Recovering and Creating 3D Experiences from Casual Data

Ethan Weber

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-75
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-75.html

May 15, 2025

Recovering and Creating 3D Experiences from Casual Data

by

Ethan Weber

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Angjoo Kanazawa, Chair
Professor Alyosha Efros
Professor Jitendra Malik
Professor Antonio Torralba

Spring 2025

Recovering and Creating 3D Experiences from Casual Data

Abstract

Recovering and Creating 3D Experiences from Casual Data

by

Ethan Weber

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Angjoo Kanazawa, Chair

In today's world, we are surrounded by casually captured visual data—photos and videos from our phones, TV shows, cartoons, social media clips, and more. These formats depict rich, physical 3D spatiotemporal worlds, even though, when you look closely, the frames themselves are often not necessarily geometrically consistent or explicitly 3D. Yet, as viewers, we effortlessly perceive the underlying structure, intuitively reconstructing the spaces and stories they represent. We could even imagine what might lie in the space behind the camera, where the photographer is standing. Why is it that humans can so easily make sense of these visual experiences, while machines still struggle to recover or create 3D from such data? This thesis aims to bridge that gap, bringing the human-like ability to recover and create 3D experiences from casual data to machines. We develop new methods that robustly reconstruct 3D environments from unstructured, in-the-wild imagery, such as videos you took with your smartphone, and introduce generative techniques to complete missing regions and hallucinate plausible content where data is sparse or absent. Our work advances the state of the art in neural rendering, scene completion, and generative modeling, with contributions including open-source frameworks, new methods for artifact removal and generative scene completion, and the first large-scale 3D reconstruction of television shows and hand-drawn cartoons. By bridging the gap between 3D reconstruction and generation, this thesis explores new possibilities for experiencing and understanding the visual world—no matter how casual or unconventional the data may be.

To my parents,

thank you for your unwavering love and support

Contents

List of Figures

List of Tables

Acknowledgments

I feel incredibly blessed to have such wonderful and caring people in my life. Thank you to Professor Angjoo Kanazawa for being the greatest PhD advisor I could ever ask for. I vividly remember the day you called me, excitedly telling me I was admitted to Berkeley. Your lab has felt like a family ever since I joined, and I'm so grateful for everything you've done to enable my PhD experience. Angjoo leads with kindness, constantly uplifts her students, and is also a brilliant, energetic researcher. I've really appreciated her support during my PhD—not just in becoming a better researcher, but also in learning how to be a better person. The way she cares for her family, Austin and Ayuna, is also inspiring. I'm so thankful to have been advised by her, and so excited for any students who have the opportunity to be advised by her in the future.

Thank you to my full committee: Angjoo Kanazawa, Alyosha Efros, Jitendra Malik, and Antonio Torralba. Alyosha and Jitendra—it's been a great privilege to be part of the UC Berkeley community and to learn from the wisdom and insight you both share. The Berkeley community is nothing short of amazing, and I'm so thankful to have crossed paths with and learned from the environment you've helped create. I was also fortunate to be part of Antonio Torralba's lab toward the end of my undergrad and master's studies. He really showed me how joyful and creative the research experience could be. I remember that during COVID, my favorite times of the week were the group meetings in his lab. I learned so much, and similar to Angjoo, Antonio was someone who consistently encouraged and uplifted me.

I want to thank my collaborators too. My biggest takeaway from the PhD is: work with friends. Working with friends makes the highs incredibly high and softens the lows. A PhD is an eventful ride with ups and downs, and doing it with friends makes it all the more fun. I was so lucky to have crossed paths with Matt Tancik, who I learned so much from. From your research insights to your work-life balance, I was genuinely inspired by you to grow in both research and personal life. The Nerfstudio and "hang in there" days are some of my greatest PhD memories. George Pavlakos, thank you for welcoming me into Berkeley and treating me with such kindness as a first-year. It was truly an honor to learn alongside you as we pushed an exciting project on reconstructing TV shows—it was the perfect introduction to Berkeley. Frederik Warburg, I'm so glad you passed through Berkeley and that we started hanging out just as Nerfstudio was wrapping up. That was a really uncertain phase of my PhD, and our collaboration on Nerfbusters re-energized me and helped reignite my drive to keep working in this space of casual data reconstruction. Riley Peterlinz, I'm so glad you joined the cartoon project—it turned into an incredible journey together. It's been an honor getting to know you and see you grow. Thanks for all the back-and-forth as we pushed each other with our complementary skill sets. I'm eager to see what you do next.

I want to thank my internship managers—Abhishek Kar and Noah Snavely at Google, and Christian Richardt and Michael Zollhöfer at Meta. I somehow keep encountering such kind, generous people to work with, and you are all great examples of that. Abhishek and Noah, thank you for being so enthusiastic to work together when I emailed you both as an early PhD student. Thank you both for supporting me as I worked on pushing the direction

of scene completion. Christian, thanks for the many thoughtful chats and ongoing support. Michael, thank you for always backing me. I've learned so much from each of you about how to support and manage others, and I hope to carry those lessons forward into the next chapter. I'd also like to express my gratitude to Professor Ren Ng for the opportunity to serve as a TA in his graphics course. I gained valuable insights from how you run your class and think about teaching, which I will carry forward.

It's been a great pleasure being part of the KAIR lab. Thank you to Justin, Brent, David, Chung Min, Aleks Holynski, Songwei, Hang, Ruilong, Shubham, Vickie, Lea, Vongani, Tyler, Hongsuk, Haven, and everyone I've interacted with in the lab over the years! I'm deeply grateful to all my collaborators at UC Berkeley and beyond who have contributed to my research journey. Your insights, support, and friendship have been instrumental in shaping my work and making this PhD experience so rewarding. Thank you for the countless discussions, brainstorming sessions, and collaborative efforts that have enriched my academic path.

Before UC Berkeley, I'm thankful to have overlapped with Dim, Ferda, Agata, Manel, Jonas, while working in Antonio Torralba. Dim Papadopoulos – thanks for first introducing me to the research process and mentoring me to write my first two papers. I learned so much through this process, and thanks Ferda and Agata for being part of this project too, and thanks Manel and Jonas for working with me on my first multi-view research paper (which ended up in my Master's thesis).

Thanks to my housemates over the years during PhD. Jessy and Sarah, we had a great run hosting parties and bringing our classes closer together. Tript, we had a fun summer and some great burrito runs! Dekel, it was really fortunate to stay with you and get to know you. I'm excited this short sublet led to a great friendship, and I hpe we can do more fishing trips!

Thanks to my grad friends, who have been an incredible support system throughout this journey. I'm grateful to Ale and Wanda, Suzie, Medhini, Michael, Dani, Alvin, Lisa, Daniel, Simeon, Jane, Sanjay, Neerja, Ritwik, Ameesh, Sasha, Eric, Amil, Ajay, and Devin for the memories we've shared. Your friendship has made this PhD experience so much richer and more meaningful. Also, my college friends mean so much to me and have been a constant source of support throughout my journey. I'm deeply grateful to Avery, Ravi, and Moin for their unwavering friendship and the countless memories we've shared. Pramoda and Rishi have been wonderful friends to see during my time in the Bay Area, and Pramoda especially during our MIT days. Matt Tung and Rahul have been incredible friends since day 1 at MIT, and I always look forward to our next time in the same location. I'm also thankful for the wonderful friendships with Wilbur, Kenny, Neeraj, Jordan, Kaveri, Gabe and Marla, Rose, Luke Melas, and Jingwei. Each of you has contributed uniquely to my life. I'm also especially thankful to Atsu; tackling our graduation requirements together was so much fun, and meeting you in the computational color was truly serendipitous. I'm grateful for my high school friends who have remained close throughout the years, especially Collin and Ally, Lu and Malissa, and Trent. Their friendship has been a constant source of joy and encouragement, especially when I get to see them when visiting home.

A special shoutout goes to Luke Roberto, who has been both a research inspiration to me and a supporter of all that I do. I'm so glad we had two summers together during our internships. You're as brilliant as you are kind, and your curiosity for research topics inspired me when I was early on in undergrad. I'm also grateful to Nikhil Punwaney, who is always there to support me. I'm inspired by your love towards others; it's truly unparalleled. Also, Ross Finman—thanks for always supporting me. I'm deeply inspired by your drive to build things while still keeping family a top priority, and I've really enjoyed our quick chats here and there.

I'm also thankful for my internship friends who made those experiences so memorable: Yash and Prat, Artem, Tobias, Aggelina, Frank, Vasu, Chen, Jiye, and Nikhil Keetha. Your camaraderie and collaboration during those periods made the Meta internship in Pittsburgh truly enjoyable, and I'm grateful to have you as friends beyond our professional lives.

Thanks also to important people who have helped along the way in formative years of my life. Thanks to my robotics mentors and team, especially John Boyle and Mark Holshuh—those were some fun years with Fondy Fire! I'm also grateful to Donna Nett-Heus, Leonard Speiser, Pete Florence, Russ Tedrake, and Soohyun Bae for their guidance, mentorship, and support during critical stages of my life leading up to this PhD.

Of course, a big thank you goes to my best friend and adventure buddy, Evonne Ng. You constantly remind me what truly matters in life, and I'm so grateful to go through life's ups and downs with you by my side. Thanks also to Evonne's family—especially her parents and her sisters, Elaine and Eley. And thank you to James and Uday as well. This PhD wouldn't have been the same without y'all. It's also been so fun hanging out with our "family friends": Colin, Zach, Amir and Hadar, Yossi and Liraz.

Last and most importantly, thank you to my parents, Dan and Karen Weber. You are the reason I'm able to pursue the PhD and my love of building. You've enabled me for as long as I have memories, and even though I've left Wisconsin, I don't feel so far away at all, considering how much we chat. I love calling home all the time whenever I get the chance. I'm also so thankful for my brothers, Alex and Isaac, and for Amanda, my sister-in-law. You make visiting home so fun!

Thanks also to my grandparents—Judy and Richard Belke, and Mike and Kathy Weber. My grandfathers have passed, but all four of my grandparents have had a significant impact on my life and helped shape my deep care for family. Grandma Belke (Judy) even helped me build many of my projects before I went off to MIT for undergrad—I still remember my excitment when we sewed the hovercraft skirt together. I'm also thankful for my extended family. Veronica, thanks for always being supportive, helping me with applications, and sending such kind messages to me. Thanks to my uncle and aunt (and godparents), Doug and Carolyn Weber, and their family, for supporting me over the years—and especially lately, for spending time together in Pittsburgh. Thanks to all my relatives and family, including all my uncles/aunts/cousins!

Above all, thank you, Mom and Dad, for your endless love and encouragement—and for setting such a strong example for me. I couldn't have done this without you.

Chapter 1

Introduction

Have you ever binge-watched a TV show and, by the end, felt like you could walk through the set with your eyes closed? Or maybe you've watched a scene jump between camera angles and still instantly understood where everyone was in the room. Perhaps you've looked at a hand-drawn cartoon and, without thinking, imagined the 3D world it depicts. Or, if you think back to your childhood home, you could probably sketch a floorplan from memory—even if you haven't been there in years.

Our brains are wired to effortlessly reconstruct and imagine the spaces around us, filling in gaps and making sense of the world from just a few visual cues. This remarkable ability to recover and create 3D experiences from everyday, unstructured visual content is something we take for granted. The central aim of this thesis is to enable machines to do the same: **recovering and creating 3D experiences from casual data**—that is, from the everyday, unstructured visual content that surrounds us.

More specifically, this thesis dives into the challenge of 3D reconstruction from what we call "casual data": the messy, everyday stuff that was never meant for 3D—yet somehow, to us, it just works. Think of shaky phone videos from a birthday party, binge-worthy TV shows, hand-drawn cartoons like SpongeBob SquarePants, Airbnb listings with just a handful of photos, home movies, Instagram reels, or even footage from natural disasters that are unfolding. Whenever there are multiple images or frames showing the same scene from different angles, our brains can piece together the 3D world behind them. However, this is not so easy for computers! Most current methods struggle when faced with this kind of wild, unstructured data and leave unseen areas missing rather than generating what could plausibly be there. This thesis takes on these challenges, aiming to make 3D reconstruction more robust and complete for in-the-wild, casual data.

1.1 Reconstruction Overview

3D reconstruction is the process of inferring the structure of a scene from multiple images. The typical pipeline begins with structure-from-motion (SfM) to estimate camera poses and

sparse geometry [192], followed by multi-view stereo methods to recover dense structure [193, 3]. In this thesis, we leverage state-of-the-art techniques such as Neural Radiance Fields (NeRF) [138] and 3D Gaussian Splatting (3DGS) [100], which are used throughout our work. Our contributions include building a flexible framework for NeRF development, improving robustness to casual data, and developing methods that succeed where traditional tools like COLMAP [192] fail—such as reconstructing hand-drawn cartoons in 3D.

1.2 Generation Overview

Generative models learn the distribution of existing data and can be used to fill in missing details or hallucinate unseen regions at inference time. In the context of scene reconstruction, this is known as "scene completion": completing a 3D scene from incomplete or casually captured input. For example, we may want to fill in occluded corners of a room, generate plausible views from novel camera angles, or infer what lies beneath a table. There is a synergistic relationship between 3D reconstruction and generation, and many works in this thesis explore this intersection—using generative models to enhance and complete reconstructed scenes [181, 211, 165, 78].

1.3 Related Work and Context

The field of 3D reconstruction and scene understanding has a rich history, with early work focusing on controlled, static environments and carefully captured data. Traditional pipelines rely on structure-from-motion (SfM) and multi-view stereo (MVS) to recover geometry [192, 193, 3], but these methods often break down when faced with casual, in-the-wild data that is noisy, incomplete, or not intended for 3D reconstruction. Recent advances in neural rendering, such as Neural Radiance Fields (NeRF) [138], have enabled more flexible and robust reconstructions, but still face challenges with artifact removal, missing data, and generalization to diverse sources like TV shows or cartoons [132, 161].

Generative models have also been explored for scene completion and novel view synthesis [181, 165, 78], but ensuring multi-view consistency and semantic plausibility remains difficult. Prior work has typically focused on either reconstruction or generation in isolation, or on narrow domains with strong assumptions about data quality [138, 11, 192, 193, 3, 135, 100]. In contrast, this thesis aims to bridge the gap between reconstruction and generation, and to push these methods into more casual, unconstrained domains. By developing new frameworks, algorithms, and datasets, we advance the state of the art in both robustness and generality, and open new directions for 3D scene understanding from everyday visual data.

