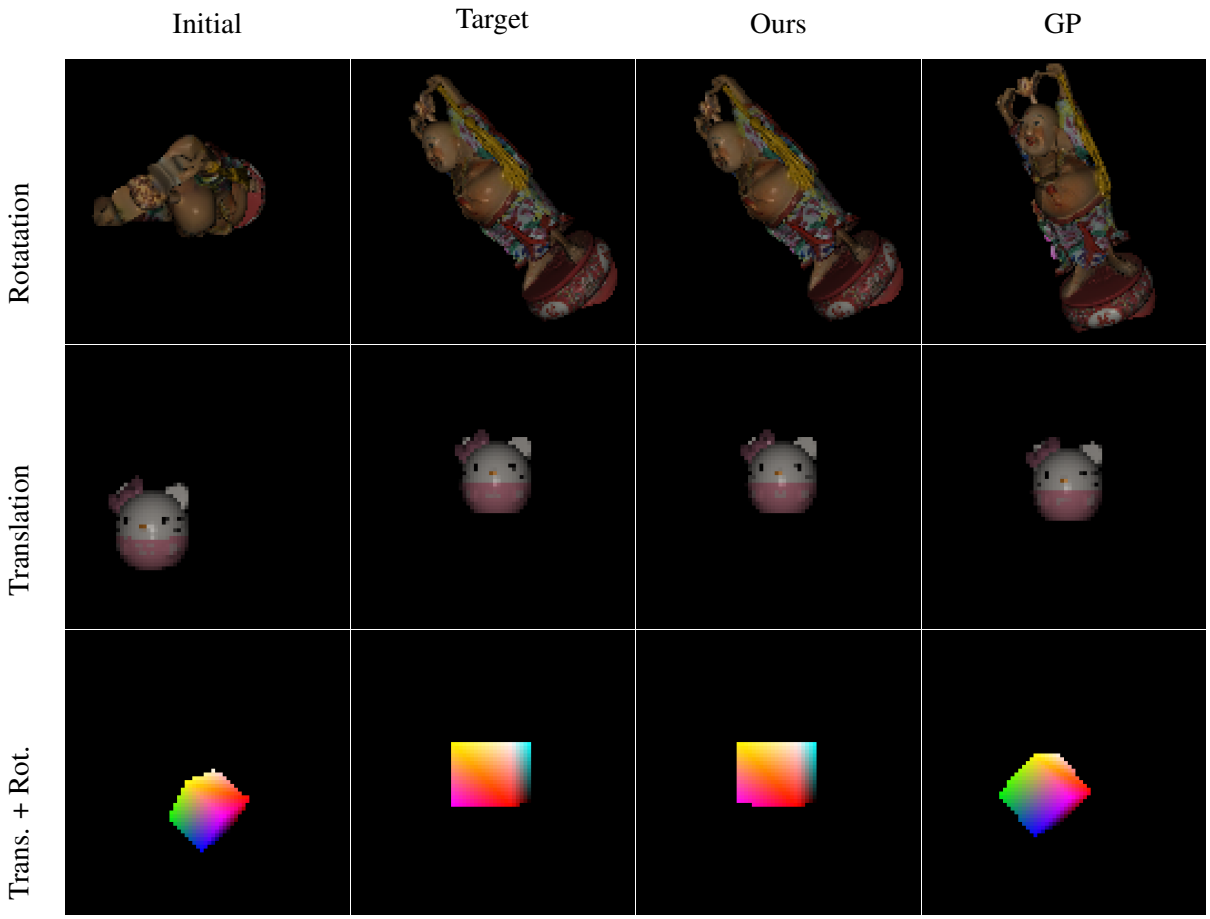


Figure 4.16. Visual comparisons on rasterization benchmark. We show visual comparisons on three tasks from the rasterization benchmark in [180].

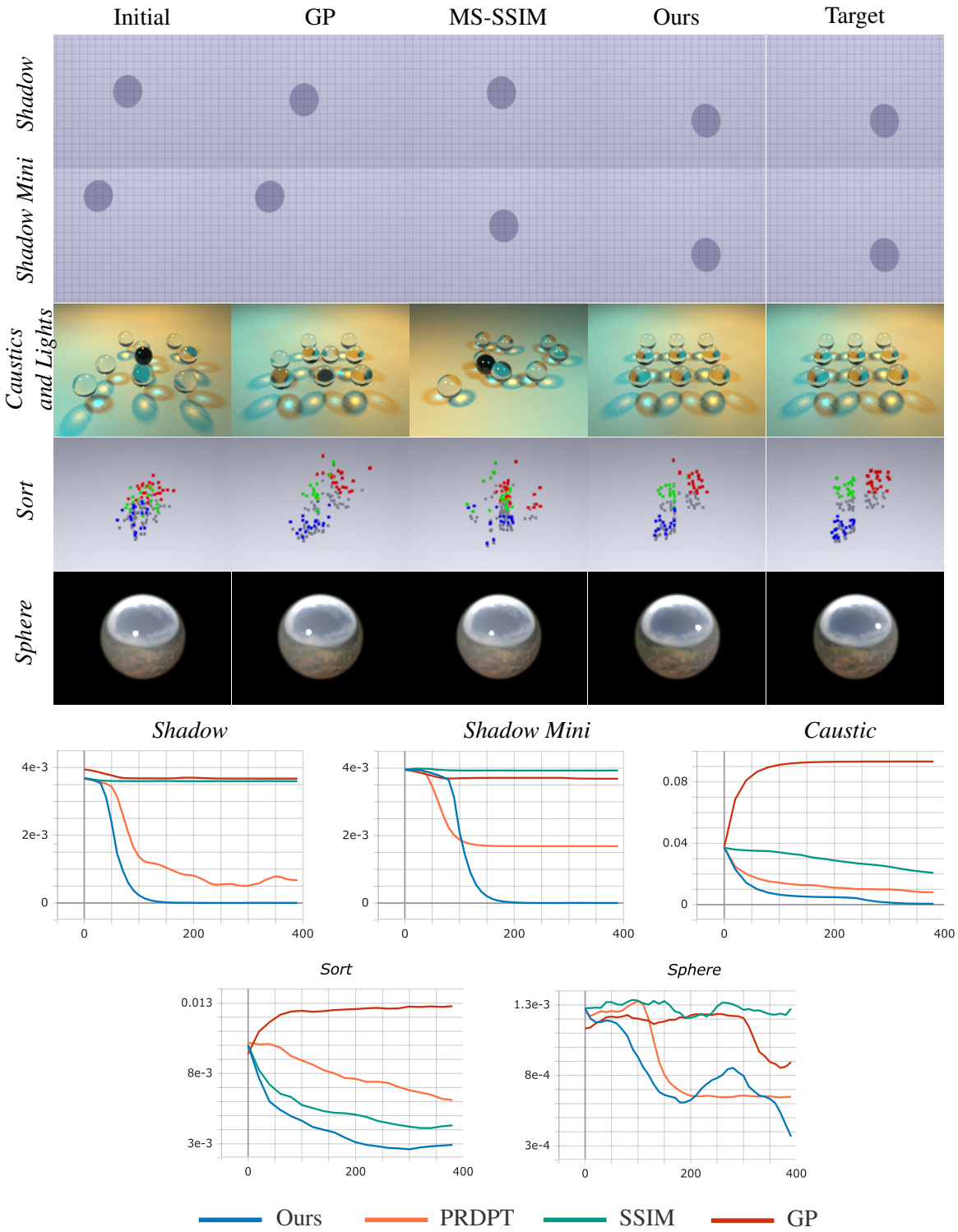


Figure 4.17. More comparisons on Differentiable Path Tracing. We show additional comparisons with GP [1] and MS-SSIM [171] along with Error plots (Image MSE *vs.*Iterations) for each scene in Figure 7 in the main manuscript.

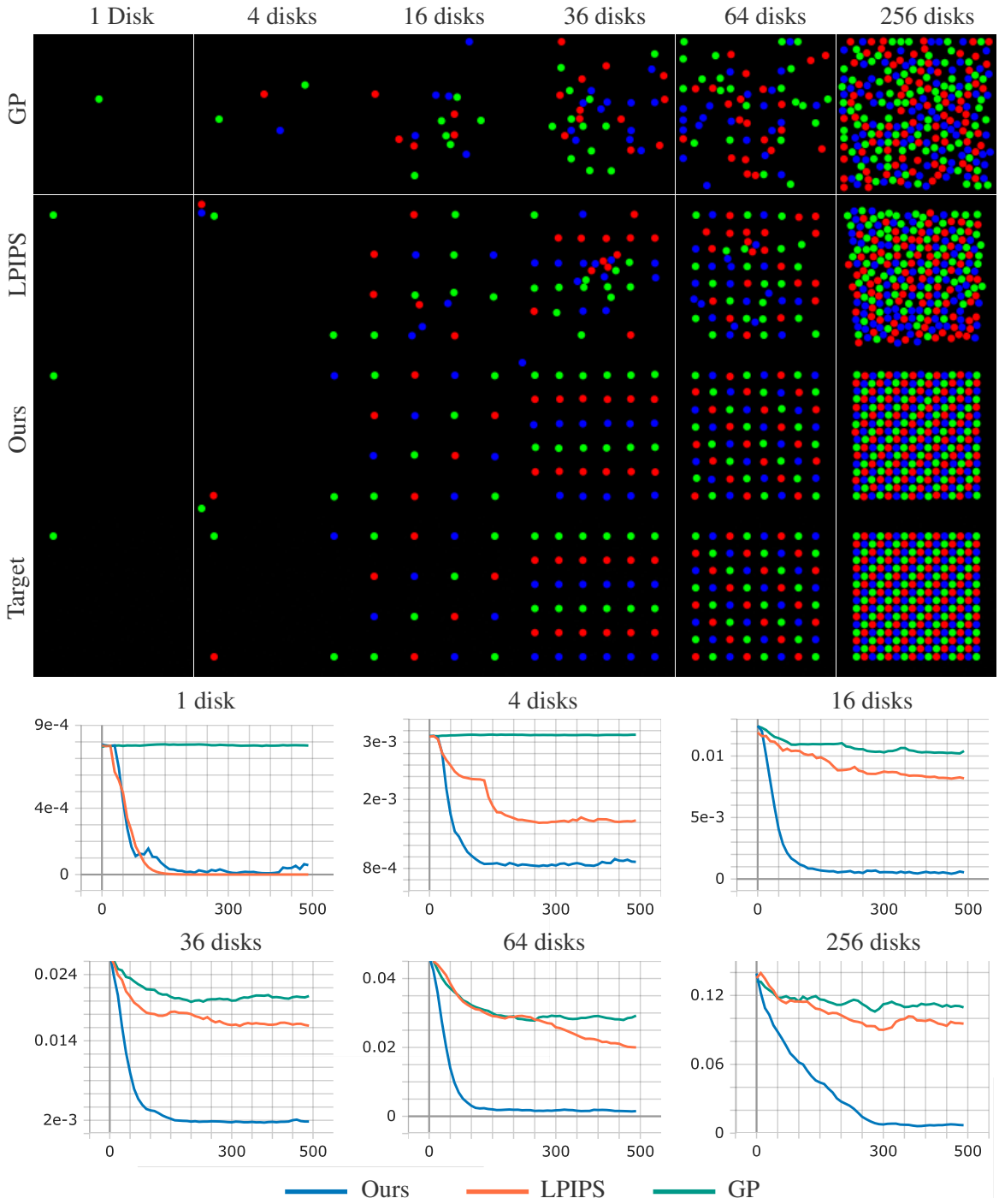


Figure 4.18. 2D Comparisons. We show convergence plots (Image MSE *vs.*Iterations) and optimized results for the 2D benchmark task of optimizing disk positions (Table 1 and Figure 6 in the main manuscript).



Figure 4.19. Our method is not designed to recover parameters that do not influence the image, such as the rotation of the mug [41].

our method yields better recovery in optimization problems where existing methods are likely to fail. Our method is straightforward to implement and integrates seamlessly with existing differentiable renderers without requiring any modifications to their core functionality.

Chapter 4, in full, is a reprint of the material as it appears in "Locally Orderless Images for Optimization in Differentiable Rendering", Ishit Mehta, Manmohan Chandraker, Ravi Ramamoorthi, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025. The dissertation author was the primary investigator and author of this paper.

5 | Conclusion

This dissertation studied the inverse rendering of geometry through analysis-by-synthesis, organized around the representation of the geometry, the gradients from the renderer, and the objective that compares images. We close by drawing together the conclusions of the three chapters.

Chapter 2 formulated a level-set evolution method for parametrically defined implicit surfaces. Unlike previous work, the method is in agreement with the classical level-set theory: it does not require surface extraction to be differentiable, and it lets a variety of mesh-based algorithms be applied to neural implicit surfaces. One limitation is that, although the surface deforms as dictated by the flow field, the implicit function need not retain its gradient characteristics during evolution. A signed-distance function, for instance, may not remain one, which can hamper algorithms like sphere tracing that rely on reliable distance queries, and is an interesting avenue for future research. We hope this work encourages further inquiry into geometry optimization and draws attention to some of the elegant methods in computer vision and graphics developed before the deep-learning era.

Chapter 3 developed topological derivatives for the variational optimization of image functionals, which prompt visibility changes away from the shape's contours. These derivatives can be read as a *soft* notion of the occupancy and segmentation losses common in shape reconstruction, and can be used even when a segmentation mask of the final geometry is unavailable; compared with volumetric-rendering methods, they sample efficiently and admit more complex materials and light-transport effects. A gap in reconstruction quality nonetheless remains, and a few challenges, if resolved, could close it: balancing the shape and topological-derivative terms during surface evolution in 3D, the constant-curvature assumption in the 3D

derivation, deriving topological derivatives for phase nucleation in 3D, and the joint optimization of geometry together with color, material and lighting.

Chapter 4 revisited the idea of locally orderless images in the context of differentiable rendering; its central contribution is bridging these previously disconnected areas of computer vision and graphics. Across our experiments, the resulting objective yields better recovery on optimization problems where existing methods are likely to fail, and it is straightforward to implement, integrating with existing differentiable renderers without requiring any modifications to their core functionality. Its main limitation follows from the property that makes it work: by summarizing each neighborhood as a histogram, the objective discards spatial detail, so two different images with the same local intensity statistics can look equivalent to it. Pairing the histogram loss with a small amount of spatial supervision, or annealing from coarse to fine histograms as the optimization converges, is a natural way to recover that detail without giving up the wider basin of attraction.

5.1 Future Directions: Inverse Design with Intuitive Control

The inverse methods in this dissertation recover physical attributes of a scene, but those attributes are legible mostly to specialists. A natural next step is to build scene representations that stay physically grounded while also being usable by non-experts and by the large foundation models now common in vision and language. These models reason well about semantics, yet they have little grasp of the physics behind the content they produce.

One way to supply that grasp is to make a differentiable physics engine part of the generative loop, so that a multimodal model can check its own output against the laws it should obey. Early steps in this direction use domain-specific languages for spatial reasoning, which give a language-level handle on physically accurate generation. The same idea carries over from inverse rendering to inverse design: instead of asking what scene produced an image, we can ask what shape meets a specification, for example a chair that holds a load of two hundred pounds

without failing.

Placing differentiable physics inside the learning pipeline has a further benefit. It constrains the space of solutions to those that are physically realizable, which speeds up both forward simulation and inverse inference. More important, it moves us toward systems whose outputs can be traced back to their causes, and so toward artificial intelligence that is easier to interpret and to trust.

A | Modulated Periodic Activations for Generalizable Local Functional Representations

The main chapters take the representation of geometry as given and study how to evolve and optimize it. This appendix steps back to the representation itself. It presents a neural functional model, developed before the work in the preceding chapters, that encodes a signal such as a surface in the weights of a network and generalizes across many instances. The local, modulated construction it introduces is the kind of building block that the inverse-rendering methods of Chapters 2 through 4 operate on, and it is included here for reference.

A.1 Introduction

Functional neural representations using Multi-Layer Perceptrons (MLPs) have garnered renewed interest for their conceptual simplicity and ability to approximate complex signals like images, videos, audio recordings [144], light-fields [102] and implicitly-defined 3D shapes [31, 124, 9]. They have shown to be more compact and efficient than their discrete counterparts [88, 145]. While recent contributions have focused on improving the accuracy of these representations, in particular to model complex signals with high-frequency details [144, 158, 102], it is still challenging to generalize them to unseen signals. Recent approaches typically require training a separate MLP for *each* signal [102, 36]. Previous efforts sought to improve generalization by imposing priors on the functional space spanned by the MLP parameterization [124, 139], using hypernetworks [71, 144], or via meta-learning [141]. But multi-instance generalization still causes significant degradations in quality.

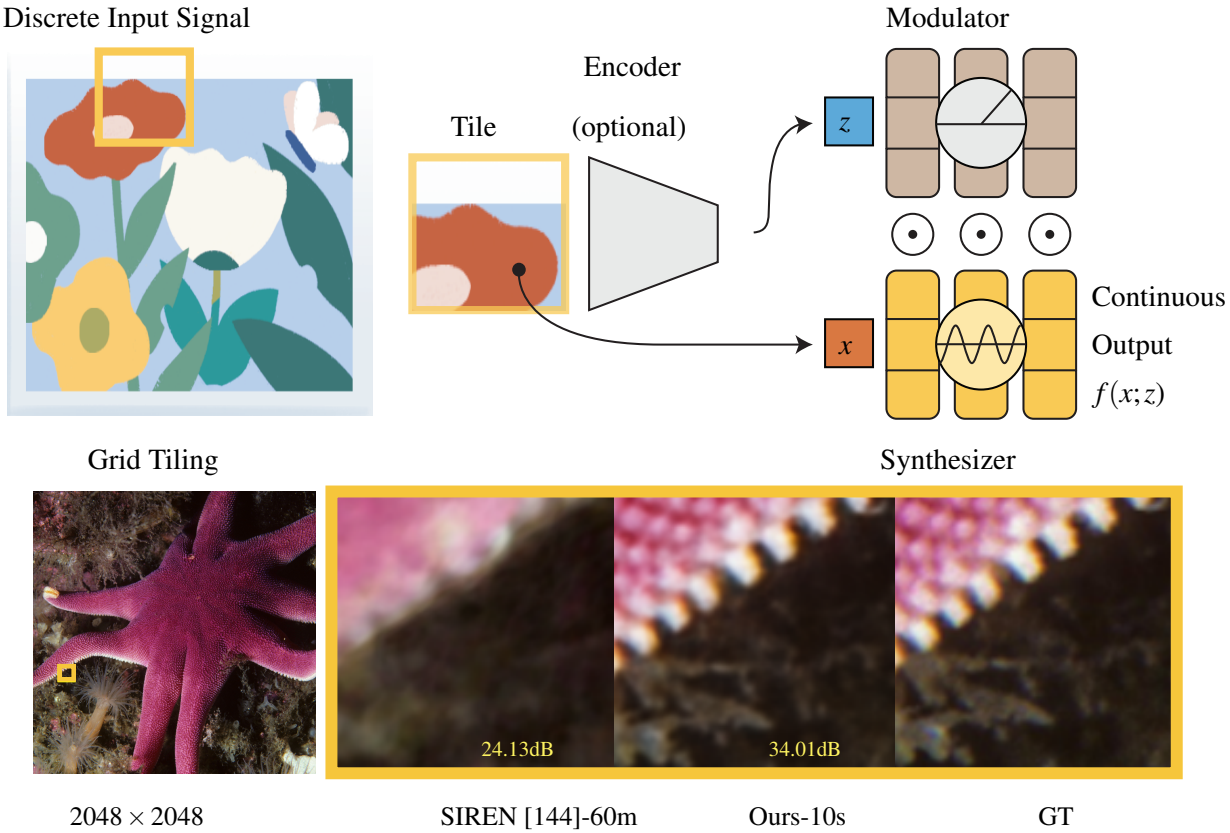


Figure A.1. (*Top*) We propose a new method to encode discrete signals as neural-functional representations. Each signal is represented as a structured grid, with each tile defined by a latent code. The tiles are encoded as continuous signals by a synthesis network using periodic activations. The latent codes corresponding to the tiles are used to modulate the activations using a modulation network. (*Bottom*) Our generalizable method is faster and enables higher quality reconstructions than previous methods.