1.4 Dissertation Overview

This thesis is organized into six technical chapters, each presenting a major contribution, followed by a concluding chapter. Below, we briefly summarize the content, focus, and key contributions of each chapter.

Chapter 2: Nerfstudio

Nerfstudio is an open-source framework designed to make NeRF research and development more accessible and collaborative. We observed that many NeRF papers were developed in isolation, hindering progress and reproducibility. This chapter details the design goals and capabilities of Nerfstudio, highlighting how it consolidates research, accelerates development, and supports a growing community of contributors. **We present an extensible, modular codebase, a real-time web viewer, and a new dataset for benchmarking NeRF methods.**

Chapter 3: Nerfbusters

Nerfbusters addresses the problem of removing artifacts from 3D reconstructions, particularly those caused by casual or imperfect captures. We introduce methods to identify and eliminate common reconstruction errors, improving the visual quality and reliability of NeRF-based models in real-world scenarios. **We present new algorithms for artifact removal and demonstrate improved reconstructions on a new dataset using a novel evaluation protocol..**

Chapter 4: Nerfiller

Nerfiller presents a generative approach to 3D scene completion. By leveraging 2D generative models and novel inpainting strategies, we enable the completion of missing or occluded regions in 3D scenes. This chapter explores how to distill multi-view inpainted images into a consistent 3D representation, allowing for plausible and diverse scene completions. **We present a new method for multi-view consistent 3D inpainting, validated on a range of incomplete scenes.**

Chapter 5: Fillerbuster

Fillerbuster builds on the limitations of Nerfiller, introducing a new model for scene completion that further enhances both consistency and realism. In particular, Fillerbuster introduces a model specifically trained for casual capture scene completion, jointly modeling camera pose and images within a diffusion framework [74, 181]. This enables more robust and consistent completion of real-world, in-the-wild scenes. We propose new techniques for integrating generative priors and optimizing 3D geometry, pushing the boundaries of what

is possible in 3D inpainting and scene completion. **We present an improved pipeline for 3D scene completion, with a diffusion model that jointly reasons about pose and appearance, and demonstrate quantitative and qualitative gains over prior approaches.**

Chapter 6: Sitcoms3D

Sitcoms3D tackles the challenge of reconstructing 3D environments and human poses from TV shows. By aggregating information across entire seasons, we recover camera parameters, scene structure, and actor locations, enabling applications such as re-identification, gaze estimation, and cinematography analysis. This chapter demonstrates the power of leveraging repetition and context in large video collections for 3D understanding. **We present the first large-scale 3D reconstructions of TV show environments and people, along with new applications and analyses.**

Chapter 7: Toon3D

Toon3D extends 3D reconstruction to the domain of cartoons and hand-drawn animation, where geometric consistency is often intentionally violated for artistic effect. We introduce a deformable optimization framework that aligns and reconstructs scenes from geometrically inconsistent images, enabling novel-view synthesis and immersive experiences from creative media. **We present the first framework and dataset for 3D reconstruction from multiple hand-drawn images depicting the same scene, and demonstrate novel-view synthesis from artistic media.**

Chapter 8: Conclusion

The concluding chapter synthesizes the key findings and contributions of this thesis, reflecting on the challenges of 3D reconstruction from casual data and the opportunities for future research in 3D reconstruction and generative scene completion.

Chapter 2

Nerfstudio: A reconstruction framework

Neural Radiance Fields (NeRF) are a rapidly growing area of research with wide-ranging applications in computer vision, graphics, robotics, and more. In order to streamline the development and deployment of NeRF research, we propose a modular PyTorch framework, Nerfstudio. Our framework includes plug-and-play components for implementing NeRF-based methods, which make it easy for researchers and practitioners to incorporate NeRF into their projects. Additionally, the modular design enables support for extensive real-time visualization tools, streamlined pipelines for importing captured in-the-wild data, and tools for exporting to video, point cloud and mesh representations. The modularity of Nerfstudio enables the development of Nerfacto, our method that combines components from recent papers to achieve a balance between speed and quality, while also remaining flexible to future modifications. To promote community-driven development, all associated code and data are made publicly available with open-source licensing at https://nerf.studio.

2.1 Introduction

Neural Radiance Fields (NeRFs) [138] are gaining popularity for their ability to create 3D reconstructions in real-world settings, with rapid research in the area pushing the field forward. Since the introduction of NeRFs in 2020, there has been an influx of papers focusing on advancements to the core method including few-image training [268, 238], explicit features for editing [120, 231, 277], surface representations for high-quality 3D mesh exports [153, 264, 236], speed improvements for real-time rendering and training [52, 208, 145], 3D object generation [165], and more [259].

These research innovations have driven interests in a wide variety of disciplines in both academia and industry. Roboticists have explored using NeRFs for manipulation, motion planning, simulation, and mapping [101, 2, 42, 20, 287, 200]. NeRFs are also explored for

[*]Denotes equal contribution

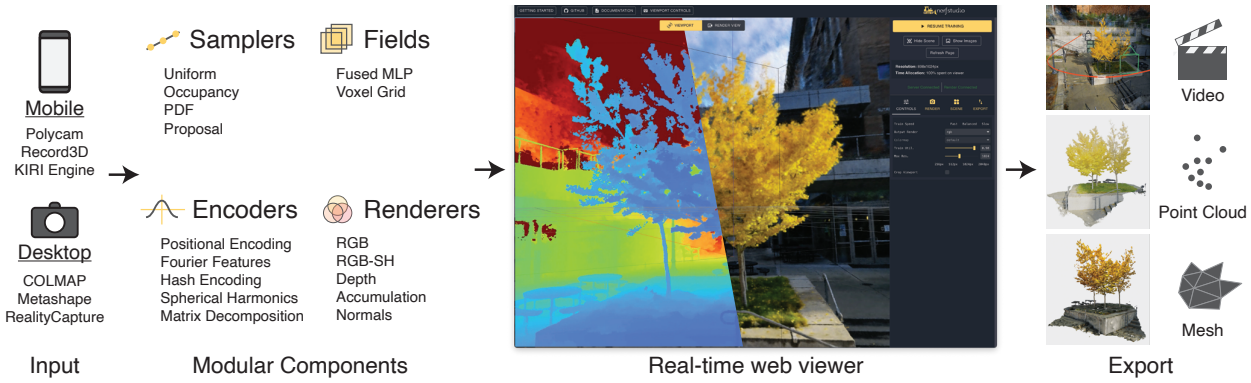

Figure 2.1: **Nerfstudio framework.** Nerfstudio is a Python framework for Neural Radiance Field (NeRF) development. Nerfstudio supports multiple input data pipelines, is built around multiple modular core NeRF components, integrates with a real-time web viewer, and supports multiple export modalities. The goal of the Nerfstudio framework is to simplify the development of custom NeRF methods, processing of real-world data, and interacting with reconstructions.

tomography applications [183], as well as perceiving people in videos [161]. Visual effects and gaming studios are exploring the technology for production and digital asset creation. News outlets capture NeRF portraits to tell stories in new formats [249]. The potential applications are vast, and even startups [*] are emerging to focus on deploying this technology.

Despite the growing use of NeRFs, support for development is still rudimentary. Due to the influx of papers and lack of code consolidation, tracking progress is difficult. Many papers implement features in their own siloed repository. This complicates the process of transferring features and research contributions across different implementations. Additionally, few tools exist to easily run NeRFs on real-world data collected by users. To address these challenges, we present Nerfstudio (Fig. 2.1), a modular framework that consolidates NeRF research innovations and makes them easier to use in real-world applications.

Furthermore, while NeRFs solve an inherently visual task, there is a lack of comprehensive and extensible tools for visualizing and interacting with NeRFs trained on real-world data. Despite the availability of several NeRF repositories, existing implementations are often focused on achieving state-of-the-art results on metrics such as PSNR, SSIM, and LPIPS. These evaluations are typically based on held-out images along the capture trajectory that are similar to the training images. This often makes them misleading indicators of performance for many real-world applications when data is captured in unstructured environments and novel views are rendered with large baselines. Qualitative evaluations have historically been a challenge due to the computational demands of NeRF, which often resulted in rendering times up to multiple seconds per image. Recent developments such as

[*]https://lumalabs.ai/

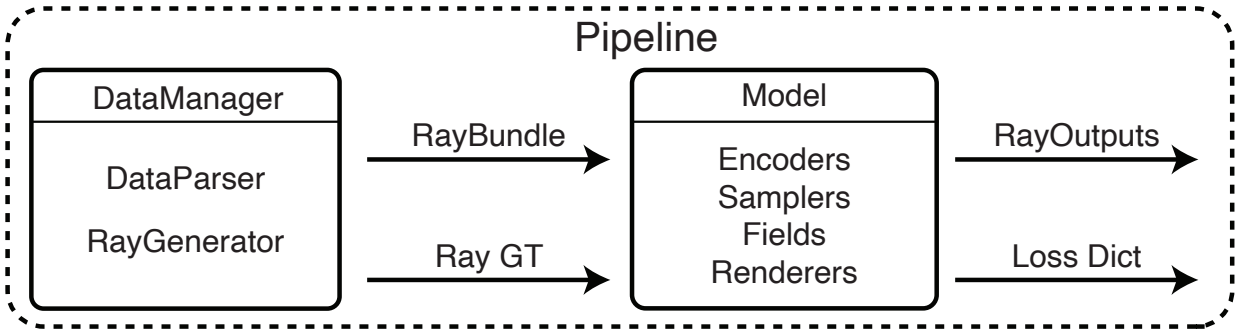

Figure 2.2: **Pipeline components.** Each NeRF method is implemented as a custom Pipeline. DataManagers process input images into bundles of rays (RayBundles) that get rendered by the Model to produce a set of NeRF outputs (RayOutputs). A dictionary of losses supervises the pipeline end-to-end.

Instant-NGP [145] significantly reduce computational overhead, enabling real-time training and rendering. However, Instant-NGP relies significantly on GPU accelerations with custom CUDA kernels, making development and quick prototyping a challenge. We present a framework that enables interactive visualizations while also being flexible and model-agnostic.

Nerfstudio is an extensible and versatile framework for neural radiance field development. Our design goals are the following:

1. Consolidating various NeRF techniques into reusable, modular components.
2. Enabling real-time visualization of NeRF scenes with a rich suite of controls.
3. Providing an end-to-end, easy-to-use workflow for creating NeRFs from user-captured data.

For modularity, we devise an organization among components across various NeRFs that allows abstracting away method-specific implementations. Our real-time visualizer is designed to work with any model during training or testing. Furthermore, the visualizer is hosted on the web, making it accessible without requiring a local GPU machine. The modular nature of our framework facilitates the integration of various data input formats, thereby simplifying the workflow for incorporating user-captured real-world scenes. We provide support for images and videos with various camera types, as well as other mobile capture applications (Polycam, Record3D, KIRI Engine) and outputs from popular photogrammetry software like RealityCapture and Metashape. In particular, integration with these applications enable users to by-pass structure-from-motion tools like COLMAP [192], which can be time-consuming. Furthermore, we provide support for multiple export formats, including video, depth maps, point clouds, and meshes.

The modularity of Nerfstudio enables developing Nerfacto, our method that combines components from recent papers to achieve a balance between speed and quality. We show

Figure 2.3: **Nerfstudio Dataset.** Our Nerfstudio Dataset contains 10 scenes: 4 phone captures with pinhole lenses and 6 Mirrorless camera captures with a fisheye lens. We focus our efforts on real-world data, and these scenes can help benchmark progress.

that this method is comparable to the other state-of-the-art methods such as MipNeRF-360 [11] while achieving an order of magnitude speedup. We also conduct an ablation study that demonstrates its flexibility on a new in-the-wild dataset consisting of 10 in-the-wild scenes. Our findings highlight the limitations of commonly used NeRF metrics and the importance of a real-time viewer for qualitative assessments. The potential of our framework as a consolidated codebase for NeRF research is reflected in the traction thus far with extensions such as SDFStudio [272]. Furthermore, Nerfstudio is an open-source project with active improvements from both academic and industry contributors.

2.2 Related Works

Frameworks and tools

Software frameworks have played a crucial role in consolidating and driving the advancement of various fields. In deep learning, Caffe [90], TensorFlow [1], and PyTorch [158] provide readily usable machine learning functionalities. Similarly, frameworks such as PyTorch3D [170] and Kornia [178] provide reusable components for 3D computer vision tasks. Other examples of frameworks include Mitsuba3 [87], Halide [168], Taichi [81], and Reyes [33] for graphics, Phototourism [203] and COLMAP [192, 193, 194] for photogrammetry and visualization, and AverageExplorer [284] for data collection. Despite the diversity of topics covered, each of these frameworks originated from the need to provide reusability and reproducibility to a rapidly expanding field. In light of the fast-paced growth of NeRFs in both academia and industry, Nerfstudio aims to streamline advancements in neural rendering by offering a flexible and comprehensive framework for development.

NeRF codebases

In recent years, several codebases for NeRFs have gained popularity among the research community, including the original NeRF codebase[138], nerf-pytorch [266, 146], Nerf_pl [167], Instant NGP [145], torch-ngp, Ngp_pl, and MultiNeRF [136]. Due to the lack of consolidation, there exists a significant number of NeRF repositories that focus on improving specific components of specific algorithms. For example, Mip-NeRF [12] aims to address the anti-aliasing problem of NeRF [138] and Mip-NeRF 360 [11] addresses the limitations of Mip-NeRF. Additionally, Plenoxels [52], TensoRF [28] and InstantNGP [145] propose different approaches to address the problem of computational efficiency. Furthermore, RawNeRF [137], Ref-NeRF [229], and NeRF-W [132] each address distinct challenges related to NeRF, resulting in parallel, non-interacting implementations. Nerfstudio aims to address the lack of consolidated development in the field of NeRFs by consolidating critical techniques introduced in the existing literature. This allows for more efficient and effective experimentation with combining components from multiple solutions into a single, comprehensive method, and facilitates the ability of the community to build upon existing prior approaches.

Neural rendering frameworks

Concurrent efforts such as NeRF-Factory [89], NerfAcc [111], MultiNeRF [136], and Kaolin-Wisp [212] all make significant efforts in advancing the usability of NeRFs. While NeRF-Factory consolidates multiple prior works into a single repository, it places less emphasis on reusable modules shared across these prior works and focuses more on benchmarking. NerfAcc prioritizes pythonic modularity, but focuses primarily on the lower-level components rather than the entire pipeline. Kaolin-Wisp and Multi-NeRF each consolidate multiple paper implementations into a single repository. None of these repositories are as comprehensive as Nerfstudio in delivering our three design goals: modularity, real-time visualization, and end-to-end usability for user-captured data. Furthermore, Nerfstudio is released under an Apache2 license, which allows for its use by both researchers and companies.