We introduce a neural functional representation that *simultaneously* achieves high-reconstruction quality and generalizes to multiple instances. Our approach can encode functional representations for multiple discrete signals using a *single* model. Unlike previous works, which train a model for each signal, it can do so in a single feed-forward pass (Fig. A.1). We represent each signal using a low-dimensional latent code. These codes serve as conditioning variables in a functional mapping that uses two MLPs: a modulator and a synthesis network. The *synthesis* network implements a mapping from coordinates (*e.g.*spatial position) to signal values (*e.g.*RGB colors). It uses the sine function as activation, which enables accurate reconstructions of high-frequency content [144], but also makes naive conditioning strategies ineffective (§ A.3.2). The

modulator is the key to generalization. It consumes latent code and outputs, at each layer, parameters that modulate the amplitude, phase and frequency of periodic activations in the synthesis network. The modulator uses ReLU activations. Our model can either be used as an autoencoder, where the latent codes are produced by a third network (the encoder); or as an auto-decoder, where the latent codes are optimized jointly with all the network parameters.

As we show in Figure A.2, the quality of functional representations that fit images as-a-whole, degrades as we increase the target resolution. High-resolution images typically have a broad power spectrum, thereby requiring more expensive models to represent them functionally. But images are usually much simpler locally: simple edges and textures re-occur commonly across images that are otherwise quite distinct at the global level. This motivates our strategy to exploit locality. We partition the signal domain into a regular tiling, and assign each tile a latent code (Fig. A.1). By "zooming in" on the local structure, computing functional approximations that generalize becomes more tractable [94], because simple parts exhibit fewer variations than complete objects [62]. Some recent work has explored locality, but they focus on relatively simple, low-frequency signals like signed distance fields [26, 62], which can locally be well approximated using a single linear decision boundary—well in the purview of ReLU-based MLPs. For more complex signals like images and videos, even local patterns contain high-frequency components that a standard ReLU-MLP fails to reconstruct (Figure A.3). We show that locality, together with our model architecture, makes it possible to obtain functional representations of large, complex signals.

Compared to previous methods, ours produces qualitatively and quantitatively superior functional representations, with improved generalization capabilities. In summary, our contributions are as follows:

- A local neural-functional representation that enables generalization and achieves high fidelity. We use a set of local functions defined on a tiling of the input domain that combine to reconstruct the target signal.

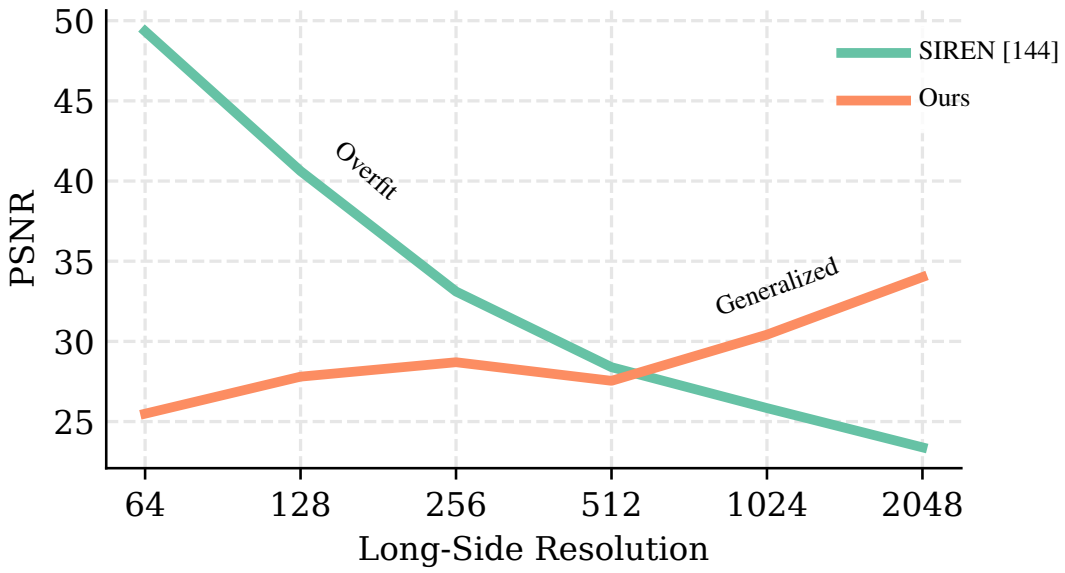


Figure A.2. SIREN struggles with high-res signals. Obtaining a neural representation of an image using SIREN [144] requires overftting a new MLP to every new image instance. This works well in practice for low-resolution images, but we observe a sharp drop in reconstruction accuracy for high-resolution images. Our method can be used to encode *multiple* images at high-resolution with only feedforward passes without gradient computation.

- A new network architecture, which uses modulation and synthesis sub-networks for high-fidelity functional neural representations of images, shapes and videos.
- A novel conditioning mechanism in which a ReLU-MLP modulates the amplitude, phase, and frequency of periodic activations in the synthesis sub-network.

A.2 Related Work

Continuous Representations of Visual Signals

Our work builds upon the extensive use of Multi Layer Perceptrons (MLPs) to encode images [131, 153], videos [144], shapes [124, 31] and 3D scenes [146, 88, 16, 102]. Once trained, these models yield continuous representations that can be queried at arbitrary locations in the signal's input domain. They have had significant impact in view-synthesis [88, 102] and other interpolation problems [105, 163]. Similar approaches have been used for end-to-end differentiable texture mapping [106] and volumetric rendering [114].

Periodic Activations

Lapedes and Farber [75] show the earliest use of periodic activations in neural networks. They observe that networks with more than one layer with periodic activations are difficult to train, and often converge to undesired local minima. This problem is formalized further in [122]. For small datasets, Sopena *et al.* [149] show compelling results using sine activations in the first layer and monotonic functions in the others. This is similar to preconditioning the input using a Fourier basis, which was shown to be useful in feature visualization [118] and image synthesis [106]. More recently, Tancik *et al.* [158] proposed Fourier Feature Networks (FFN), where they encode the MLP's input into a high-dimensional space using a random sampling of Fourier basis functions. Concurrently, Sitzmann *et al.* [144] showed that, with careful network initializations, sine activations can be used in all layers. They demonstrate regressions of small, single images and videos, as well as more complex shapes. However, as we show in Section A.4, these networks struggle with larger datasets, or individual instances when the complexity is increased. In Figure A.2, we show how quality degrades with these networks as we regress a progressively higher-resolution image, or a longer video. Our work lifts these limitations by exploiting locality, and introduces an effective modulation mechanism to enable generalization.

Instance-conditioned Implicit Functions

A major limitation of current implicit representations, is that they need to be optimized for each test signal individually, unlike more established models than only require a forward pass at test time, having been trained on large datasets. Building implicit models with similar generalization properties is typically done via a conditioning mechanism, using latent variables. Conditioning by contatenating the latent code with an MLPs spatial input coordinates has been successful in signed-distance fields regression tasks [31, 124]. Schwarz *et al.* [139] use the same strategy for MLPs encoding radiance fields, although with limited resolution. In Section A.3.2, we show why this conditioning-by-concatenation approach is inadequate for MLPs with sine activations, and limits reconstruction quality. Conditional hypernetworks [53]

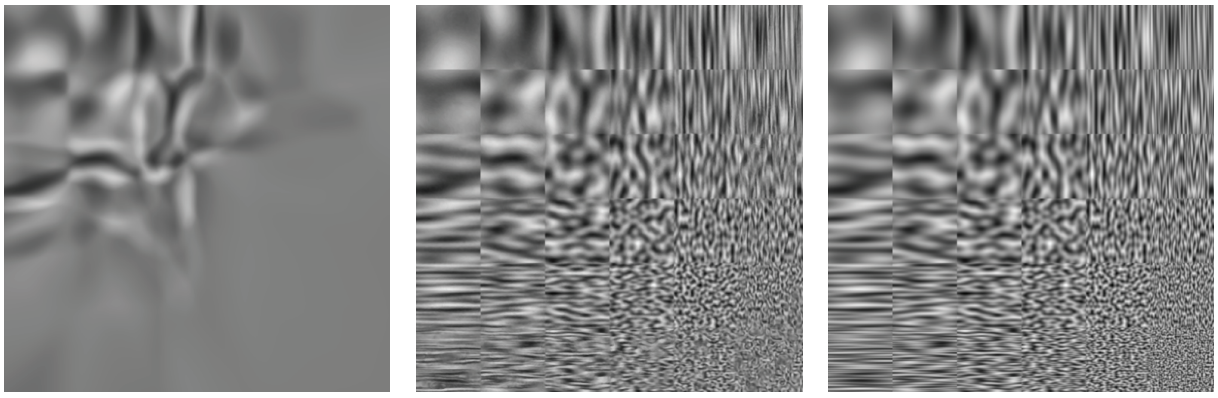
achieves similar goals. A hypernetwork estimates all the parameters of a hyponetwork [144] from the latent code. Hypernetworks are prohibitively expensive, in both compute and memory, thereby limiting the resolution of the reconstructed signal in practice. We propose a new approach to modulate the implicit function based on the conditioning variable, which is inspired from attention mechanisms [167].

Applications of Local Models

Local methods have been used largely to process complex systems in the form of KD-trees for real-time fluid simulation [34, 191], as regular grids for photon mapping [51] and popularly for fast ray tracing [5]. Local representations have also been used to compress surface light fields [177] and for pre-computed radiance transfer [94]. Our method also relates to recent work on voxelized implicit models for 3D representation [26, 44, 145, 62]. We show that our approach is general, and can be used for a variety of applications.

A.3 Method

We introduce a novel parameterization of neural functions approximating signals defined on a Euclidean input domain $\mathbb{R}^n$. Our pipeline is illustrated in Figure A.1. Our model can simulatenously encode a large number of functions, each of which is summarized into a latent code. The latent codes are processed by a modulation network, which conditionally modulates the activations of a synthesis network, that acts as a template for the functional mapping (§ A.3.1). We show that this new architecture is crucial to generate functional representation of multiple signals with a single model, a task where our approach significantly outperforms previous work that use a single MLP (§ A.3.2). This ability to generalize lets use compute functional representations for signals with much higher-resolution than previously possible, and with much higher fidelity. For this, we decompose the input signals into local tiles, each represented with a latent code (§ A.3.3). The latent codes can either be estimated from the discrete input by a convolutional encoder (§ A.3.4), or they can be optimized simultaneously with the model



ReLU Sine (ours) Reference

Figure A.3. Sine vs. ReLU synthesizer. MLPs using ReLU activations (*left*) fail to reconstruct high-frequency components of the target signal; here a 6×6 grid of Perlin [126] texture patches, where the horizontal and vertical frequencies increase from the top-left to bottom-right corner (*right*). Sine-MLPs (*middle*) are able to reconstruct elements from a wider frequency spectrum.

parameters (§ A.3.4), as in [26].

A.3.1 Modulated Periodic Activations

Concretely, we define our functional representation as a continuous conditional mapping.

$$f_\theta : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^m. \tag{A.1}$$

f_θ is a neural network, with parameters θ, $d = 256$ is the dimension of the latent space. In the case of images, we use $n = 2$ for pixel coordinates, $m = 3$ for the RGB color values. We find the optimal model parameters θ by minimizing a domain-specfic reconstruction loss $\mathscr{L}(f_\theta(x_i; z), y_i)$ on a datset of signal values $y_i \in \mathbb{R}^m$, sampled at coordinate $x_i \in \mathbb{R}^n$. The signal is encoded into a latent code z. Our network architecture has two components—a synthesis network (§ A.3.1), and a modulation network (§ A.3.1).

Synthesis Network

The synthesis network defines a continuous function from the spatial coordinates of a discrete signal like an image to its output domain (*e.g.*color). It is a composition of K hidden layers, with hidden features $h_1, \ldots, h_K$. Each layer uses a periodic-nonlinear activation function

and is defined recursively as:

$$h_i = \alpha_i \odot \sin(w_i h_{i-1} + b_i), \tag{A.2}$$

with $w_i \in \mathbb{R}^{d_i \times d_{i-1}}$ and $b_i \in \mathbb{R}^{d_i}$, the learnable weights and biases for layer i, and $\alpha_i \in \mathbb{R}^{d_i}$ a modulation variable discussed in Section A.3.2. We set $h_0 := x \in \mathbb{R}^n$ to be the input coordinates. The sine function is applied pointwise and $\odot$ denotes element-wise multiplication. Sine activations have proven to be beneficial in modeling high-frequency signals [144]. We confirm this by comparing our synthesis network to an alternative that uses ReLU activations in Figure A.3.

Modulation Network

We modulate the activations of the synthesis network with a second MLP using ReLU activations, which acts on the latent code z corresponding to the target signal. It is defined recursively as:

$$h'_0 = \text{ReLU}(w'_0 z + b'_0), \tag{A.3}$$

$$\alpha_{i+1} = h'_{i+1} = \text{ReLU}(w'_{i+1}[h'_i \;\; z]^T + b'_{i+1}), \tag{A.4}$$

where $\text{ReLU}(\cdot) = \max(0, \cdot)$ and w'_i, b'_i are the weights and biases of the MLP. We establish an explicit relationship between w' and z, by feeding in z at every layer in the modulation network in the form of a skip connection. As can be seen from Equations A.4 and A.2, the latent codes z can modulate the amplitude of the sine activations of each hidden layer in the synthesis network, through the modulation parameters α_i Furthermore, expanding Equation A.2, we get

$$h_i = \alpha_i \odot \sin(w_i \cdot (\alpha_{i-1} \odot \sin(\dots)) + b_i), \tag{A.5}$$

which shows the latent codes also indirectly control the *frequency* and *phase shift* of the sinusoids in subsequent layers. Figure A.4 illustrates the expressivity of our modulation mechanism visually.

A.3.2 Expressivity of Modulation

A simpler alternative to using a separate modulation network would be to concatenate the latent codes and the input coordinates, and use the resultant vector as a *single* input to the synthesis network. This strategy has shown to be fruitful for ReLU-based synthesis networks for encoding signed distance fields [124]. However, we find it consistently fails with sine activations (see Section A.4.1 for details).

In this alternative conditioning mechanism, the network takes the concatenation $[x, z]$ as input, so the first layer can be rewritten as:

$$h_1 = \sin\left(w_{1,x}x + w_{1,z}z + b_1\right), \tag{A.6}$$

where $w_{1,x}$ and $w_{1,z}$ are submatrices of w_1, corresponding to x and z respectively. The latent codes z can therefore only act as a *phase shift*, $w_{1,z}z$, on the first layer. This severely limits expressivity, in contrast to our model, where the latent code z modulates the amplitude, frequency *and* phase-shift of the functional representation at *all* layers, via the α_i. Figure A.4 illustrates the difference in expressivity between our model and a concatenation-based MLP.

A.3.3 Local Functional Representations

The ability to generalize to more than one signal gives us an additional opportunity. Rather than computing a single neural function for the entire signal, we decompose the domain into a regular grid, and calculate a local continuous representation for each tile (illustrated in Figure A.5). Concretely, we assign each tile a latent code z_i, so that the entire signal is represented by a codebook $\{z_i\}$, and the corresponding neural functions $\{f_\theta(\cdot, z_i)\}$, whose first argument is

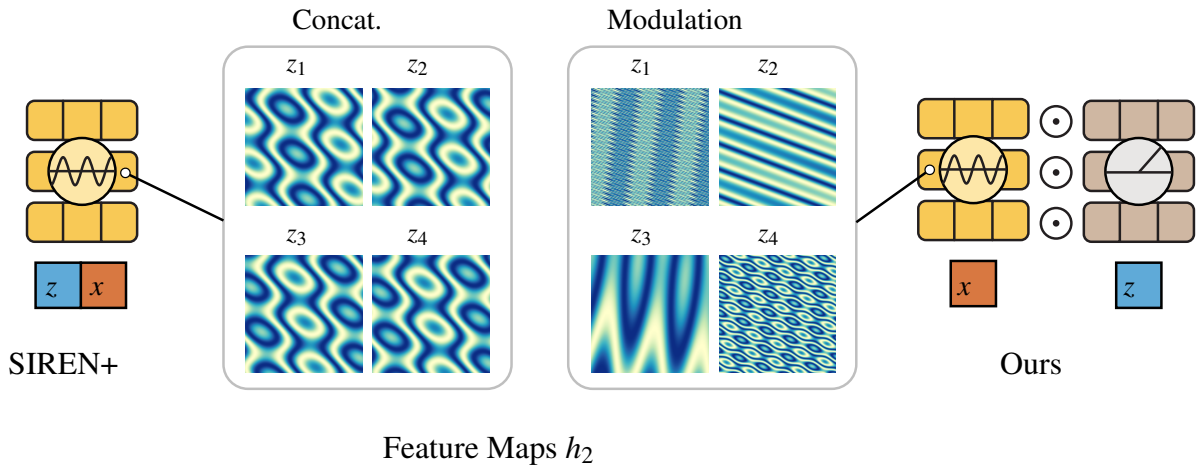


Figure A.4. Our modulator improves expressivity. Conditioning a sine-synthesizer by concatenation does not yield much control over the MLP's internal feature maps, thereby limiting the expressivity of the latent space. Here, we show feature maps obtained at the second layer of a randomly intialized synthesis network condtionally modulated on four different latent vectors z_i. Concatenating z_i with the input (*left*) only changes the phase of the signals at the second layer. Using our modulator sub-network (*right*) provides much more control over the internal feature maps, with variations in phase, amplitude and frequency.

the normalized local coordinates in the tile $x \in [0,1]^n$.

Continuity at tile boundaries.

In practice, to eliminate visual discontinuities at the tile boundaries, the images are split into a set of overlapping tiles. When evaluating the continuous representation, the contribution



Figure A.5. Continuity at tile boundaries. We partition the target signals into uniform grids of overlapping tiles. Here, an image (*left*) is partitioned into 4 tiles (*middle*) with overlapping areas. During inference the tiles are blended using an $n-$linear (bilinear here) weighting scheme (*right*). The colors represent the resultant contribution of each tile at a given position. (Best viewed in color)

of overlapping tiles is weighted n-linearly according to the distance between the point and the tile centers (Fig. A.5).

A.3.4 Training procedure

We present two modes of training of our model. In the auto-encoder setting, (§ A.3.4), the latent codes are estimated using a discrete encoder. In the auto-decoder configuration (§ A.3.4), the latent codes are randomly initialized and optimized with the network parameters as in [124].

Auto-encoder

Unless otherwise specified, we use our model in an auto-encoder configuration. Auto-encoding lets us to build a continuous representation, from discrete input signals, using an auxiliary encoder network (shown in Figure A.1). We use different encoders, depending of the task. Details can be found in supplemental. This could be useful in spatial super-resolution (images, videos), frame interpolation (videos), or reconstruction problems from sparse samples (lightfields, compression).

Auto-decoder configuration

In the auto-decoder configuration, we jointly optimize the network parameters θ, and the latent codes for all the training signals. That is, we do not use the optional encoder of Figure A.1. We use this configuration in our shape reconstruction experiments, as proposed by [124]. After training, we obtain a functional representation for new, unseen test signals by sampling a new latent code z for the unseen signal, and optimizing it with the same objective used during training, but this time keeping the network parameters θ constant. We initialize all latent codes $z \sim \mathcal{N}(0, s^2)$ as Gaussian random vectors with $s = 10^{-2}$.

A.4 Experiments

We demonstrate two classes of experiments, on three domains (images, videos, 3D shapes). First, we demonstrate the generalization capabilities of the proposed model (§ A.4.1) in a *global* setting. That is, we compute functional representations for many discrete signals, each of which represented (as a whole) by a latent code (i.e., without the tiling procedure described in Section A.3.3). Second, we show how our model can be used to learn *local* functional representations of discrete signals (§ A.4.2), with high reconstruction quality. In this set of experiments, each signal is defined using a latent codebook (one code per tile of the input signal). We also show our model can be applied to other multi-domain tasks, such as image relighting (§ A.4.3), where the function's input is a 2D pixel coordinate and a 3D lighting direction.

We compare to state-of-the-art MLP-based functional baselines, illustrated in Figure A.6, together with our model. These are:

ReLU/FFN a standard MLP with two inputs—latent code and sample coordinates. In case of FFN [158], the sample coordinates are transformed using a random fourier gaussian matrix with scale σ.

SIREN+ A single MLP with sine activations adapted from [144], with an additional input for the latent code z, concatenated with the coordinates x.

HyperNet-SIREN A SIREN whose weights are conditionally predicted using a hypernetwork [53], as described in [144]. Instead of a modulator sub-network, a hypernetwork takes the latent code z as input, and predicts *all* the parameters of the synthesis network (the w_i, b_i).