2.3 Framework Design

The goals of Nerfstudio are to provide (1) modularity, (2) real-time visualization for development, and (3) ease of use with real data. In designing the framework, we consider trade-offs against designs that optimize for faster rendering or higher quality results on synthetic scenes. For instance, we prefer an implementation that allows for a modularized pythonic non-CUDA method over one that supports a faster, non-modularized CUDA method. Additionally, our design choices lead to simpler interfacing with an extensive visualization ecosystem which supports real-time rendering during test and train with custom camera paths. Finally, we focus on delivering results for real-world data rather than synthetic scenes to address audiences outside research including those in industry and non-technical users.

With these three goals, the design of Nerfstudio promotes collaborations by providing a consolidated platform on which people can request for or contribute to new features. The long-term goal is for Nerfstudio to continue improving through community-driven contributions.

Modularity

We propose an organization of components that is both intuitive and abstract, enabling the implementation of existing and novel NeRFs by swapping reusable components. Fig. 2.1 shows a subset of the components types and implementations we currently have available in Nerfstudio.

Visualization for development

The Nerfstudio real-time viewer offers an interactive and intuitive way to visualize Neural Radiance Fields (NeRFs) during both training and testing phases. To ensure ease of use, the visualizer is simple to install, works seamlessly across both local and remote GPU compute environments, supports different models, and offers a user interface for creating and rendering custom camera paths, shown in Fig. 2.6 (a).

Our real-time visualization interface is particularly useful for qualitatively evaluating a model, allowing for more informed decisions during method development. While metrics such as PSNR can provide some insight, they do not offer a comprehensive understanding of performance–especially for views that are far away from the capture trajectory. Qualitative evaluation with an interactive viewer addresses these limitations and allows developers to gain a more holistic understanding of the model performance.

Easy workflow for user-captured data

While we offer support for synthetic datasets (Blender [138], D-NeRF [166]), in Nerfstudio we focus primarily on "real world data" — images or videos from a physical phone or camera. To this end, we present a new Nerfstudio Dataset (shown in Fig. 2.3) composed of real-world scenes casually captured with mobile phones and a mirrorless camera. Our motivation is to provide a framework compatible with a diverse array of applications which requires supporting real data. For instance, a few use cases for Nerfstudio outside of research include VFX, gaming, and non-technical film-makers who create 3D and video art. To support this wide range of expertise in NeRFs, we ensure our codebase is easily installable and deployable.

2.4 Core components

The proposed framework of Nerfstudio, illustrated in Fig. 2.2, is based on the conceptual grouping of NeRF methods into a series of basic building blocks. Nerfstudio takes a set of

posed images and optimizes for a 3D representation of the scene, which is defined by radiance (color), density (structure), and possibly other quantities (semantics, normals, features, etc.). We ingest these inputs into the framework which comprises of a DataManager and a Model, where the DataManager is responsible for (1) parsing image formats via a DataParser and (2) generating rays as RayBundles. These rays are then passed into a Model, which will query Fields and render quantities. Finally, the whole Pipeline is supervised end-to-end with a loss.

DataManagers and DataParsers

The first step of the Pipeline is the DataManager which is responsible for turning posed images into RayBundles, which are slices of 3D space that start at a camera origin. Within the DataManager, the DataParser first loads the input images and camera data. The DataParser is designed to be compatible with arbitrary data formats such as COLMAP. Previous research codebases primarily utilize COLMAP with helper scripts [145], however, COLMAP can be challenging to install and use for non-technical users. To make the framework more accessible to a wider range of users, including scientists, artists, photographers, hobbyists and journalists, we have implemented DataParsers for mobile apps (Record3D, Polycam, KIRI Engine) and 3D tools such as Metashape and Reality Capture. Once the images are properly loaded and formatted, the DataManager iterates through the data, generating RayBundles and ground truth supervision. It can also optimize camera poses during training.

RayBundles, RaySamples, and Frustums

NeRFs operate on regions of 3D space, which can be parametrized in many different ways. We have adopted a more generic representation of 3D space through the use of Frustum for both point-based and volume-based samples. The RayBundles, which are primitives that represent a slice through 3D space, are parameterized with an origin, direction, and other meta-information such as camera indices and time. By specifying the interval bin spacing, the RayBundles generate RaySamples, which represent sampled chunks of 3D space along each ray. These chunks, represented as Frustums, can be encoded either as point samples [138] or as Gaussians with mean and covariance [12], which have been shown to help with anti-aliasing. This abstraction allows for flexibility in representation, as the user can decide which representation to use with a simple function call. A visualization of this abstraction can be found in Fig. 2.4.

Models and Fields

The RayBundles are sent to Models as input, which samples them into RaySamples. The RaySamples are consumed by Fields to turn regions of space (i.e., Frustums) into quantities such as color or density. The Nerfstudio framework contains various implementations of models and fields. We've implemented various feature encoding schemes including fourier

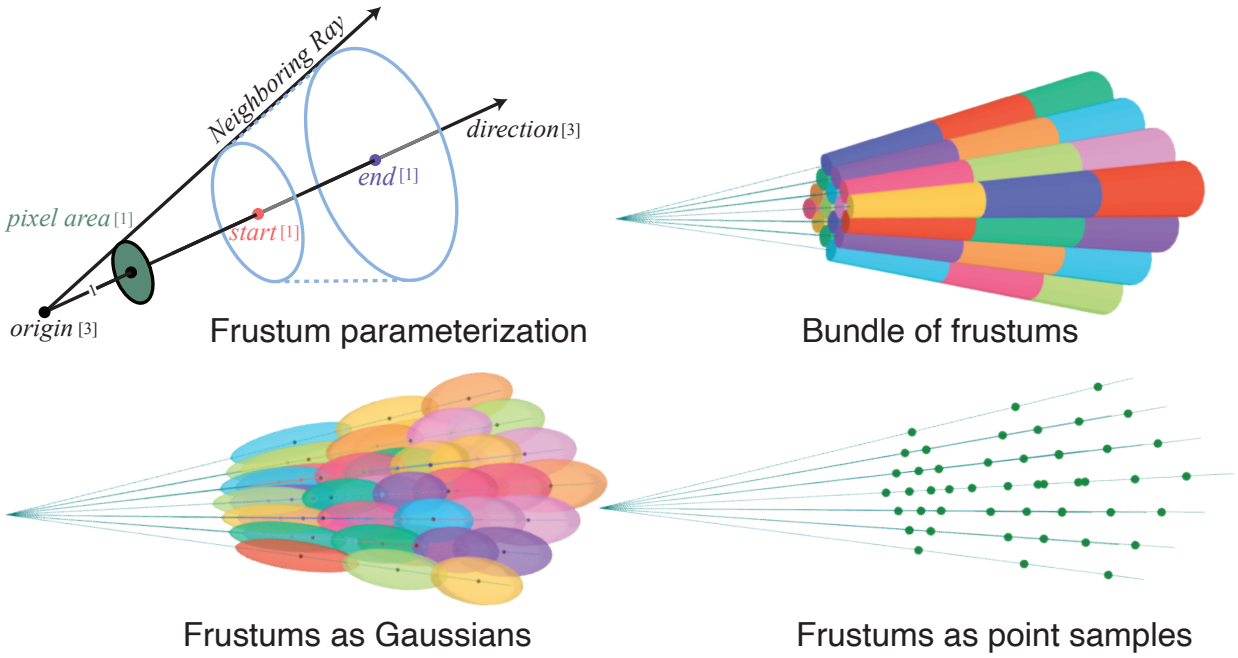

Figure 2.4: **Sample representations.** (Top) We define a frustum as a cone with a start and end. This region of space can be converted into Gaussians (bottom left) or point samples (bottom right) depending on the field input format.

features, hash encodings [145], spherical harmonics, and matrix decompositions [28]. Field components include fused MLPs, voxel grids, and surface normal MLPs [229], activation functions, spatial distortions [11], and temporal distortions [166].

Real-time web viewer

We draw inspiration from the real-time viewer presented in Instant NGP [145], which facilitates real-time rendering during training. However, the viewer in Instant NGP is designed to work on local compute, which can be cumbersome to setup in remote settings. To address this issue, we have developed a ReactJS-based web viewer packaged as a publicly hosted website at https://viewer.nerf.studio.

The viewer is designed to be accessible to a wide range of users, including those utilizing both local and remote GPUs. The process of utilizing remote compute is streamlined, requiring only the forwarding of a port locally via SSH. Once training begins, the web interface renders the NeRF in real-time as training progresses. Users can pan, zoom and rotate around the scene as the optimization runs or while evaluating a trained model. The design of the viewer is illustrated in Fig. 2.5.

Figure 2.5: **Web viewer design.** A machine with a GPU (left) starts a NeRF training session. When a user navigates to to the hosted web viewer (right), the viewer client will establish WebSocket and WebRTC connections with the training session.

Implementation

Real-time training visualization utilizes WebSockets and WebRTC to establish a connection between the NeRF training session and the web client. This approach eliminates the need to install local screens and other GUI software. Upon opening the web viewer, a WebSocket connection is established with the training session, which subsequently populates the scene with training images as illustrated in Fig. 2.5 (right). The web viewer continuously streams the viewport camera pose to the training session during the training process. The training session utilizes this camera pose to render images and transmits them via a WebRTC video stream. Additionally, the viewer camera controls and UI are implemented using ThreeJS, allowing us to overlay 3D assets such as images, splines, and cropping boxes in front of the NeRF renderings. For instance, the viewer displays training images at their capture locations, letting users intuitively compare performance at seen and novel viewpoints.

Viewer features

Our viewer is compatible with different models of varying rendering speeds. We accomplish this by balancing the computation of training and viewer rendering on a single GPU. Similar to Instant-NGP [145], we adjust the rendering resolution based on the speed of the camera movement. When the camera moves quickly, the rendering resolution will be smaller to maintain a frame rate and prevent lag in the user experience. We can also reduce the time spent on training and allocate more resources for rendering in the viewer. Some of the features of our viewer include:

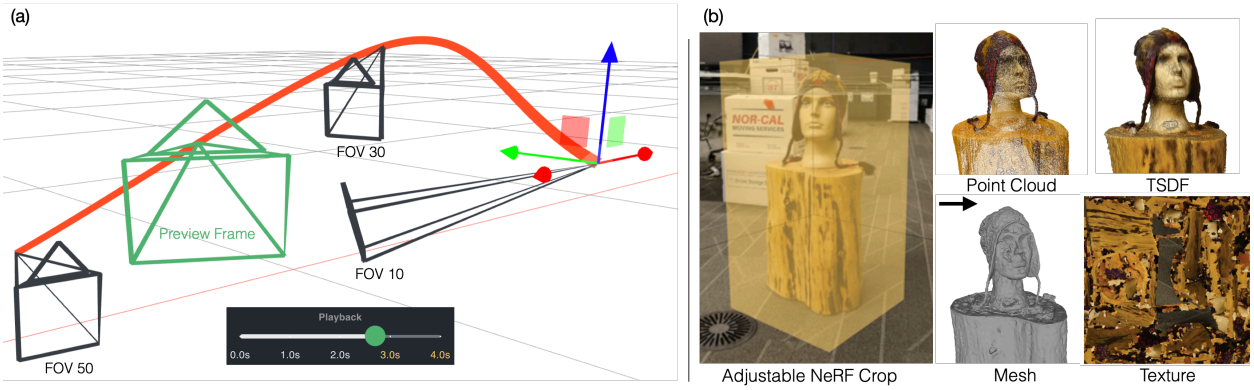

Figure 2.6: **Exporting videos and geometry.** We make exporting videos (a) and geometry (b) easy with real-data captures. The left side shows the interactive camera trajectory editor, which allows animatable poses, FOVs, and speed, to eventually render videos of NeRF's outputs. On the right we show the cropping interface in the viewer and resulting export formats including point clouds, TSDFs, and textured meshes.

- Switching between various model outputs (e.g., rgb, depth, normals, semantics).
- Creating custom camera paths composed of keyframes with position and focal length interpolation (Fig. 2.6).
- Visualizing the captured training images in 3D.
- Crop and export options for point clouds and meshes.
- Mouse and keyboard controls to easily navigate in the scene.

The viewer played an instrumental role in providing qualitative assessments that informed design choices in our default method Nerfacto. Other codebases have integrated our viewer into their own codebases, including ArcNerf [274] and SDFStudio [272].

Geometry export

Many creators and artists have workflows that require exporting to point clouds or meshes for further processing and incorporation in downstream tools such as game engines. Hence, our framework accommodates various export methods and facilitates the easy addition of new export methods. Fig. 2.6b illustrates our export interface, as well as some of the supported formats, including point clouds, a truncated signed distance function (TSDF) to mesh, and Poisson surface reconstruction [97]. We apply texture to the mesh by densely sampling the texture image, utilizing barycentric interpolation to determine corresponding 3D point locations, and rendering short rays near the surface along the normals to obtain RGB values.

2.5 Nerfacto Method

We leverage our modular design to integrate ideas from multiple research papers into our default and recommended method, Nerfacto. This method is heavily influenced by the structure of MipNeRF-360 [11], but certain parts of the original design are replaced to improve performance. We reference papers such as NeRF-- [244], Instant-NGP [145], NeRF-W [132], and Ref-NeRF [229] in Nerfacto. Fig. 2.7 illustrates how these papers are used.

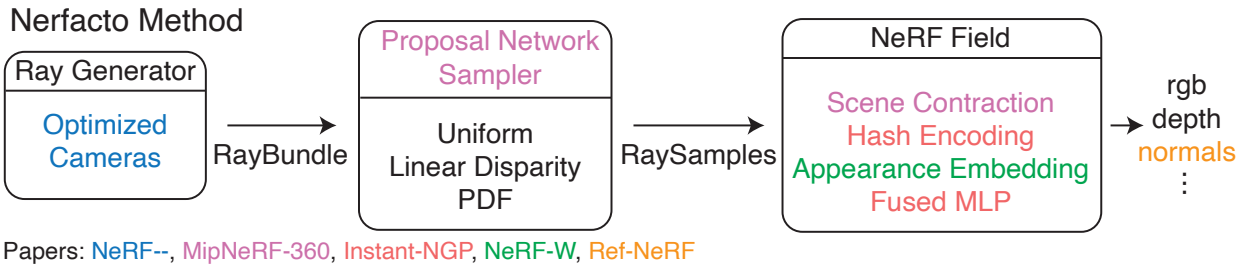

Figure 2.7: **Nerfacto method.** Diagram of the Nerfacto method. It combines features from many papers (bottom left). The method will evolve over time as new papers and features are added to the Nerfstudio codebase.