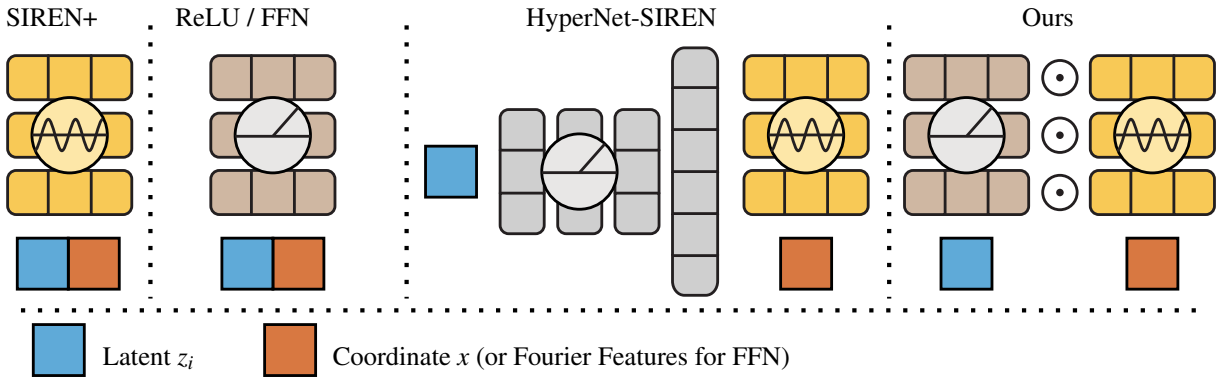


Figure A.6. Comparison Baselines. In all our experiments, we compare with state-of-the art baselines that use MLPs to encode visual signals. For consistency, we kepp the size of the MLPs constent across all models.

Table A.1. Comparison of generalization ability on CelebA. We train conditional functional on 200K 32×32 images from the CelebA [87] dataset, with a conditional latent code per image. We show PSNR on the test set at $1\times$ resolution to show reconstruction accuracy, and at $2\times$ resolution to show the interpolation behaviors of the functions. We mark the best result in yellow. Without careful tuning of σ, FFN produces discontinuous functions (red). HyperNetwork requires $\sim 60\times$ more parameters (orange).

Method	PSNR ↑	PSNR-2× ↑	Params.
ReLU	28.31	24.91	1.1M
SIREN+ [144]	20.15	19.19	1.1M
FFN ($\sigma = 10$) [158]	26.48	4.22	1.2M
FFN ($\sigma = 1$) [158]	27.87	24.37	1.2M
HyperNet-SIREN [53, 144]	26.70	24.54	72M
Ours	29.42	25.49	1.3M
Ours - Large Conv.	29.64	26.05	17M

A.4.1 Global Functional Representation

Images

We run our image experiments on the CelebA [87] and CIFAR-10 [73] datasets separately. We use our model in an auto-encoder setting (§ A.3.4). A convolutional encoder estimates a latent code for each image (details in supplemental). From the latent codes, we decode a functional representation for each image. All the images are resampled to 32×32 resolution for training. The parameters of the modulator, synthesizer and encoder are trained simultaneously

Table A.2. Comparison of generalization ability on CIFAR-10. We also show results on CIFAR-10 [73] dataset, which contains more diverse images than the CelebA dataset. The numbers are reported for the test set with 10K images. PSNR values for 2× resolution are computed against ground-truth images at 1× resolution upsampled using bilinear interpolation. SIREN+ fails to converge on this dataset.

Method	PSNR ↑	PSNR-2× ↑
ReLU	23.65	25.18
SIREN+ [144]	*n/a*	*n/a*
FFN ($\sigma = 10$) [158]	22.75	5.75
FFN ($\sigma = 1$) [158]	24.97	25.51
HyperNet-SIREN [53, 144]	18.07	19.16
Ours	25.73	27.05

to minimize the sum of a reconstruction loss,

$$\mathscr{L}(f_\theta(x;z_i),y) = ||f_\theta(x;z_i) - y||_2^2, \tag{A.7}$$

We train all the models for 1000 epochs. Our train/test splits contain 167K/33K and 60K/10K images respectively. We use 128×128 center-crops for CelebA and entire image for CIFAR-10 as ground truh. The images are resampled to 64×64 and 32×32 using bicubic sampling. More implementation details are provided in the supplemental.

We evaluate generalization by sampling f_θ at pixel centers, at the input image resolution 32×32 (1×) and computing the PSNR. Additionally, we evaluate continuity by sampling f_θ more finely, at 64×64 (2×) resolution, and comparing to ground-truth resampled to the same resolution; we do *not* train the models with these higher resolution targets. Table A.1, summarizes our result on the CelebA dataset, and Table A.2 on CIFAR-10 [73]. SIREN+ struggles with generalization, and in the case of CIFAR-10, does not even converge. We hypothesize this is due to the higher image variability in CIFAR-10, in comparison to CelebA where faces are aligned. We observe a similar behavior with HyperNet-SIREN. As shown in Tables A.1 and A.2, the scaling parameter σ of FFN [158] is critical for continuity: PSNR 2× drops with

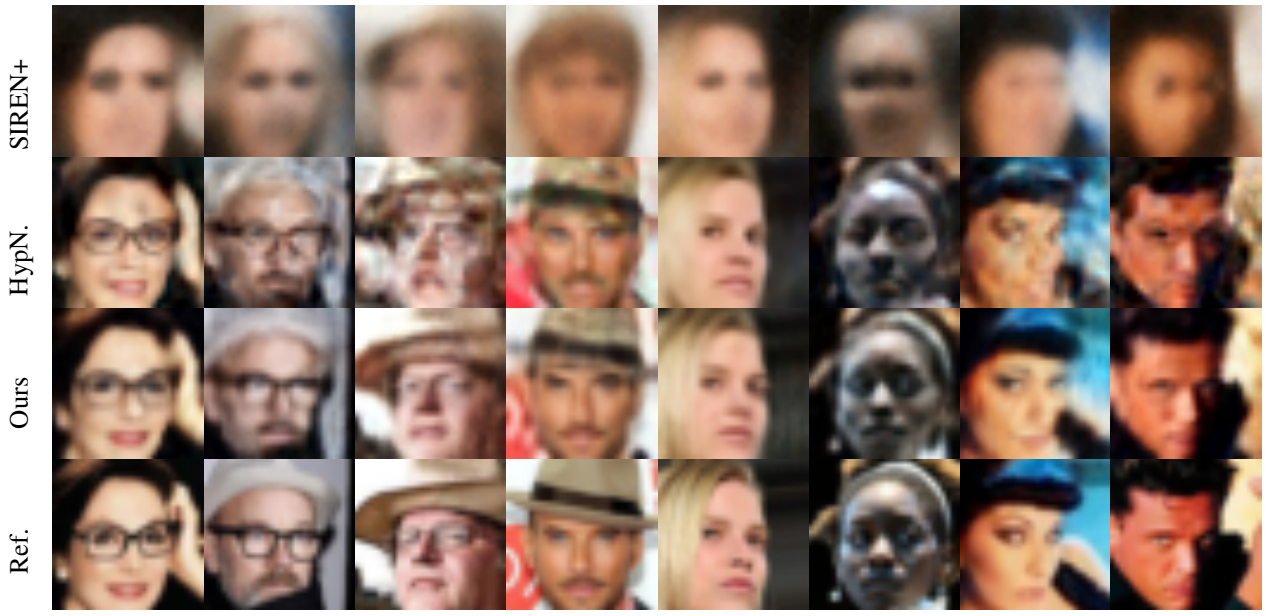


Figure A.7. Functional representation of CelebA images. We show qualitative comparisons with other neural representations which use periodic activations. SIREN [144] with the latent codes concatenated with the input pixel coordinates fails to recover sharp details. The HyperNet [53] (HypN.) conditioned SIREN [144] has $\sim 60\times$ more parameters.

the recommended $\sigma = 10$ value. Since in the case of HyperNet-SIREN, the last layer predicts *all* the parameters of the hyponetwork, it makes the last layer highly over-parameterized. This leads to *slow* training, *unstable* convergence and *inefficient* memory usage. We show reconstructions on CelebA test images in Figure A.7.

Shapes

Generative modeling of 3D shapes has recently been driven by implicit neural representations [124, 31] trained to regress a shape's signed distance field (SDF), by sampling discrete locations in the 3D space. The shape can be reconstructed from the learned SDF using sphere tracing [84] or marching cubes. We show that our model is a powerful replacement for the conditional ReLU-MLPs typically used for this application; it can encode SDFs more accurately. For this experiment, we sample 500K points for each shape in the `cars` category of ShapeNet [29]. Half these points are sampled close to the surface, the remaining are randomly sampled inside the unit sphere encompassing the shapes [55]. We use a similar training objective

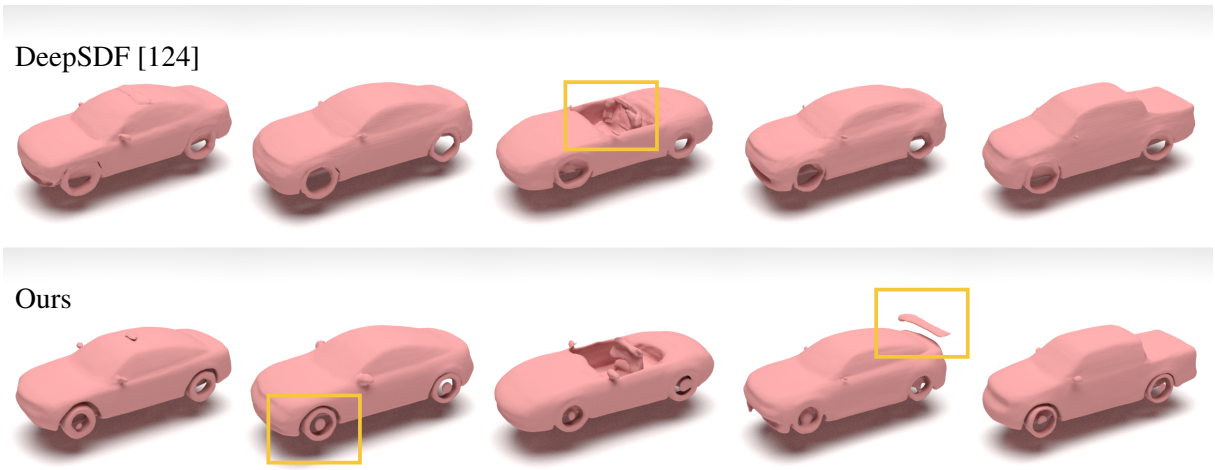


Figure A.8. Implicit shape encoding. Our model used as an auto-decoder with a latent code z_i assigned to each shape from the `cars` category in the ShapeNet [29] dataset. Compared to DeepSDF [124] which uses a single MLP with ReLU activations, our model is able to recover finer high-frequency details. Note the highlighted regions.

as in case of images (Eq. A.7), but we repalce the L_2 loss with an L_1 penalty in the fidelity term. Following [124], all conditional models arer trained in as auto-decoder for this experiment (§ A.3.4). Table A.3 shows quantitative comparisons in terms of bi-directional Chamfer distance, computed between the ground-truth shapes and the reconstructions. We show renderings in Figure A.8. Compared to DeepSDF [124], we produce higher-quality reconstructions, with finer details. As for images, we found SIREN+ does not converge. In this comparison, we do not include recent improvements that are orthogonal to our contribution, e.g., improvements to the spatial sampling [36], training procedure [141] or loss functions [50] These improvements would benefit our method as well as the baselines. Other implementation details are in the supplemental.

Videos

For videos, we train our model on ∼90K videos from the Vimeo-90k *septuplet* dataset [182]. Each video is 7 frames long and has a spatial resolution of 448×256. During training, we randomly crop $32 \times 32 \times 7$ tiles from the videos. We use a 3D convolutional encoder to predict the latent codes. At test time, the videos are structured in a grid and each tile is

Table A.3. Comparison of generalization ability on ShapeNet. Our model can be used in an auto-decoder setting, where the latent codes for each shape are optimized simultaneously with the network parameters. We implicitly represent 3514 car shapes from the ShapeNet [29] dataset and report the bi-directional Chamfer distance to the ground truth shapes.

	Chamfer Distance		
Method	Median ↓	Mean ↓	Std ↓
DeepSDF [124]	0.00284	0.00363	0.00559
DeepSDF + SIREN [124, 144]	*n/a*	*n/a*	*n/a*
DeepSDF + FFN [124, 158]	0.00399	0.00519	0.00757
Ours	0.00230	0.00273	0.00285

Table A.4. Comparison of generalization ability on Vimeo-90k. For videos, we observe trends similar to the image experiment. We report reconstruction PSNR on 7K videos from the Vimeo-90k dataset.

Method	PSNR ↑
ReLU	23.06
SIREN+ [144]	*n/a*
FFN [158]	19.38
HyperNet-SIREN [53, 144]	23.61
Ours	25.28

Table A.5. Encode high-res images faster. We use our model as a functional auto-encoder to encode images from Div2K [2] dataset. The global methods are *overfit* on each image separately and do not generalize. Our local methods are pre-trained and numbers are reported on a test set.

Method	Local	PSNR ↑	Overfit	Time ↓
ReLU		18.94	✓	∼60m
SIREN [144]		22.88	✓	∼60m
FFN [158]		28.48	✓	∼60m
Ours-ReLU	✓	34.73		13s
Ours	✓	38.03		15s

reconstructed with the estimated latent code. The train-test split is used as provided in the dataset.

We show quantitative comparisons in Table A.4.

A.4.2 Local Functional Representations

Images

Our dual-MLP model can also be used to generalize to high-resolution implicit functions. We train our model on 100 images from Div2K [2]. Each image has a long-side resolution of 2K and split into 32×32 overlapping tiles. We found 32×32 tile size to provide good reconstruction accuracy as well as good interpolation properties (as we show in the supplemental). The total number of tiles in the training set is $\sim$ 2M. Each tile is encoded using our method as an auto-encoder. At test time we sample 50 unseen images, and encode them using the trained model. Since other baselines do not generalize, we train a separate MLP for each of the images individually for global methods (*i.e.*ReLU, SIREN, FFN). Reconstruction PSNR at $1\times$ resolution is reported in Table A.5. Additionally, we perform an ablation on our model by using a standard ReLU MLP with our local parameterization. We show a qualitative comparison in the supplemental.



SIREN [144]-28.51dB Ours-30.55dB Ref

Figure A.9. SIREN struggles with high-frequencies in temporal domain. Our model faithfully reconstructs high-frequencies in both the spatial and temporal dimensions. We show the last frame of a video segment before a jump-cut to a different shot. This is a sharp transition in time, which SIREN struggles to model accurately.

Shape

For this experiment, we collect 18 high-resolution ($\sim$2M triangles) shapes from the the ThreedScans project [76]. These shapes are split into two `Scenes A` and `B`, each of which have 9 shapes. We compute a ground truth signed-distance-field (SDF) as in the global shapes experiment (§ A.4.1) for supervision. We use our model in an auto-decoder configuration. We train it on $48 \times 48 \times 48$ voxels extracted from `Scene A`. and we evaluate reconstruction accuracy

Table A.6. Implicit encoding of large 3D scenes. Our method can be used to approximate a signed-distance-field for large scenes. The global methods are overfit on each scene separately. Our local models are trained on `Scene A` and then generalized to `Scene B` by freezing the weights of the model and optimizing the latent codes.

		Chamfer Distance ($\times 10^{-5}$)	
Method	Local	`Scene A` (9 shapes)	`Scene B` (9 Shapes)
ReLU [124]		2.16	3.61
SIREN [124, 144]		2.00	7.36
FFN [124, 158]		1.93	3.54
Ours-ReLU	✓	1.37	8.37
Ours	✓	1.32	2.40

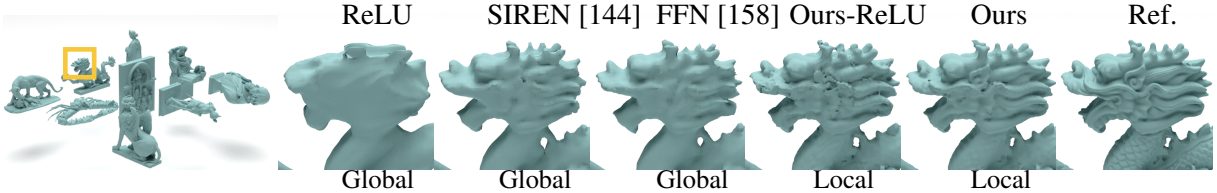


Figure A.10. Encoding large 3D scenes. Our method can be used to encode signed-distance-fields (SDFs) for large scenes (example shown on the left). The local representation enables the MLP f_θ to distribute its capacity sparsely in the scene. As a result, the estimated SDF is not overly *smooth* like we observe in the case of global methods.

Table A.7. Encoding high-res videos faster. We use our per-trained model on Vimeo-90k [182] to encode high-res videos faster. To achieve a similar reconstruction accuracy, overfitting other models take upto 15 hours.

Method	Local	PSNR ↑	Overfit	Time ↓
ReLU		19.28	✓	∼15hr
FFN [158]		20.87	✓	∼15hr
SIREN [144]		25.19	✓	∼15hr
Ours-Generalized	✓	25.21		∼1m

on `Scene B`, where we only optimize the latent codes. For our global baselines, we overfit the models individually for each scene. We extract meshes from the learned neural SDFs using marching cubes [91], and report the chamfer-distance from the ground truth to the reconstructions in Table A.6. In Figure A.10 we show `Scene A` renders using all the baselines and our method. We give implementation details in the supplemental.

Video

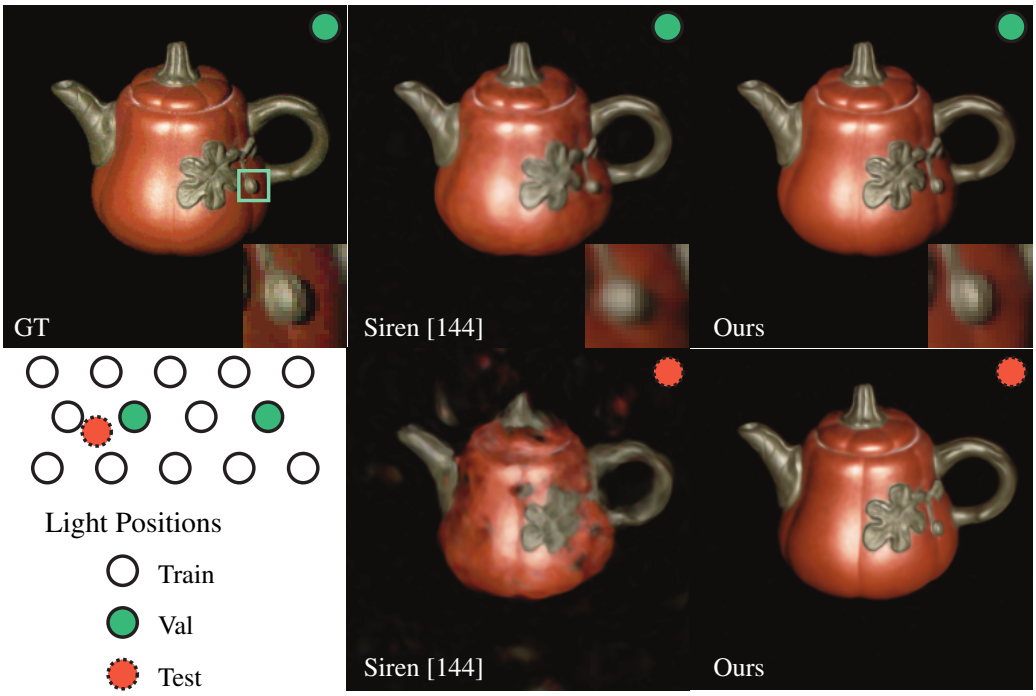
Similar to experiments shown in [144], we encode high-resolution videos using our method. We use 6 videos (`pexels.com`) with 1920×1080 resolution and downsample them to 640×256. Each video is split into a grid of $32 \times 32 \times 7$ tiles for the local model. The reconstruction PSNR is reported in Table A.7. Our model is pre-trained on Vimeo-90k [182] (§ A.4.1) and tested on the collected videos. It is able to achieve similar reconstruction accuracy as to the one obtained by previous methods while being $\sim$1000$\times$ faster. We found that SIRENs [144] struggle to reconstruct high-frequency content for complex and varied video signals, in both the spatial and time dimensions as shown in Figure A.9.

A.4.3 Image-based Relighting

We perform image-based relighting, where the input is a pixel coordinate $x \in \mathbb{R}^2$ and a lighting direction is taken as the latent code $z \in \mathbb{R}^3$. The output is radiance $y \in \mathbb{R}^3$. We use 90 cropped and aligned 300×300 images of a *real* scene, captured in an OLAT (One-Light-At-a-Time) setup [140]. The 90 images are divided into a (80, 10) train-val split. We train MLPs to learn the function $y_{ij} = f_\theta(x_{ij}; z_i)$, where y_{ij} is the radiance at x_{ij} with z_i as the light direction. A standard L_2 loss is used to train the parameters θ. At test time, we pick arbitrary z_i's within reasonable bounds and use f_θ to synthesize images with the corresponding light direction. For comparison, we implement SIREN+ [144] and a ReLU MLP with positional encodings (PE) [102] (we found them to work better than FFN) with naïve concatenation of domains, *i.e.* $[x_{ij}, z_i]$ as the input. Figure A.11 shows a scene relit using light-directions not present in the training set. Our method reconstructs light-dependent effects like shadows and specular highlights with higher fidelity. We find SIREN+ to struggle with interpolation in this experment.