Ray generation and sampling

The Nerfacto method first optimizes camera views using an optimized SE(3) transformation [244, 115, 213]. These camera views are then used to generate RayBundles. To improve the efficiency and effectiveness of the sampling process, we employ a piece-wise sampler. This sampler samples uniformly up to a fixed distance from the camera, followed by samples that are distributed such that the step size increases with each sample. This allows efficient sampling of distant objects while still maintaining a dense set of samples for nearby objects. These samples are then fed into a proposal network sampler, proposed in the MipNeRF-360 method [11]. The proposal sampler consolidates the sample locations into regions of the scene that contribute most to the final render, typically the first surface intersection. This importance sampling greatly improves reconstruction quality. Furthermore, we use a small fused MLP with a hash encoding [145] for the scene's density function as it has been found to have sufficient accuracy and is computationally efficient. To further reduce the number of samples along rays, the proposal network sampler can contain multiple density fields. These density fields iteratively reduce the number of samples. Empirically, using two density fields works well. In our base Nerfacto configuration, we generate 256 samples from the piece-wise sampler, which gets resampled into 96 samples in the first iteration of the proposal sampler followed by 48 samples in the second.

Scene contraction and NeRF field

Many real-world scenes are unbounded, meaning they could extend indefinitely. This poses a challenge for processing as input samples could have position values that vary across many scales of magnitude. To overcome this issue, we utilize *scene contraction*, which compresses the infinite space into a fixed-size bounding box. Our method of contraction is based on the one proposed in MipNeRF-360 [11], but we use L^∞ norm contraction instead of L^2 norm, which contracts to a cube rather than a sphere. The cube better aligns with voxel-based hash encodings. Fig. 2.8 illustrates how L^∞ contraction maps samples into the range with minimum values of -2,-2,-2 and maximum values of 2,2,2. These samples can then be used with the hash encoding introduced by Instant-NGP and is available via the tiny-cuda-nn [144] Python bindings.

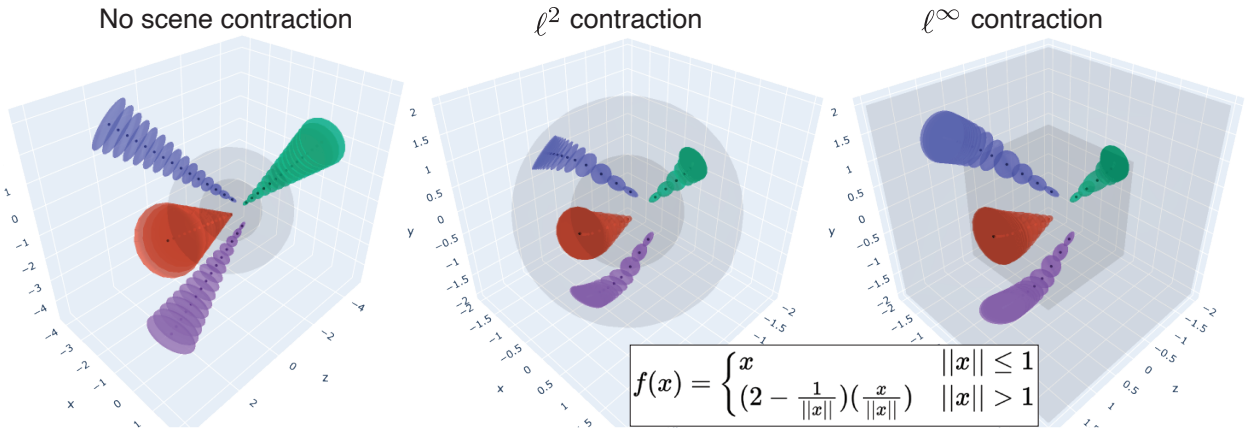

Figure 2.8: **Scene contraction.** Here we show cameras contained in an inner sphere with Gaussian samples along rays. Scene contraction warps the unbounded samples into bounded space before querying a NeRF field. We use L^∞ contraction rather than MipNeRF-360's L^2 contraction to better accommodate the geometry/capacity of the hash grid.

Nerfacto's field incorporates per-image appearance embeddings to account for differences in exposure among training cameras [132]. Additionally, we use techniques from Ref-NeRF [229] to compute and predict normals. Nerfacto is implemented using PyTorch, which allows for easy customization and eliminates the need for complex and custom CUDA code. We will incorporate new papers into Nerfacto as the field progresses.

2.6 Nerfstudio Dataset

Our "Nerfstudio Dataset" includes 10 in-the-wild captures obtained using either a mobile phone or a mirror-less camera with a fisheye lens. We processed the data using either COLMAP or the Polycam app to obtain camera poses and intrinsic parameters. Our goal

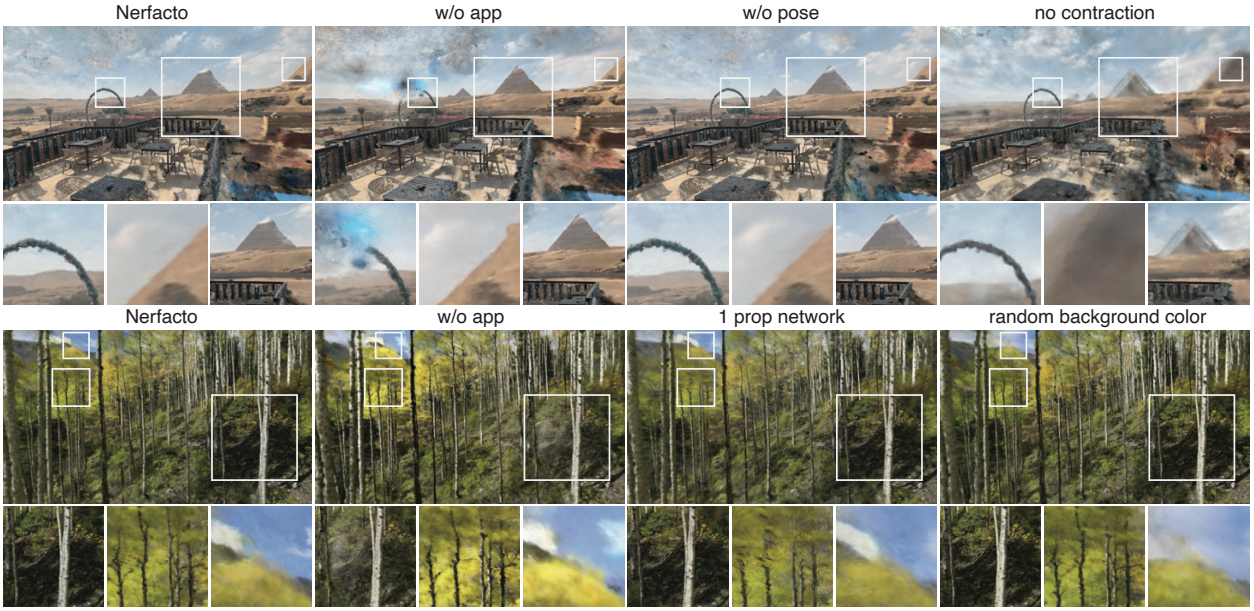

Figure 2.9: **Nerfstudio ablation qualitative examples.** Here we show renderings from different Nerfacto ablation variants. (Top) is the "Egypt" capture and (bottom) is the "aspen" capture from the Nerfstudio Dataset. These novel views are far from the training images to get a sense of how well these methods perform qualitatively. We zoom in on crops to highlight differences in the rendered images.

is to provide researchers with more 360 real-world captures that are not limited to forward-facing scenes [135]. Our dataset is similar to MipNeRF-360 [11] but does not focus on a central object and includes captures with varying degrees quality. We have used this dataset to select the default settings for our proposed NeRF-based method, Nerfacto, and we encourage other researchers to similarly employ real-world data in the development and evaluation of NeRF methods.

2.7 Experiments

We benchmark Nerfacto against a state-of-the-art method MipNeRF-360 and emphasize the modularity of our repository by conducting ablation studies. Furthermore, we highlight the limitations of commonly used evaluation metrics such as PSNR, SSIM, and LPIPS when applied to subsampled evaluation images.

Mip-NeRF 360 dataset comparison

Here we compare Nerfacto with numbers reported in the MipNeRF-360 [11] paper. We evaluate on their 7 publicly available scenes. We train our method for up to 30K iterations

Table 2.1: **Average metrics on the MipNeRF360 dataset.** Our methods are evaluated without pose optimization or per-image appearance embeddings. MipNeRF-360 takes several hours to train. Our metrics reported as { after 30K iterations (∼30min) / after 5k iterations (∼5min) }.

Method	**PSNR** ↑	**SSIM** ↑	**LPIPS** ↓
MipNeRF-360	29.23	0.844	0.207
Nerfacto (ours)	26.75 / 25.38	0.748 / 0.688	0.307 / 0.390

(30 minutes) on an NVIDIA RTX A5000, but we also report results at 5K iterations (5 minutes).

Evaluation protocol. The evaluation protocol followed is similar to that of MipNeRF360, but we process their data using our COLMAP pipeline to recover poses. The original images were downsampled by a factor of 4x. We used 7/8 of the images for training and the remaining 1/8 images were evenly spaced and used for evaluation. Note that this protocol does not include camera pose optimization as it is not an option implemented in MipNeRF360.

Findings. Table 2.1 presents the averages of the results across the 7 captures in the MipNeRF-360 dataset. The complete table can be found in the appendix. In as little as 5K iterations (∼5 minutes), our Nerfacto method achieves reasonable quality in contrast to MipNeRF-360 which takes several hours on a TPU with 32 cores. Training for up to 30K (∼30 minutes) iterations further improves quality. While Nerfacto falls short of metric results obtained by MipNeRF-360, we prioritize efficiency and general usability over optimizing quantitative metrics on this particular benchmark.

It is worth emphasizing that our Nerfacto method is optimized for qualitative novel-view quality by using the web viewer, rather than solely relying on common metrics. For further illustration, we refer the reader to the appendix where we provide rendered videos from our Nerfacto method.

Nerfacto component ablations

Given the modularity of our codebase, we can easily conduct ablation studies on our method Nerfacto, a unified approach that combines important components from various papers to achieve a fast, high-quality method. We experiment with disabling the pose optimization, appearance embeddings, scene contraction, and variations of the proposal networks, and more. The modularity of our codebase allows for easy implementation of these modifications through the use of different flags with the command line interface.

Evaluation protocol. In our ablation study, we utilize the Nerfstudio Dataset for evaluation. Due to the complexity of the appearance embeddings and pose optimization modules, we adopt a test-time optimization procedure for the evaluation. Specifically, we

Table 2.2: **Average metrics for ablations on the Nerfstudio Dataset.** We remove and change various components of the Nerfacto method and report { PSNR, SSIM, LPIPS } on the Nerfstudio Dataset. Further details on the experiments can be found in the Appendix.

Nerfacto method	**PSNR** ↑	**SSIM** ↑	**LPIPS** ↓
Nerfacto (default)	20.99	0.663	0.389
w/o pose	20.93	0.659	0.393
w/o app	22.65	0.672	0.406
w/o pose & app	22.53	0.671	0.411
1 prop network	21.07	0.669	0.396
l2 contraction	20.98	0.664	0.388
shared prop network	20.95	0.661	0.391
random backg. color	21.00	0.663	0.392
no contraction	18.59	0.534	0.506
synthetic on real	20.09	0.542	0.509

employ Adam optimizers to optimize the evaluation camera poses. Once the camera poses are fixed, we randomly select either the left or right side of the evaluation image and optimize the appearance code as done in Martin et al. [132]. Finally, with the optimized camera pose and appearance embedding, we compute PSNR, SSIM, and LPIPS. For these experiments, we hold out 1 in every 10 frames of our data as the evaluation set to evaluate on a representative distribution of our data.

Findings. Table 2.2 presents the average results of our ablation studies. The complete table for all 10 scenes can be found in the appendix. This study highlights the challenge in extracting meaningful insights from quantitative metrics alone (Table 2.2), due to the fact that held-out evaluation images are close to the training images. For instance, disabling the appearance embeddings ("w/o app") leads to an improvement in PSNR and SSIM. However, Fig. 2.9 illustrates that the "w/o app" method results in the production of blurry "floater" artifacts. These artifacts correspond with the training camera locations because the model overfits to small discrepancies in lighting conditions in the training data by placing these artifacts directly in front of the training cameras. (bottom row, bottom left crop). Furthermore, ablations such as "1 prop network" result in subtle changes in the metrics but are more evident in visualizations of the novel views. The use of "1 prop network" as opposed to "Nerfacto (default)" with 2 prop networks leads to aliasing artifacts as can be seen around the small tree branches (bottom row, middle crop). While these artifacts are visible to the eye especially in the interactive viewer, such temporal discontinuity caused by aliasing is not captured by the quantitative metrics. Furthermore, scene contraction is necessary to correctly recover far objects (top row, right crop).

Overall, the real-time viewer proves to be useful for viewing out-of-distribution renders. The crops in Fig. 2.9 aid in illustrating where certain methods excel over others, regardless

of the metrics on the evaluation images. Developing more appropriate evaluation metrics is an important avenue for future research.

2.8 Open-source Contributions

One of the key strengths of our proposed framework is its versatility and ease of use, as demonstrated by our open-source contributions. Our GitHub repository has grown to include over 60 contributors and over 3K stars, reflecting a strong and active community. Additionally, two new libraries, SDFStudio [272] and ArcNerf [274], have been built on top of our framework. Since the release of Nerfstudio in October 2022, our contributors have enhanced and expanded Nerfstudio by addressing various GitHub issues and feature requests including improved camera paths, colab support, additional camera models, reconstruction of dynamic objects. In the future, we plan on supporting 3D generative pipelines, NeRF compositing, and more.

2.9 Conclusion and Future Work

We draw upon existing techniques and propose a framework that supports a more modularized approach to NeRF development, allows for real-time visualization, and is readily usable with real-world data. We emphasize the importance of utilizing the interactive real-time viewer during training to compensate for imperfect quantitative metrics in model design decisions. We hope the consolidation brought about by this new framework will facilitate the development of NeRF-based methods, thereby accelerating advances in the neural rendering community. Future research directions include the development of more appropriate evaluation metrics and integration of the framework with other fields such as computer vision, computer graphics, and machine learning.