A.5 Conclusion

We propose a novel method for representing signals using multi-layer perceptrons (MLPs). We show that partitioning the signal domain into tiles simplifies the signal locally. This leads to



Method	Train PSNR	Validation PSNR
SIREN+ [144]	40.92	39.54
PE [102]	42.27	41.06
Ours	43.36	43.71

Figure A.11. Image-based relighting using MLPs. Using light direction and image coordinates concatenated as input to a SIREN fails to reconstruct cast-shadows and specularities faithfully (insets). Our method of using conditional modulation reconstructs at higher fidelity with smoother interpolation in the light domain.

representing images, videos and shapes using MLPs with high-quality reconstructions. MLPs with ReLU activations fail to reconstruct high-frequency components of the signals. Instead, we use sine activations which we show to work with a wider frequency spectrum. Using local models requires MLPs to be conditioned on latent codes. We show that concatenating latent codes with the input hinders expressivity. Our method uses a dual-MLP architecture instead. The proposed model also generalizes to multiple instances of these signals. Our local parameterization is general enough to be applied in other applications that use implicit neural functions [102]. We merge local functions using n-linear blending which mitigates perceptual discontinuities for the

tasks that we explored; however, it is still unclear if that strategy can be applied to other function domains.

Appendix A, in full, is a reprint of the material as it appears in "Modulated Periodic Activations for Generalizable Local Functional Representations", Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, Manmohan Chandraker, International Conference on Computer Vision (ICCV), 2021. The dissertation author was the primary investigator and author of this paper.

Bibliography

[1] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

[2] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[3] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. 2020.

[4] Thiemo Alldieck, Hongyi Xu, and Cristian Sminchisescu. imghum: Implicit generative models of 3d human shape and articulated pose. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5461–5470, 2021.

[5] John Amanatides, Andrew Woo, et al. A fast voxel traversal algorithm for ray tracing. In *Eurographics*, volume 87, pages 3–10, 1987.

[6] Luigi Ambrosio and Halil Mete Soner. Level set approach to mean curvature flow in arbitrary codimension. *Journal of differential geometry*, 43(4):693–737, 1996.

[7] Samuel Amstutz and Antonio A Novotny. Topological optimization of structures subject to von mises stress constraints. *Structural and Multidisciplinary Optimization*, 41(3):407, 2010.

[8] James Arvo. The irradiance jacobian for partially occluded polyhedral sources. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 343–350, 1994.

[9] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020.

[10] Dejan Azinović, Tzu-Mao Li, Anton Kaplanyan, and Matthias Nießner. Inverse path tracing for joint material and lighting estimation. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

[11] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. *arXiv preprint arXiv:2104.04532*, 2021.

[12] Sai Bangaru, Michael Gharbi, Tzu-Mao Li, Fujun Luan, Kalyan Sunkavalli, Milos Hasan, Sai Bi, Zexiang Xu, Gilbert Bernstein, and Fredo Durand. Differentiable rendering of neural sdfs through reparameterization. In *ACM SIGGRAPH Asia 2022 Conference Proceedings*, 2022.

[13] Sai Bangaru, Tzu-Mao Li, and Frédo Durand. Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph.*, 39(6):245:1–245:18, 2020.

[14] Jonathan T. Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *TPAMI*, 2015.

[15] Cx K Batchelor and GK Batchelor. *An introduction to fluid dynamics*. Cambridge university press, 2000.

[16] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Deep reflectance volumes: Relightable reconstructions from multi-view photometric images. *arXiv preprint arXiv:2007.09892*, 2020.

[17] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[18] Mario Botsch and Leif Kobbelt. A remeshing approach to multiresolution modeling. In *Proceedings of Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 185–192, 2004.

[19] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon mesh processing*. CRC press, 2010.

[20] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE transactions on visualization and computer graphics*, 14(1):213–230, 2007.

[21] Mario Botsch, Robert Sumner, Mark Pauly, and Markus Gross. Deformation transfer for detail-preserving surface editing. In *Vision, Modeling & Visualization*, pages 357–364. Citeseer, 2006.

[22] Adrien Bousseau, Emmanuelle Chapoulie, Ravi Ramamoorthi, and Maneesh Agrawala. Optimizing environment maps for material depiction. In *Computer graphics forum*, volume 30, pages 1171–1180. Wiley Online Library, 2011.

[23] Martin Burger, Benjamin Hackl, and Wolfgang Ring. Incorporating topological derivatives into level set methods. In *Journal of Computational Physics*, 2004.

[24] Ana Carpio and Maria Luisa Rapún. *Topological Derivatives for Shape Reconstruction*, volume 1943 of *Lecture Notes in Mathematics*. 2008.

[25] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 694–699. IEEE, 1995.

[26] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. *arXiv preprint arXiv:2003.10983*, 2020.

[27] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5799–5809, 2021.

[28] Tony F Chan and Luminita A Vese. Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277, 2001.

[29] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[30] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[31] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[32] Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization*. John Wiley & Sons, 2004.

[33] Forrester Cole, Kyle Genova, Avneesh Sud, Daniel Vlasic, and Zhoutong Zhang. Differentiable surface rendering via non-differentiable sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6088–6097, 2021.

[34] Keenan Crane, Ignacio Llamas, and Sarah Tariq. Real-time simulation and rendering of 3d fluids. *GPU gems*, 3(1), 2007.

[35] Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. On the effectiveness of weight-encoded neural implicit 3d shapes. *arXiv preprint arXiv:2009.09808*, 2020.

[36] Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. Overfit neural networks as a compact shape representation. *arXiv preprint arXiv:2009.09808*, 2020.

[37] Amaël Delaunoy and Emmanuel Prados. Gradient Flows for Optimizing Triangular Mesh-based Surfaces: Applications to 3D Reconstruction Problems Dealing with Visibility. In *International Journal of Computer Vision*, 2011.

[38] Thomas Deliot, Eric Heitz, and Laurent Belcour. Transforming a non-differentiable rasterizer into a differentiable one with stochastic gradient estimation. *arXiv preprint arXiv:2404.09758*, 2024.

[39] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, 1999.

[40] Olivier Faugeras and Renaud Keriven. *Variational Principles, Surface Evolution, PDE's, Level Set Methods and the Stereo Problem*. IEEE, 2002.

[41] Michael Fischer and Tobias Ritschel. Plateau-reduced differentiable path tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4285–4294, 2023.

[42] Michael Fischer and Tobias Ritschel. Zerograds: Learning local surrogates for non-differentiable graphics. *ACM Transactions on Graphics (TOG)*, 43(4):1–15, 2024.

[43] Pau Gargallo, Emmanuel Prados, and Peter Sturm. Minimizing the Reprojection Error in Surface Reconstruction from Images. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2007.

[44] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020.

[45] Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. Inverse volume rendering with material dictionaries. *ACM Transactions on Graphics (TOG)*, 32(6):1–13, 2013.

[46] Purvi Goel, Loudon Cohen, James Guesman, Vikas Thamizharasan, James Tompkin, and Daniel Ritchie. Shape from tracing: Towards reconstructing 3d object geometry and svbrdf material from images via differentiable path tracing. In *2020 International Conference on 3D Vision (3DV)*, pages 1186–1195. IEEE, 2020.

[47] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[48] Lewis D Griffin. Scale-imprecision space. *Image and Vision Computing*, 15(5):369–398, 1997.

[49] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of Machine Learning and Systems 2020*. 2020.

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

[51] Johannes Günther, Ingo Wald, and Philipp Slusallek. Realtime caustics using distributed photon mapping. *Rendering Techniques*, 4:111–122, 2004.

[52] Kunal Gupta and Manmohan Chandraker. Neural mesh flow: 3d manifold mesh generation via diffeomorphic flows. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1747–1758. Curran Associates, Inc., 2020.

[53] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[54] Thomas C Hales. Jordan's proof of the jordan curve theorem. *Studies in logic, grammar and rhetoric*, 10(23):45–60, 2007.

[55] Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge Belongie. Dualsdf: Semantic shape manipulation using a two-level representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7631–7641, 2020.

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

[57] M Hintermuller and Antoine Laurain. Electrical impedance tomography: from topology to shape. *Control and Cybernetics*, 37(4):913–933, 2008.

[58] Douglas Hofstader. *Gödel, Escher, Bach: An Eternal Golden Braid*. 1979.

[59] Lars Hörmander. *The analysis of linear partial differential operators I: Distribution theory and Fourier analysis*. Springer, 2015.

[60] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. Mitsuba 3 renderer, 2022. https://mitsuba-renderer.org.

[61] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014.

[62] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020.

[63] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1251–1261, 2020.

[64] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[65] Hailin Jin. *Variational Methods for Shape Reconstruction in Computer Vision*. Washington University in St. Louis, 2003.

[66] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer Graphics and Interactive Techniques*, pages 143–150, 1986.

[67] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[68] Petr Kellnhofer, Lars C Jebe, Andrew Jones, Ryan Spicer, Kari Pulli, and Gordon Wetzstein. Neural lumigraph rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4287–4297, 2021.

[69] Pramook Khungurn, Daniel Schroeder, Shuang Zhao, Kavita Bala, and Steve Marschner. Matching real fabrics with micro-appearance models. *ACM Trans. Graph.*, 35(1):1–1, 2015.

[70] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[71] Sylwester Klocek, Łukasz Maziarka, Maciej Wołczyk, Jacek Tabor, Jakub Nowak, and Marek Śmieja. Hypernetwork functional image representation. In *International Conference on Artificial Neural Networks*, pages 496–510. Springer, 2019.

[72] Jan J Koenderink and Andrea J Van Doorn. The structure of locally orderless images. *International Journal of Computer Vision*, 31:159–168, 1999.

[73] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[74] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020.

[75] Alan Lapedes and Robert Farber. Nonlinear signal processing using neural networks: Prediction and system modelling. Technical report, 1987.

[76] Oliver Laric. Three d scans.

[77] Ignacio Larrabide, RA Feijóo, AA Novotny, and EA Taroco. Topological derivative: a tool for image processing. *Computers & Structures*, 86(13-14):1386–1403, 2008.