Acknowledgements

The authors of this paper are Matthew Tancik*, Ethan Weber*, Evonne Ng*, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. * denotes equal contribution. We also want to thank the many open-source contributors who have helped create Nerfstudio, including Cyrus Vachha and Rohan Mathur from UC Berkeley and the following Github users: machenmusik, kevinddchen (Kevin Chen), dozeri83, nikmo33 (Nikhil Mohan), lsongx (Liangchen Song), zmurez (Zak Murez), JulianKnodt (Julian Knodt), katrinbschmid (Katrin Schmid), mpmisko (Michal Pandy), RandomPrototypes, ManuConcepBrito , Zunhammer (Nicolas Zunhammer), nlml (Liam Schoneveld), jkulhanek (Jonáš Kulhánek), mackopes (Martin Piala), cnsumner, devernay (Frédéric Devernay), matsuren (Ren Komatsu),

Mason-McGough (Mason McGough), hturki, decrispell (Daniel Crispell), dkorolov (Dmytro Korolov), gilureta (Francisca T. Gil Ureta).

Chapter 3

Nerfbusters: Removing reconstruction artifacts

Casually captured Neural Radiance Fields (NeRFs) suffer from artifacts such as floaters or flawed geometry when rendered outside the camera trajectory. Existing evaluation protocols often do not capture these effects, since they usually only assess image quality at every 8th frame of the training capture. To push forward progress in novel-view synthesis, we propose a new dataset and evaluation procedure, where two camera trajectories are recorded of the scene: one used for training, and the other for evaluation. In this more challenging in-the-wild setting, we find that existing hand-crafted regularizers do not remove floaters nor improve scene geometry. Thus, we propose a 3D diffusion-based method that leverages local 3D priors and a novel density-based score distillation sampling loss to discourage artifacts during NeRF optimization. We show that this data-driven prior removes floaters and improves scene geometry for casual captures.

3.1 Introduction

Casual captures of Neural Radiance Fields (NeRFs) [138] are usually of lower quality than most captures shown in NeRF papers. When a typical user (e.g., a hobbyist) captures a NeRFs, the ultimate objective is often to render a fly-through path from a considerably different set of viewpoints than the originally captured images. This large viewpoint change between training and rendering views usually reveals *floater* artifacts and bad geometry, as shown in fig. 3.1a. One way to resolve these artifacts is to teach or otherwise encourage users to more extensively capture a scene, as is commonly done in apps such as Polycam* and Luma*, which will direct users to make three circles at three different elevations looking inward at the object of interest. However, these capture processes can be tedious, and

*https://poly.cam/

*https://lumalabs.ai/

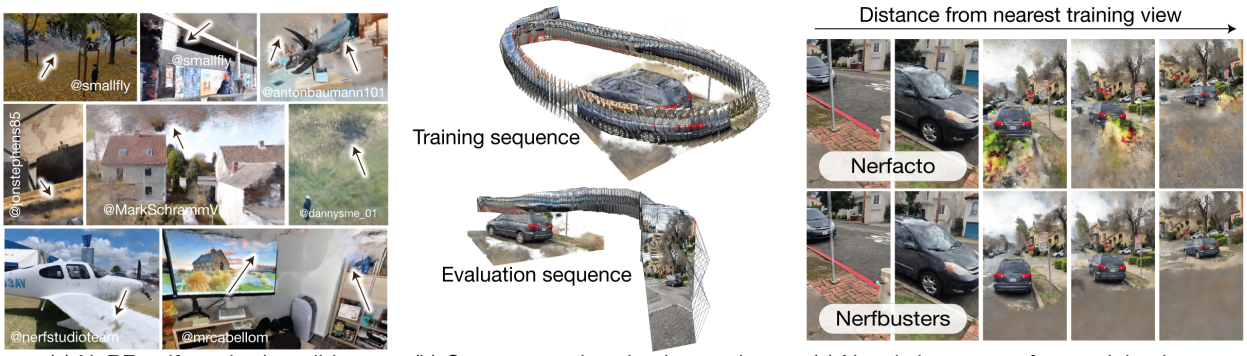

Figure 3.1: **Nerfbusters.** Rendering NeRFs at novel views far away from training views can result in artifacts, such as floaters or bad geometry. These artifacts are prevalent in in-the-wild captures (a) but are rarely seen in NeRF benchmarks, because evaluation views are often selected from the same camera path as the training views. We propose a new dataset of in-the-wild captures and a more realistic evaluation procedure (b), where each scene is captured by two paths: one for training and one for evaluation. We also propose Nerfbusters, a 3D diffusion-based method that improves scene geometry and reduces floaters (c), significantly improving upon existing regularizers in this more realistic evaluation setting.

furthermore, users may not always follow complex capture instructions well enough to get an artifact-free capture.

Another way to clean NeRF artifacts is to develop algorithms that allow for better out-of-distribution NeRF renderings. Prior work has explored ways of mitigating artifacts by using camera pose optimization [244, 115] to handle noisy camera poses, per-image appearance embeddings to handle changes in exposure [132], or robust loss functions to handle transient occluders [185]. However, while these techniques and others show improvements on standard benchmarks, most benchmarks focus on evaluating image quality at held-out frames from the training sequence, which is not usually representative of visual quality at novel viewpoints. fig. 3.1 shows how the Nerfacto method starts to degrade as the novel-view becomes more extreme.

In this paper, we propose both (1) a novel method for cleaning up casually captured NeRFs and (2) a new evaluation procedure for measuring the quality of a NeRF that better reflects rendered image quality at novel viewpoints. Our proposed evaluation protocol is to capture two videos: one for training a NeRF, and a second for novel-view evaluation (fig. 3.1b). Using the images from the second capture as ground-truth (as well as depth and normals extracted from a reconstruction on *all* frames), we can compute a set of metrics on visible regions where we expect the scene to have been reasonably captured in the training sequence. Following this evaluation protocol, we capture a new dataset with 12 scenes, each with two camera sequences for training and evaluation.

We also propose *Nerfbusters*, a method aimed at improving geometry for everyday NeRF captures by improving surface coherence, cleaning up floaters, and removing cloudy artifacts. Our method learns a local 3D geometric prior with a diffusion network trained on synthetic 3D data and uses this prior to encourage plausible geometry during NeRF optimization. Compared to global 3D priors, local geometry is simpler, category-agnostic, and more repeatable, making it suitable for arbitrary scenes and smaller-scale networks (a 28 Mb U-Net effectively models the distribution of all plausible surface patches). Given this data-driven, local 3D prior, we use a novel unconditional Density Score Distillation Sampling (DSDS) loss to regularize the NeRF. We find that this technique removes floaters and makes the scene geometry crisper. To the best of our knowledge, we are the first to demonstrate that a learned local 3D prior can improve NeRFs. Empirically, we demonstrate that Nerfbusters achieves state-of-the-art performance for casual captures compared to other geometry regularizers.

We implement our evaluation procedure and Nerfbusters method in the open-source Nerfstudio repository [214]. The code and data can be found at `https://ethanweber.me/nerfbusters`.

3.2 Related Work

Evaluating NeRFs in-the-wild. Early works in neural rendering [135], including NeRF [138], established an evaluation protocol for novel view synthesis, where every 8th frame from a camera trajectory is used for evaluation. Most follow-up works have adapted this protocol and demonstrated impressive results on forward-facing scenes in LLFF [135], synthetic scenes [138], or 360 scenes [11, 173]. In these datasets, the training and evaluation views share camera trajectories, thus the methods are evaluated only for small viewpoint changes, as illustrated in fig. 3.2. In contrast, we propose to record two camera trajectories, one for training and one for evaluation. Our supplementary material compares existing datasets (synthetic scenes [138], LLFF [135], MipNeRF 360 [11], and Phototourism [93]) with the proposed Nerfbusters dataset. We visualize the training and evaluation poses for each scene and quantify the difficulty of each dataset by computing the average rotation and translation difference between evaluation images and their closest training images. We find that viewpoint changes are very limited, and the proposed Nerfbusters dataset is much more challenging. Recently, Gao et al. [56] revisited the evaluation process for dynamic NeRFs, also highlighting shortcomings in dynamic NeRF evaluation. NeRFs for extreme viewpoint changes and few-shot reconstruction have been explored on ShapeNet [27], DTU [88], and CO3D [173], where a few or just a single view is available during training. These works focus on the generalization and hallucination of unseen regions, and either assume a category-specific prior [283, 268] or focus on simple scenes [268]. In contrast, our casual captures setting assumes that a $10-20$ second video is available at training time, better reflecting how people capture NeRFs. We then evaluate fly-throughs with extreme novel views on an entirely different video sequence, as illustrated in fig. 3.2.

Diffusion models for 3D. Recently, several works have proposed the use of diffusion

Figure 3.2: **Evaluation protocols.** Current evaluation of NeRFs (e.g., MipNeRF 360) measures render quality at every 8th frame of the captured (training) trajectory, thus only testing the model's ability to render views with small viewpoint changes. In contrast, we propose a new evaluation protocol, where two sequences are captured of the same scene: one for training and one for evaluation. Please see the supplementary material for plots showing the training and evaluation sequences for various NeRF datasets, including our proposed Nerfbusters Dataset.

models for 3D generation or manipulation [165, 234, 142, 257]. These approaches can be divided into (1) methods that distill priors from existing 2D text-to-image diffusion models into a consistent 3D representation [165, 234], and (2) methods that train a diffusion model to explicitly model 3D objects or scenes [142]. These directions are complementary, where the former benefits from the sheer size of image datasets, and the latter from directly modeling 3D consistency. DreamFusion [165] proposes Score Distillation Sampling (SDS), where the text-guided priors from a large text-to-image diffusion model can be used to estimate the gradient direction in optimization of a 3D scene. We take inspiration from the SDS optimization procedure but instead adapt it to supervise NeRF densities in an unconditional manner, directly on 3D density values. As the underlying model, we train a 3D diffusion model on local 3D cubes extracted from ShapeNet [27] objects. We find the distribution of geometry (surfaces) within local cubes is significantly simpler than 2D natural images or global 3D objects, reducing the need for conditioning and high guidance weights. To the best of our knowledge, we are the first to suggest a learned 3D prior for category-agnostic, unbounded NeRFs.

Data-driven local 3D priors. Approaches for learning 3D geometry can be divided into local and global approaches, where global approaches reason about the entire scene, and local approaches decompose the scene into local surface patches. We learn a prior over local

geometric structures, as local structures are simple, category-agnostic, and more repeatable than global structures [24, 141]. DeepLS [24] proposes to decompose a DeepSDF [155] into local shapes, and finds that this simplifies the prior distribution that the network learns. Similarly, AutoSDF [141] learns a local 3D prior and tries to learn the distribution over a 3D scene with an autoregressive transformer. We are inspired by their approach, but use a diffusion model rather than a VQ-VAE [171, 225], and show that the learned prior can be used to regularize NeRF geometry.

Regularizers in NeRFs. Our work can be seen as a regularizer for NeRFs. Most existing regularizers are hand-crafted priors that encourage smoothness and discourage non-empty space. Plenoxels [52] proposed a Total-Variation (TV) regularizer in 3D that penalizes the large changes between neighboring voxels. TV has also been applied in 2D rendered images in RegNeRF [152] and on the basis of factorized plenoptic fields [28, 51]. Plenoctrees [269] proposed a sparsity loss that penalizes densities from randomly queried 3D locations in space. This sparsity loss removes densities unnecessary in explaining the training views. To avoid penalizing all densities equally, MIP-NeRF 360 [11] proposes a distortion loss on accumulated weights that encourages surfaces to be sharp. Concurrent and most similar to our work, DiffusionRF [257] proposes a data-driven RGB-D diffusion prior. The method trains a diffusion model on synthetic RGB-D examples and uses the learned prior to regularize a NeRF. In contrast to the proposed local 3D diffusion prior, operating in 2.5D comes with several disadvantages, namely 1) occlusions are not modeled, 2) the joint distribution of RGB-D images is more complex than that of 3D occupancy (and as a result requires more data for generalization), 3) and unlike a 3D diffusion model, it is not by definition 3D-consistent, i.e., the view consistency has to come from the NeRF rather than the regularizer.

3.3 Evaluation Procedure

We propose an evaluation protocol that better reflects the rendering quality of a captured NeRF at novel viewpoints. Under this protocol, a scene should be captured by two videos, one for training and one for evaluation. Training videos should be around $10 - 20$ seconds, which is representative of what a user might do when prompted to scan an object or scene (as anything longer than this may reduce the appeal and practicality of NeRF captures). The second video should ideally capture a set of novel views that a user may wish to render. For everyday user captures, the second video is not needed, as it is only used as ground truth in evaluation. We record 12 scenes (two videos each) following this protocol to construct our Nerfbusters Dataset. All videos were taken with a hand-held phone to best simulate a typical casual capture setup.

Evaluating on casual captures. The steps to create our evaluation data can be boiled down to the following straightforward steps:

1. Record a video to capture the scene (training split).
2. Record a second video from a different set of viewpoints (evaluation split).

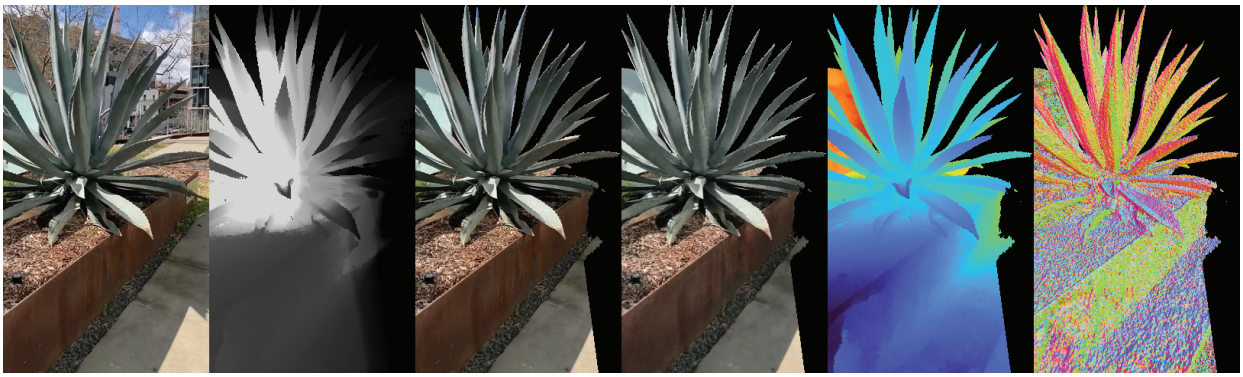

Figure 3.3: **Evaluation capture.** Here we show the data used in our evaluation protocol. The evaluation trajectory is a separate capture that is held out during optimization of the NeRF. Individual components shown here are further described in section 3.3.

3. Extract images from both videos and compute camera poses for all images.
4. Train a "pseudo ground truth" model on a combination of both splits and save depth, normal, and visibility maps for evaluation viewpoints.
5. Optimize a NeRF on the training split and evaluate at novel viewpoints (using the captured images and pseudo ground truth maps) from the evaluation split.

In fig. 3.3, we show an evaluation image and its visibility, depth, and normal maps. These pseudo ground truth properties are high-quality since they are extracted from a NeRF model that was trained on a combination of the training and evaluation views. The visibility map is computed by taking the depth map, back-projecting each pixel into a 3D point, and then counting how many training views observe that 3D point. Our final Nerfbusters dataset with associated visibility masks and processing code can be found at `https://ethanweber.me/nerfbusters`.

Masking valid regions. Rendering extreme novel views exposes parts of the scene that were not captured in the training views. As most existing NeRFs are not designed to hallucinate completely unseen views, we only evaluate regions in the evaluation images that were co-observed in the training views. We accomplish this by using visibility masks, which are defined to be regions that are either (1) not seen by any training views or (2) are predicted to be too far away (i.e., predicted depth $>$ distance threshold). We set this threshold to two times the largest distance between any two camera origins in both the training and evaluation splits. In the Nerfstudio codebase, this corresponds to a value of 2 because camera poses are scaled to fit within a box with bounds (-1,-1,-1) and (1,1,1).

Coverage. We additionally report "coverage", which is the percent of evaluated pixels (i.e., after masking by both visibility [56] and depth) among all pixels in the evaluation viewpoints, a metric commonly reported in depth completion [280, 247, 245]. For example, removing all densities and predicting infinite depth would result in zero coverage.

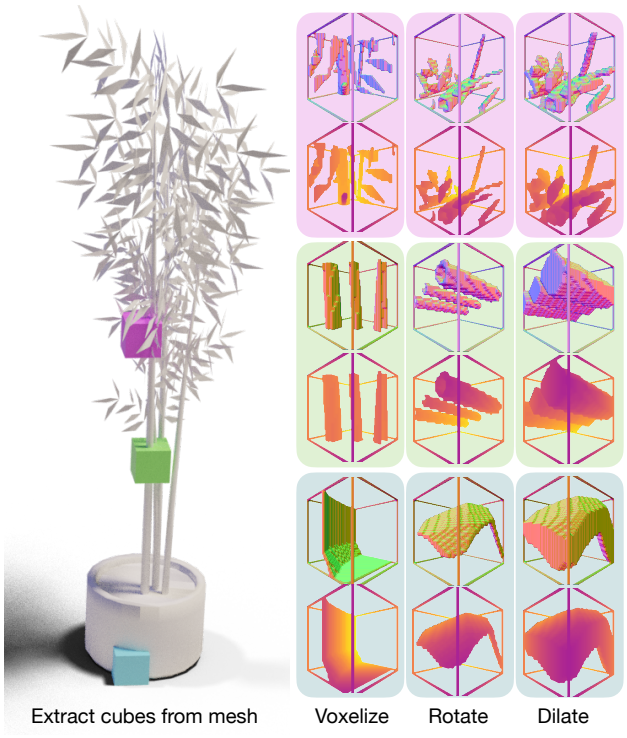

Figure 3.4: **Training data for Nerfbusters diffusion model.** Given a mesh, we extract local cubes scaled $1-10\%$ of the mesh size. We voxelize these cubes with resolution 32^3, and augment them with random rotations and random dilation. We illustrate each step with renderings of depth and normals from three cubes. The synthetic scenes from Shapenet offer a high variety in local cubes, containing both flat surfaces, round shapes, and fine structures.

Image quality and geometry metrics. We use masked versions of PSNR, SSIM, and LPIPS for image quality. We also report on depth (MSE and mean abs. disparity difference) and normals (mean and median degrees, and the percent of valid pixels with normals < 30 degrees). We report averages for all images in the Nerfbusters Dataset in Sec. 3.5.

3.4 Nerfbusters

We propose a novel diffusion-based approach for regularizing scene geometry to improve NeRF rendering quality at novel viewpoints. Our method consists of two steps. First, we train a diffusion model to learn the distribution of 3D surface patches. This model is trained on synthetic data to unconditionally generate local 3D cubes. Second, we demonstrate an approach to apply this local prior in NeRF reconstruction of real 3D scenes. We do this by querying densities in local 3D patches in the scene during training and using a novel Density Score Distillation Score (DSDS) loss to regularize sampled density. This prior improves

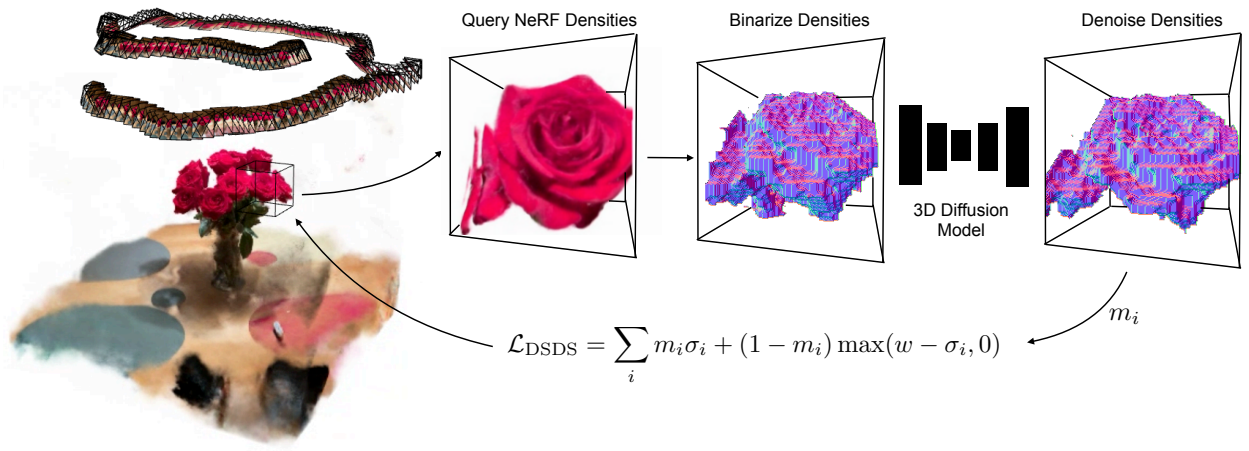

Figure 3.5: **Method overview.** We learn a local 3D prior with a diffusion model that regularizes the 3D geometry of NeRFs. We use importance sampling to query a 32^3 cube of NeRF densities. We binarize these densities and perform one single denoising step using a pre-trained 3D diffusion model. With these denoised densities, we compute a Density Score Distillation Sampling (DSDS) loss that penalizes NeRF densities where the diffusion model predicts empty voxels and pushes the NeRF densities above the target w where the diffusion model predicts occupied voxels $m = \mathbb{1}\{x_0 < 0\}$.

reconstructions in regions with sparse supervision signals and removes floaters. fig. 3.5 provides an overview of our pipeline.

Data-driven 3D prior

Following the recent process in generative models [204, 205, 151, 181, 165], we formulate our local 3D prior as a denoising diffusion probabilistic model (DDPM) [74], which iteratively denoises a voxelized $32 \times 32 \times 32$ cube of occupancy x. Our diffusion model ϵ_θ is trained with the loss:

$$\mathcal{L}_{\text{Diff}} = \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\|_2^2 \tag{3.1}$$

where $t \sim \mathcal{U}(0, 1000)$, $\epsilon \sim \mathcal{N}(0, I)$ and $\bar{\alpha}$ follows a linear schedule that determines the amount of noise added at timestep t. We implement our diffusion model as a small 3D U-Net [182] with only 7.2M parameters (28MB).We train the model on synthetic 3D cubes extracted from ShapeNet [27] scenes.

Curate synthetic 3D cubes

We train our diffusion model on local cubes sampled from ShapeNet [27], illustrated in fig. 3.4. To collect 3D cubes for training, we select a random ShapeNet mesh and extract

N local cube-bounded patches along the object surface with cube sizes varying between 1-10% of the object's bounding volume. We voxelize these local samples at a resolution of 32^3. We then augment the cubes with random amounts of rotation and dilation. This data processing pipeline is fast and performed online during training to increase the diversity of 3D cubes. We find that adjusting the thickness of the surface with dilation (rather than marking interior pixels as occupied) is faster and better defined for non-watertight meshes. fig. 3.4 illustrates the large diversity in the local cubes—some contain flat surfaces (bottom of the vase), round shapes (stem), and fine structures (leaves).

Applying 3D prior in-the-wild

We represent a 3D scene with a Neural Radiance Field (NeRF), [138] which takes a 3D point as input and outputs color and density, and is trained with differentiable volume rendering [138, 133]. We build on the Nerfacto model from Nerfstudio [214] that combines recent progress in NeRFs including hash grid encoding [145], proposal sampling [11], per-image-appearance optimization [132], and scene contraction [11]. Although Nerfacto has been optimized for in-the-wild image captures, it still reveals floaters when rendered from novel views. To address these issues, we propose a novel regularization strategy that includes (1) a mechanism for sampling cubes of occupancy from non-empty parts of the NeRF and (2) a novel Density Score Distillation Score (DSDS) loss that encourages sampled occupancy to agree with the learned geometry prior of the diffusion model.

Cube importance sampling. Since the NeRF represents a density field, we can query cubes of occupancy in 3D space at any size, location, and resolution. Sampling these cubes uniformly from the NeRF volume is inefficient, particularly if much of the scene is empty. To enable more efficient sampling, we store a low-resolution grid of either *accumulation weights* or *densities*, which can be used to inform the probability with which to sample different positions in the scene. This grid is jointly updated with exponential moving average (EMA) decay over the course of NeRF training, such that regions with deleted floaters are not repeatedly sampled during later stages of training. The choice between the use of accumulation weights and densities has associated trade-offs: using *accumulation weights* yields cubes sampled mostly on frequently seen surfaces, whereas using *densities* enables sampling of occluded regions. In practice, our experiments use a grid of densities, clamped to $[0, 1]$ to avoid a few densities dominating the sampling probability. This importance sampling method comes with almost no added cost since we store the densities or weights along the rays already used for volume rendering, and use a small 20^3 grid. This grid informs the selection of the 3D cube center, and the size is randomly chosen in a range of 1-10% of the scene, with a voxel resolution of 32^3.

Density Score Distillation Sampling (DSDS). Our diffusion model is trained on discretized synthetic data in $\{-1, 1\}$ indicating free or occupied space, respectively. NeRF densities, on the other hand, are in $[0, \infty)$, where low densities indicate free space and larger densities mean more occupied space. In practice, we observe that densities less than 0.01 are mostly free space, whereas occupied space have density values ranging from $[0.01, 2000]$.

We propose a Density Score Distillation Sampling (DSDS) loss that handles the domain gap between the densities.

Given a cube of NeRF densities σ, we discretize the densities into binary occupancy: $x_t = 1$ if $\sigma > \tau$ else -1 for diffusion timestep t, where τ is a hyperparameter that decides at what density to consider a voxel for empty or occupied. The Nerfbusters diffusion model then predicts the denoised cube x_0. The timestep t is a hyperparameter that determines how much noise the diffusion model should remove and can be interpreted as a learning rate. In practice, we choose a small $t \in [10, 50]$. With the denoised cube x_0, we penalize NeRF densities that the diffusion model predicts as empty or increase densities that the diffusion model predicts as occupied with:

$$\mathcal{L}_{\text{DSDS}} = \sum_i m_i \sigma_i + (1 - m_i) \max(w - \sigma_i, 0), \tag{3.2}$$

where $m = \mathbb{1}\{x_0 < 0\}$ is a mask based on the denoised predictions. We penalize densities where the diffusion model predicts emptiness and increase densities where the model predicts occupancy. w is a hyperparameter that determines how much to increase the densities in occupied space. The max operator ensures that no loss is applied if an occupied voxel already has a density above w. Similar to SDS [165, 234], the DSDS loss distills the diffusion prior with a single forward pass and without backpropagating through the diffusion model. Unlike SDS, our DSDS loss does not add noise to the original sampled occupancy before providing it to the diffusion model. While one may imagine that occupancy grids sampled during the NeRF optimization process do not exactly match the distribution of noised cubes used in training, we find that the denoising process nevertheless produces plausible samples that are both on the manifold of clean surfaces and similar in content to the input cubes.

Why not just... use a differentiable function to convert densities to the valid range of the diffusion model, then compute the SDS loss [165, 234], and then backpropagate through the activation function? This would require a function $s : \sigma \to x_t$ to map $s(0) = -1$, $s(\tau) = 0$, and $s(2\tau) = 1$, where τ is the crossing value where densities begin to be occupied. A scaled and shifted sigmoid function or a clamped linear function satisfies these requirements, but both have very steep gradients in some regions and no gradients in other regions, resulting in issues when backpropagating. In contrast, DSDS has gradients for any density predicted to be empty or occupied. In practice, we set $\tau = w = 0.01$ meaning our method deletes densities at points predicted to be empty and otherwise leaves the points unconstrained for the NeRF RGB loss to freely optimize.

Why not just... use accumulated weights, which are in the range [0, 1]? Weights are more well-behaved than densities but more expensive to compute as they require shooting a ray through the scene, evaluating and accumulating the densities along a ray. This results in significantly more function calls, but more fundamentally, requires one to specify a view from which to shoot the rays. This limits the diffusion prior to improving regions that are visible regions from the chosen view. A similar issue arises when using 2D or 2.5D priors [152, 257], where they may not regularize occluded regions unless viewpoints are chosen in a scene-specific way.