[78] Svetlana Lazebnik, Cordelia Schmid, Jean Ponce, et al. Spatial pyramid matching. *Object Categorization: Computer and Human Vision Perspectives*, 3(4), 2009.

[79] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018.

[80] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):193:1–193:15, 2020.

[81] Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. 39(6):1–15.

[82] Hsueh-Ti Derek Liu, Francis Williams, Alec Jacobson, Sanja Fidler, and Or Litany. Learning smooth neural functions via lipschitz regularization. *arXiv preprint arXiv:2202.08345*, 2022.

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

[84] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2019–2028, 2020.

[85] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

[86] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7708–7717, 2019.

[87] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[88] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019.

[89] Matthew M Loper and Michael J Black. Opendr: An approximate differentiable renderer. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII 13*, pages 154–169. Springer, 2014.

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

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

[92] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019.

[93] Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. Unified shape and svbrdf recovery using differentiable monte carlo rendering. *ArXiv*, 2021.

[94] Dhruv Mahajan, Ira Kemelmacher Shlizerman, Ravi Ramamoorthi, and Peter Belhumeur. A theory of locally low dimensional light transport. In *ACM SIGGRAPH 2007 papers*, pages 62–es. 2007.

[95] George H Markstein. *Nonsteady flame propagation: AGARDograph*. Elsevier, 2014.

[96] Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation. *ACM Trans. Graph. (SIGGRAPH)*, 40(4), 2021.

[97] Ishit Mehta, Manmohan Chandraker, and Ravi Ramamoorthi. A level set theory for neural implicit evolution under explicit flows. In *European Conference on Computer Vision (ECCV)*, pages 711–729. Springer, 2022.

[98] Ishit Mehta, Manmohan Chandraker, and Ravi Ramamoorthi. A theory of topological derivatives for inverse rendering of geometry. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 419–429, October 2023.

[99] Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, and Manmohan Chandraker. Modulated periodic activations for generalizable local functional representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14214–14223, October 2021.

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

[101] Luke Metz, C Daniel Freeman, Samuel S Schoenholz, and Tal Kachman. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2021.

[102] 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. *arXiv preprint arXiv:2003.08934*, 2020.

[103] 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. *Communications of the ACM*, 65(1):99–106, 2021.

[104] John Milnor and David W Weaver. *Topology from the differentiable viewpoint*, volume 21. Princeton university press, 1997.

[105] Hans-Peter Seidel Mojtaba Bemana, Karol Myszkowski and Tobias Ritschel. X-fields: Implicit neural view-, light- and time-image interpolation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2020)*, 39(6), 2020.

[106] Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, and Chris Olah. Differentiable image parameterizations. *Distill*, 2018. https://distill.pub/2018/differentiable-parameterizations.

[107] Jiteng Mu, Weichao Qiu, Adam Kortylewski, Alan Yuille, Nuno Vasconcelos, and Xiaolong Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13001–13011, 2021.

[108] Ken Museth, David E Breen, Ross T Whitaker, and Alan H Barr. Level set surface editing operators. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 330–338, 2002.

[109] Shree K Nayar, Katsushi Ikeuchi, and Takeo Kanade. Shape from interreflections. *International Journal of Computer Vision*, 6:173–195, 1991.

[110] Serguei A Nazarov and Jan Sokołowski. Asymptotic analysis of shape functionals. *Journal de Mathématiques pures et appliquées*, 82(2):125–196, 2003.

[111] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. Ieee, 2011.

[112] Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. Large steps in inverse rendering of geometry. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 40(6), December 2021.

[113] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[114] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020.

[115] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6), December 2019.

[116] Antonio André Novotny, Jan Sokołowski, and Antoni Żochowski. *Applications of the topological derivative method*. Springer, 2019.

[117] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5589–5599, 2021.

[118] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[119] Shaul Oron, Aharon Bar-Hillel, Dan Levi, and Shai Avidan. Locally orderless tracking. *International Journal of Computer Vision*, 111:213–228, 2015.

[120] Stanley Osher and Ronald P Fedkiw. Level set methods: an overview and some recent results. *Journal of Computational physics*, 169(2):463–502, 2001.

[121] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 1988.

[122] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Taming the waves: sine as activation function in deep neural networks. 2016.

[123] 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 (CVPR)*, June 2019.

[124] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[125] Ofir Pele and Michael Werman. The quadratic-chi histogram distance family. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part II 11*, pages 749–762. Springer, 2010.

[126] Ken Perlin. Improving noise. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 681–682, 2002.

[127] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[128] Henri Poincaré. Science and hypothesis., 1905.

[129] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[130] Inigo Quilez.

[131] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[132] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500, 2001.

[133] Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. A first-order analysis of lighting, shading, and shadows. *ACM Transactions on Graphics (TOG)*, 26(1):2–es, 2007.

[134] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[135] Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Meshsdf: Differentiable iso-surface extraction. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22468–22478. Curran Associates, Inc., 2020.

[136] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022.

[137] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[138] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *arXiv preprint arXiv:2110.06635*, 2021.

[139] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *arXiv preprint arXiv:2007.02442*, 2020.

[140] Boxin Shi, Zhe Wu, Zhipeng Mo, Dinglong Duan, Sai-Kit Yeung, and Ping Tan. A benchmark dataset and evaluation for non-lambertian and uncalibrated photometric stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3707–3716, 2016.

[141] Vincent Sitzmann, Eric Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[142] Vincent Sitzmann, Eric R. Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. In *Proc. NeurIPS*, 2020.

[143] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *arXiv*, 2020.

[144] Vincent Sitzmann, Julien NP Martel, Alexander W Bergman, David B Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *arXiv preprint arXiv:2006.09661*, 2020.

[145] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

[146] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, pages 1121–1132, 2019.

[147] Jan Sokolowski and Antoni Zochowski. On the topological derivative in shape optimization. *SIAM journal on control and optimization*, 37(4):1251–1272, 1999.

[148] Jan Erik Solem and Niels Chr. Overgaard. A Geometric Formulation of Gradient Descent for Variational Problems with Moving Surfaces. In *Scale Space and PDE Methods in Computer Vision*, volume 3459 of *Lecture Notes in Computer Science*, pages 419–430. 2005.

[149] Josep M Sopena, Enrique Romero, and Rene Alquezar. Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. 1999.

[150] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007.

[151] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, 2004.

[152] Joe Staines and David Barber. Variational optimization. *arXiv preprint arXiv:1212.4507*, 2012.

[153] Kenneth O Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007.

[154] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6229–6238, 2021.

[155] Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In *International Conference on Machine Learning*, pages 20668–20696. PMLR, 2022.

[156] E. Süli and D.F. Mayers. *An Introduction to Numerical Analysis*. An Introduction to Numerical Analysis. Cambridge University Press, 2003.

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

[158] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

[159] Jiaxiang Tang. Stable-dreamfusion: Text-to-3d with stable-diffusion, 2022. https://github.com/ashawkey/stable-dreamfusion.

[160] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358, 1995.

[161] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, 1987.

[162] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. *arXiv preprint arXiv:2111.05849*, 2021.

[163] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[164] Joost van de Weijer and Theo Gevers. Color mode filtering. In *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, volume 1, pages 125–128. IEEE, 2001.

[165] Joost Van de Weijer and Rein Van den Boomgaard. Local mode filtering. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 2, pages II–II. IEEE, 2001.

[166] Bram van Ginneken and Bart M ter Haar Romeny. Applications of locally orderless images. *Journal of Visual Communication and Image Representation*, 11(2):196–208, 2000.

[167] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[168] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Differentiable signed distance function rendering. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022.

[169] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Path replay backpropagation: Differentiating light paths using constant memory and linear time. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 40(4), 2021.

[170] 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. *arXiv preprint arXiv:2106.10689*, 2021.

[171] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[172] Zichen Wang, Xi Deng, Ziyi Zhang, Wenzel Jakob, and Steve Marschner. A simple approach to differentiable rendering of sdfs. *arXiv preprint arXiv:2405.08733*, 2024.

[173] Greg Ward and Paul S Heckbert. Irradiance gradients. 1992.

[174] Ross T Whitaker. A level-set approach to 3d reconstruction from range data. *International journal of computer vision*, 29(3):203–231, 1998.

[175] Andrew Witkin, Demetri Terzopoulos, and Michael Kass. Signal matching through scale space. *International Journal of Computer Vision*, 1(2):133–144, 1987.

[176] Andrew P Witkin. Scale-space filtering. In *Readings in computer vision*, pages 329–332. Elsevier, 1987.

[177] Daniel N Wood, Daniel I Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 287–296, 2000.

[178] Liwen Wu, Rui Zhu, Mustafa B Yaldiz, Yinhao Zhu, Hong Cai, Janarbek Matai, Fatih Porikli, Tzu-Mao Li, Manmohan Chandraker, and Ravi Ramamoorthi. Factorized inverse path tracing for efficient and accurate material-lighting estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3848–3858, 2023.

[179] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *arXiv preprint arXiv:2111.11426*, 2021.

[180] Jiankai Xing, Fujun Luan, Ling-Qi Yan, Xuejun Hu, Houde Qian, and Kun Xu. Differentiable Rendering Using RGBXY Derivatives and Optimal Transport. *ACM Transactions on Graphics*, 41(6):1–13, December 2022.

[181] Yinghao Xu, Sida Peng, Ceyuan Yang, Yujun Shen, and Bolei Zhou. 3d-aware image synthesis via learning structural and textural representations. 2021.

[182] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision (IJCV)*, 127(8):1106–1125, 2019.

[183] Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. Geometry processing with neural fields. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[184] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34, 2021.

[185] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020.

[186] Greg Zaal.

[187] Rhaleb Zayer, Christian Rössl, Zachi Karni, and Hans-Peter Seidel. Harmonic guidance for surface deformation. In *Computer Graphics Forum*, volume 24, pages 601–609. Citeseer, 2005.

[188] Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. Path-space differentiable rendering. *ACM Trans. Graph.*, 39(4):143:1–143:19, 2020.

[189] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[190] Hong-Kai Zhao, Tony Chan, Barry Merriman, and Stanley Osher. A variational level set approach to multiphase motion. *Journal of computational physics*, 127(1):179–195, 1996.

[191] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. *ACM Transactions on Graphics (TOG)*, 27(5):1–11, 2008.

[192] Zhiqian Zhou, Cheng Zhang, Zhao Dong, Carl Marshall, and Shuang Zhao. Estimating uncertainty in appearance acquisition. 2024.

[193] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. *arXiv preprint arXiv:2112.12130*, 2021.