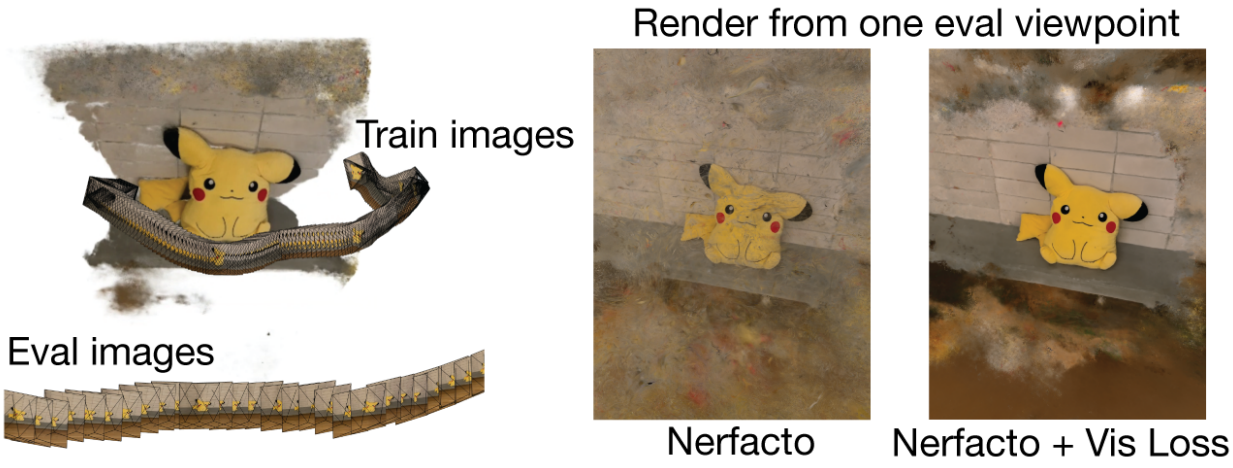

Figure 3.6: **Visibility loss.** Our visibility loss enables stepping behind or outside the training camera frustums. We accomplish this by supervising densities to be low when not seen by at least one training view. Other solutions would be to store an occupancy grid [145] or compute ray-frustum intersection tests during rendering. Our solution is easy to implement and applicable to any NeRF.

Visibility Loss

Our proposed local 3D diffusion model improves scene geometry and removes floaters, but it requires decent starting densities since it operates locally and thus needs contextual information to ground its denoising steps. To this end, we propose a simple loss that penalizes densities at 3D locations that are not seen by multiple training views. We find this simple regularizer effective in removing floaters from regions outside the convex hull of the training views. We define our visibility loss as

$$\mathcal{L}_{\text{vis}} = \sum_i V(q_i) f_\sigma(q_i) \tag{3.3}$$

where $f_\sigma(q_i) = \sigma_i$ is the NeRF density at the 3D location q_i, and $V(q_i) = \mathbb{1}\{\sum_{j=1} v_{ij} < 1\}$ indicates if the location is not visible from any training views. We approximate the visibility $v_{ij} \in \{0, 1\}$ of the i'th 3D location in the j'th training view with a frustum check. This approximation does not handle occlusions, instead overestimates the number of views a location is visible from. This loss penalizes densities in regions not seen by training images.

In practice, we implement this by defining a single tight sphere around our training images and render batches of rays that shoot from a random location on the sphere surface, through the center of the scene, and far off into the distance. We render rays with Nerfacto and apply this loss to the sampled points. Nerfacto uses a proposal sampler [11] to importance sample around surfaces, so our loss is effective in quickly culling away any floating artifacts

Table 3.1: **Quantitative evaluation.** NeRFs suffer when rendered away from the training trajectories. Existing regularizers do not suffice to improve the geometry. Nerfbusters learns a local 3D prior with a diffusion model, which removes floaters and improves the scene geometry. Results are averaged across 12 scenes.

	PSNR↑	SSIM↑	LPIPS↓	Depth↓	Disp.↓	Mean°↓	Med.°↓	% 30°↑	Cov.↑
Nerfacto Pseudo GT	25.98	0.8591	0.1019	0.0	0.0	0.0	0.0	1.0	0.89
Nerfacto	17.00	0.5267	0.3800	126.28	1.510	60.63	54.64	0.25	**0.90**
+ Visibility Loss	17.81	0.5538	0.3432	100.06	1.041	57.73	51.34	0.28	0.85
+ Vis + Sparsity [269]	17.81	0.5536	0.3445	92.17	1.145	57.77	51.40	0.28	0.85
+ Vis + TV [52]	17.84	0.5617	0.3409	74.02	0.382	61.93	56.16	0.24	0.84
+ Vis + RegNeRF [152]	17.49	0.5396	0.3585	182.45	1.200	59.39	53.27	0.27	0.86
+ Vis + DSDS (Ours)	**17.99**	**0.6060**	**0.2496**	**54.45**	**0.11**	**54.77**	**47.98**	**0.30**	0.63

Table 3.2: **Ablation study.** Ablation on the "garbage" scene for different settings of using our 3D prior as a NeRF loss. Cube sampling refers to uniformly sampling the entire scene versus importance sampling with accumulated weights or densities.

Cube sampling strategies

	PSNR	SSIM	Disp.	Mean °	Cov.
Uniform	14.61	0.4276	10.288	61.52	**0.886**
Densities σ	**16.46**	**0.5086**	**0.081**	**49.21**	0.606
Weights	15.86	0.4466	0.112	53.09	0.634

Activation functions

	PSNR	SSIM	Disp.	Mean °	Cov.
Clamp+SDS	12.53	0.2652	2.065	87.33	**1.000**
Sigmoid+SDS	12.53	0.2652	2.065	87.33	**1.000**
σ_τ+DSDS	**15.86**	**0.4466**	**0.112**	**53.09**	0.634

Cube size range as % of scene

	PSNR	SSIM	Disp.	Mean °	Cov.
1-20%	**17.05**	**0.5005**	**0.083**	54.87	0.600
10-20%	16.93	0.4884	0.090	**50.78**	**0.640**
1-10%	15.86	0.4466	0.112	53.09	0.634

with high density outside visible regions. See fig. 3.6 for a qualitative result where we render from behind training images.

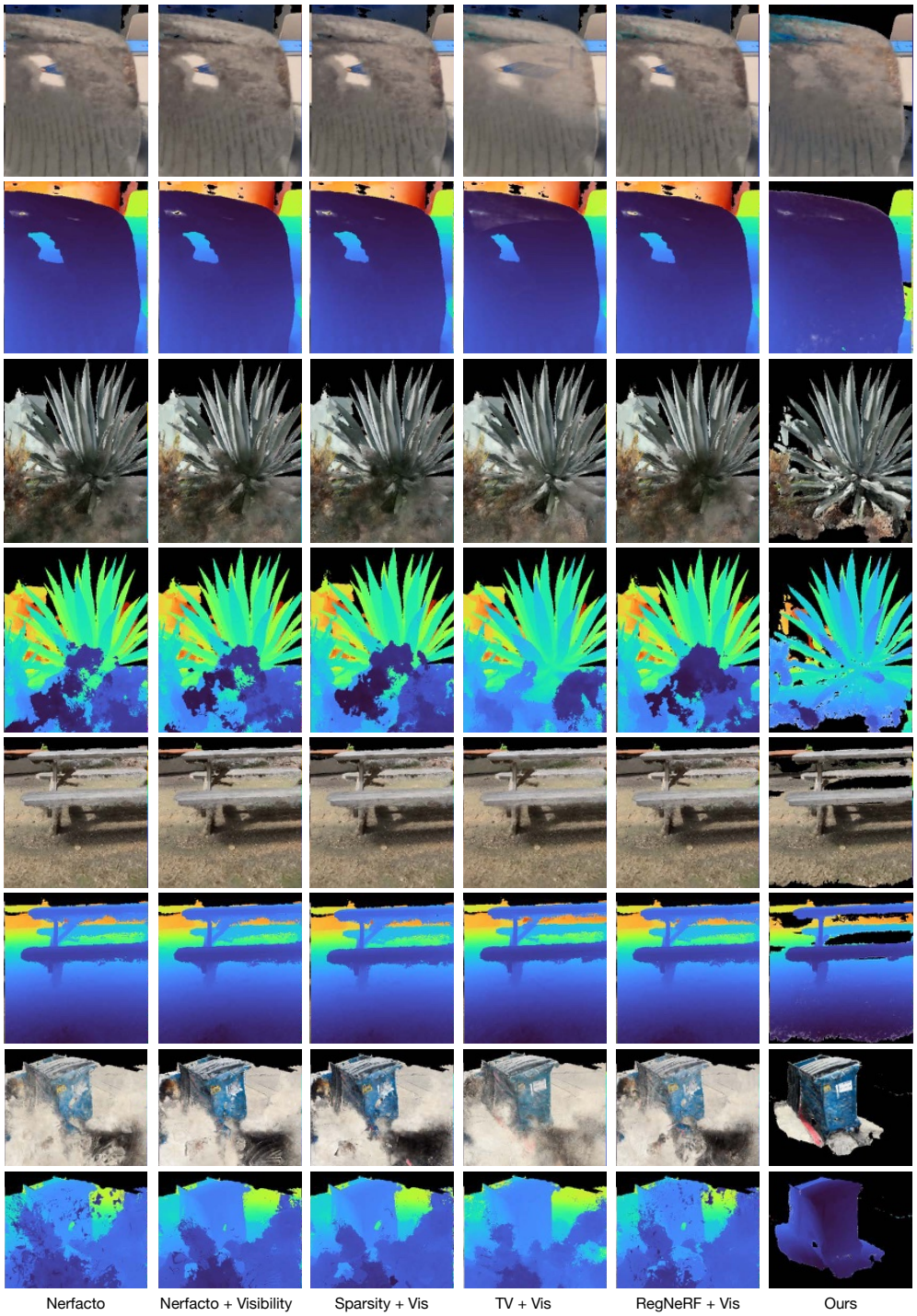

Figure 3.7: **Qualitative results.** NeRFs suffer from floaters and bad geometry when rendered away from training views. We show rendered RGB and depth from four scenes. In contrast to existing hand-crafted regularizers, our proposed diffusion prior fills holes (first scene), removes floaters (second and fourth scenes), and improves geometry (all scenes). Please see the associated website for video results on our evaluation splits.

Figure 3.8: **Ablations results.** Using a simple activation function and SDS results in a not-well-behaved gradient signal, increasing the number of floaters in the scene. Importance sampling more effectively applies the 3D cube loss in space, cleaning up floaters and improving the scene geometry.

3.5 Experiments in-the-wild

In this section, we follow our proposed evaluation protocol described in Sec. 3.3 to ablate and compare the proposed method with a number of common regularizers aimed at cleaning up NeRFs.

Implementation details. For each experiment, we use the Nerfacto model within the Nerfstudio [214] codebase. We turn off pose estimation for evaluation purposes and then train Nerfacto for 30K iterations which takes up to half an hour. We then fine-tune from this checkpoint with different regularizer methods. We compare the proposed method with vanilla Nerfacto, Nerfacto with the proposed visibility loss, Nerfacto with our visibility loss and 3D sparsity loss [269], 3D TV regularization [52], and 2D TV (as in RegNeRF [152]). Our implementations also use the distortion loss [11] which is on by default with Nerfacto. All methods are effective within the first 1K iterations of fine-tuning ($\sim$4 minutes on an NVIDIA RTX A5000 for Nerfbusters), but we train for 5K iterations. For the 3D baselines, we sample 40 32^3 cubes per iteration and for the 2D baseline RegNeRF, we render ten 32^2 patches. The usual NeRF reconstruction loss is also applied during fine-tuning with 4096 rays per batch.

Results. table 3.1 shows that visibility loss improves vanilla Nerfacto across all quality metrics. Existing hand-crafted regularizers do not improve upon this baseline. In contrast, our data-driven local diffusion prior removes floaters and improves the scene geometry, yielding state-of-the-art results on these challenging casual captures. The proposed method deletes floaters, and thus we find that it has lower coverage than the baselines. Our sup-

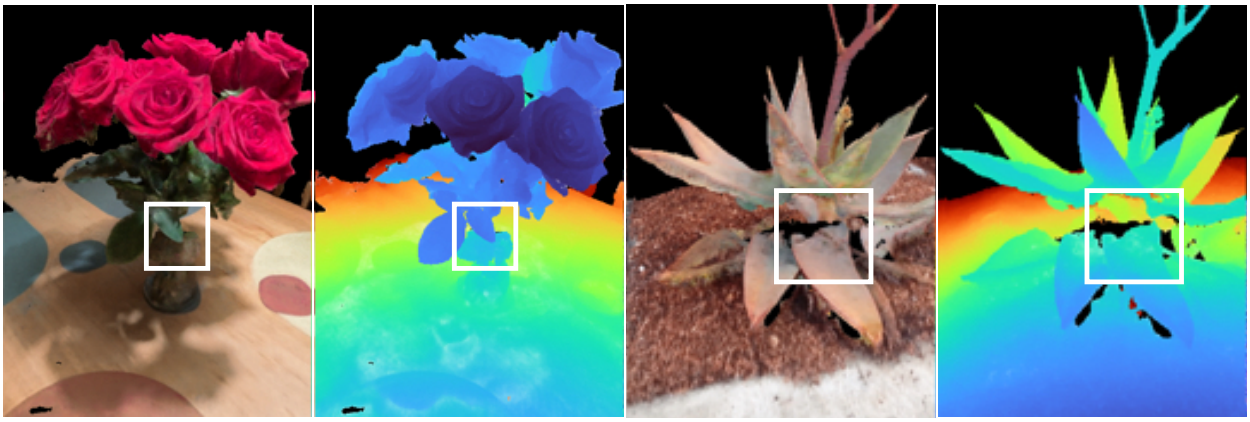

Figure 3.9: **Limitations.** The proposed model only operates on densities, which comes with some limitations. We find that it cannot distinguish floaters from transparent objects (left). It does not hallucinate texture and thus ends up removing regions that are occluded in all training views (right).

plementary material shows per scene results. fig. 3.7 shows a qualitative comparison of the methods for both indoor and outdoor scenes. We find that our method improves geometry by completing holes (see the chair in the first row), removing floaters (see in front of century plant in the second row and garbage truck in the fourth row), and sharping geometry (see the under the bench in the third row).

Ablations of our 3D prior on real data We ablate our method on the "garbage" scene (table 3.2). We find that the cube sampling strategies (i.e., where to apply the diffusion prior) are important, and using the proposed importance sampling with densities yields the best performance. fig. 3.8 compares uniform sampling with importance sampling (using densities). Importance sampling samples less empty space, and thus is more effective at cleaning up floaters and scene geometry. We compare the proposed DSDS loss against SDS with either a scaled and shifted sigmoid or a clamped sigmoid that satisfies our requirements (see section 3.4). We find the gradients do not flow well through this activation function resulting in a distorted scene with many floaters (see fig. 3.8 left). We also ablate the cube sizes used cubes size ranging from 1% to 20% of the scene scale. We find that our method is relatively robust to the cube sizes, yielding a trade-off between removing more with larger cubes and removing less with smaller cubes.

3.6 Conclusion and future work

Transparent objects. NeRFs are able to represent transparent objects by assigning low densities to the transparent object. These transparent densities behave similarly to floaters, and it requires semantic information to distinguish the two. Since our local diffusion prior

does not have semantic information, it removes transparent objects as illustrated in the vase in fig. 3.9.

Hallucinating texture. The proposed method cleans geometry but cannot edit texture, as our method operates on densities. This means that we can remove regions that contain floaters or fill holes, but we cannot colorize these regions. We leave colorization and inpainting low-confidence regions to future work, where 2D diffusion priors [165, 134] or 3D-consistent inpainting [118, 112] may be relevant.

Conclusion. We propose a new evaluation procedure of Neural Radiance Fields (NeRFs) that better encompasses how artists, designers, or hobbyists use the technology. We present a dataset with 12 captures recorded with two camera trajectories each, one used for training and one for evaluation. We find that current hand-crafted regularizers are insufficient when NeRFs are rendered away from the training trajectory. We propose a data-driven, local 3D diffusion prior, Nerfbusters, that removes floaters and improves the scene geometry. We have implemented our proposed evaluation procedure and method in the widely adopted codebase Nerfstudio and it is released for the benefit of the community.

Acknowledgements

The authors of this paper are Frederik Warburg*, Ethan Weber*, Matthew Tancik, Aleksander Holynski, and Angjoo Kanazawa. * denotes equal contribution. This project is funded in part by the Hellman Foundation and BDD sponsors. We thank our colleagues for their discussions and feedback throughout the project, especially those within the Kanazawa AI Research (KAIR) lab and those part of the Nerfstudio development team. We thank Kamyar Salahi, Abhik Ahuja, Jake Austin, Xinyang Han, and Alexander Kristoffersen for helping to improve paper drafts.

Chapter 4

Nerfiller: Scene completion

We propose NeRFiller, an approach that completes missing portions of a 3D capture via generative 3D inpainting using off-the-shelf 2D visual generative models. Often parts of a captured 3D scene or object are missing due to mesh reconstruction failures or a lack of observations (e.g., contact regions, such as the bottom of objects, or hard-to-reach areas). We approach this challenging 3D inpainting problem by leveraging a 2D inpainting diffusion model. We identify a surprising behavior of these models, where they generate more 3D consistent inpaints when images form a 2×2 grid, and show how to generalize this behavior

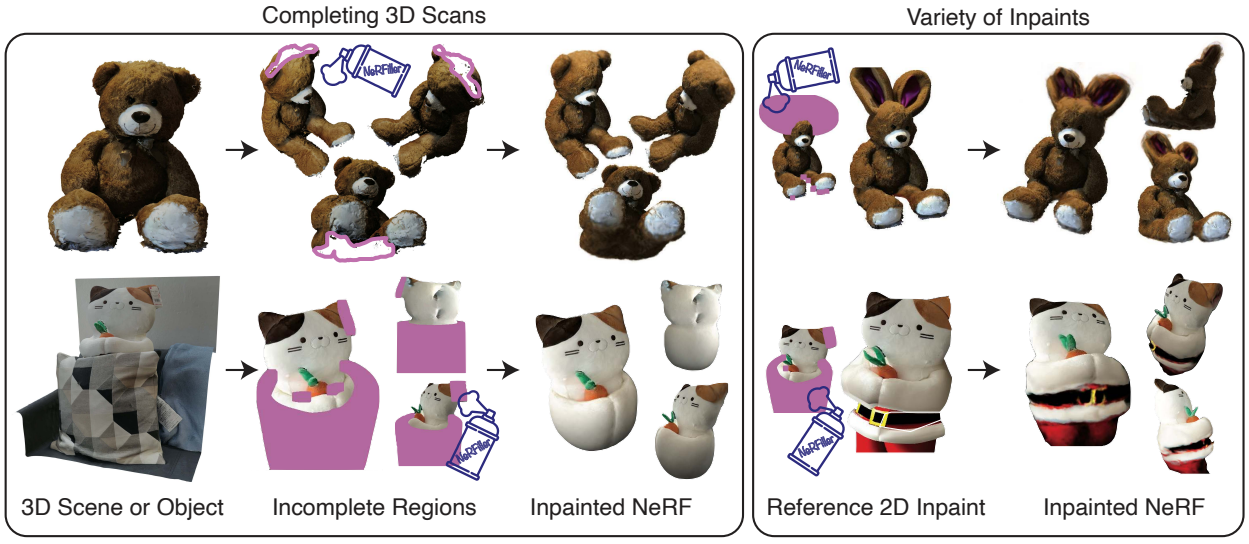

Figure 4.1: **NeRFiller.** We propose a generative 3D inpainting approach for scene or object completion. Given a 3D capture with incomplete regions (left), our approach completes scenes such as the incomplete teddy bear scan (top left) and deletes unwanted occluders such as the pillow and the price tag (bottom left). We can also control the completions using a reference inpainted exemplar (right) to guide the process.

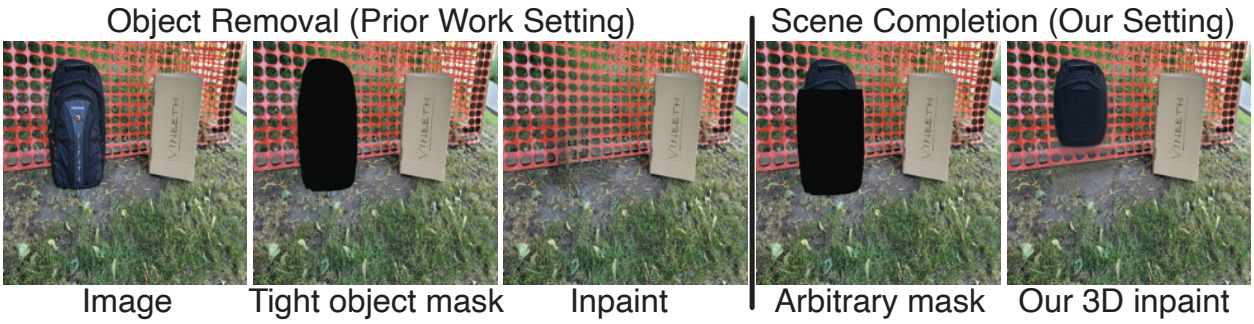

Figure 4.2: **Object removal vs. scene completion.** We focus on scene completion (right) as opposed to object removal (left). Prior work focuses on removing entire objects with tight masks, while we tackle the more general setting of completing scenes with arbitrary missing regions across wide baselines. More realistic scenarios include missing regions or parts of scenes to edit, as illustrated in Figure 4.1.

to more than four images. We then present an iterative framework to distill these inpainted regions into a single consistent 3D scene. In contrast to related works, we focus on completing scenes rather than deleting foreground objects, and our approach does not require tight 2D object masks or text. We compare our approach to relevant baselines adapted to our setting on a variety of scenes, where NeRFiller creates the most 3D consistent and plausible scene completions. Our project page is at https://ethanweber.me/nerfiller.

4.1 Introduction

Consider the 3D scanned teddy bear and cat in Figure 4.1. In many 3D captures such as these, parts of the scene may not be as one desires: there may be unobserved regions such as the bottom of the bear and behind the cat, or there may be unwanted parts such as the price tag on the cat ear. Additionally, one may want to modify a feature, or generate a variety of alternative models, e.g., a bear with bunny ears or a santa cat. All of these tasks require the ability to edit and inpaint content in a 3D-aware and multi-view consistent manner. This is a challenge, since 2D generative inpainting models will not by default generate 3D consistent images. Our goal is to take a step in this direction and present a method that can create new content via scene completion conditioned on a set of multi-view images.

Specifically, we present a 3D scene completion framework called *NeRFiller*, which given a scene and specified parts of the scene to inpaint, returns a 3D scene that is completed in a multi-view consistent manner. Our approach not only completes missing regions (Figure 4.1, center), but can also generate multiple variations of the missing regions (Figure 4.1, right). Furthermore, our approach does not require text prompting and can operate from the scene context alone.

We achieve this by proposing a novel approach to generate inpaints with an off-the-shelf 2D generative image model in a manner that encourages multi-view consistency. Specifically, we identify a useful phenomenon in text-to-image diffusion models that we refer to as a *Grid Prior*: denoising four images with missing observations that are tiled in a 2×2 grid results in more consistent multi-view inpaints than inpainting them independently, shown in Figure 4.3. We propose a method called *Joint Multi-View Inpainting* that generalizes this behavior to more than four images. While this technique results in more 3D consistent inpaints, it is still a 2D-based approach and 3D consistency is not guaranteed. Therefore, we propose a way to distill these inpaints in a global 3D scene representation in an iterative manner.

While there has been a surge of recent works that generate 3D scenes completely from scratch using text [50, 78] or image guidance [118, 112], our approach differs in that we focus on completing scenes given the context of an existing 3D scene. Our approach is related to recent methods that remove a specified object from a scene [140], but we can generate new content that goes beyond completing a textured background, as illustrated in Figure 4.2. We also do not assume a tight object mask, and can generate a diverse set of inpaints.

To demonstrate the efficacy of our approach, we experiment with a diverse set of scenes including 3D indoor photogrammetry captures lacking coverage in certain areas, 3D scenes with specified missing regions, and 3D objects. While our problem is challenging, we show that NeRFiller can recover more 3D consistent and plausible results compared to recent state-of-the-art methods adapted to our setting.

4.2 Related Work

Our goal is to complete missing parts of an existing 3D scene. There are several ways to approach this, via 2D inpainting or via distilling a 2D generative model for 3D generation.

2D inpainting. 2D inpainting methods take an image and mask and complete the missing content at the mask location. Early methods relied on inpainting by copying texture from known regions into the unknown regions [44]. A state-of-the-art model is LaMa (Large Mask inpainting) [211], which is particularly good at infilling large missing areas. It uses fast Fourier convolutions, a large receptive field, and large training masks. This model is highly effective at completing plausible "background" textures within a specified mask (Fig. 4.2 left) but lacks diverse outputs as it is deterministic. Probabilistic diffusion models [74, 163] have recently produced remarkable results for image generation. They can also be used for inpainting and can generate diverse inpainted outputs. Pixel-based diffusion models do not have to be trained explicitly for inpainting, but can be modified at test-time by setting known regions before each denoising iteration [127]. Latent diffusion models (LDMs) [181] are also effective at inpainting and are efficient because they operate in latent space. However, they require fine-tuning for inpainting with image and mask conditioning. 2D inpainting models can be prompted [9] and/or fine-tuned [215, 96] enabling additional flexibility for downstream applications.

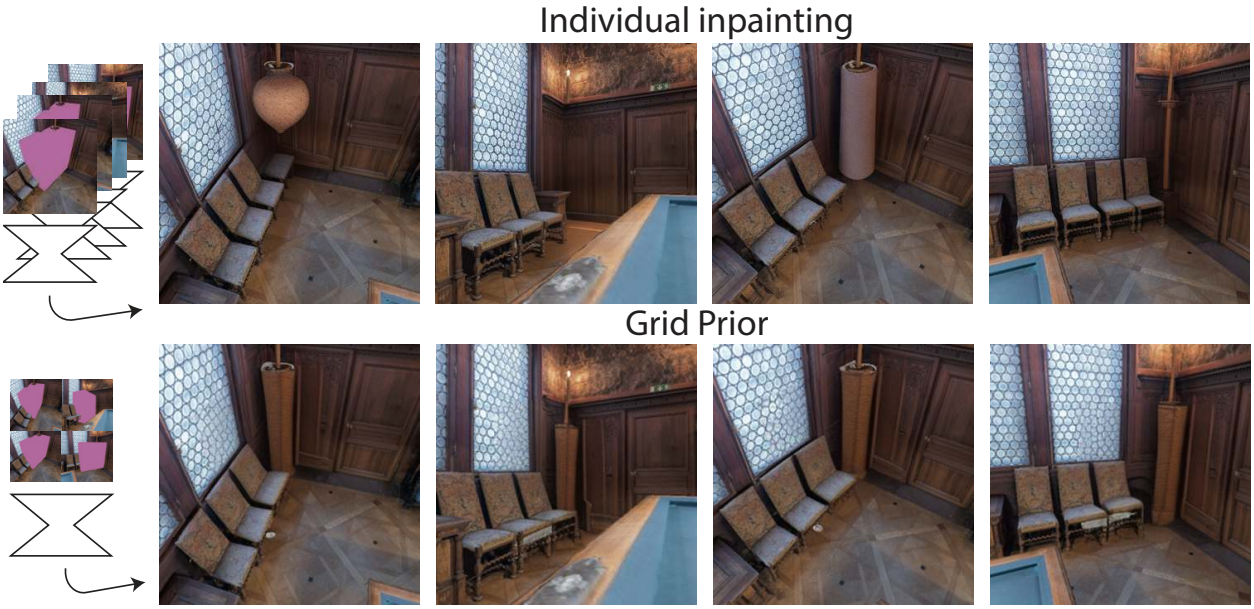

Figure 4.3: **Grid Prior.** Here we inpaint the corner of the room (left illustrated in pink) with individual inpainting (top) and our Grid Prior method (bottom). Individual inpaints are diverse, while the Grid Prior encourages multi-view consistency.

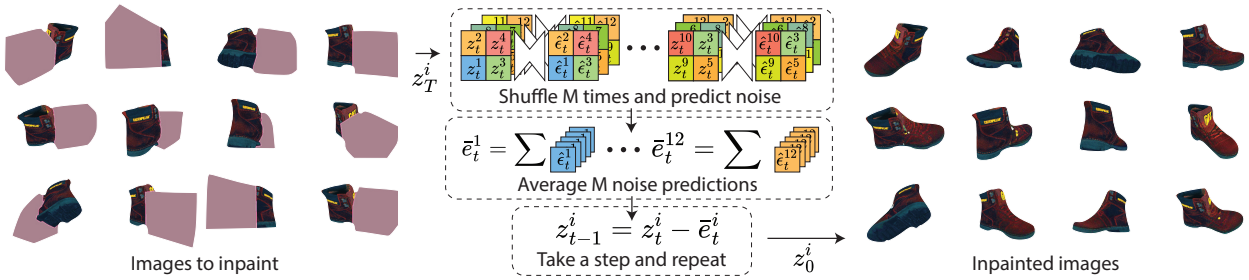

Figure 4.4: **Joint Multi-View Inpainting.** We use the Grid Prior with more than four images by averaging diffusion model predictions. We take N images (left), create $N/4$ grids, and obtain a noise prediction from SD [181]. We do this M times and average the noise predictions before taking a denoising step. At z_0, the images (right) are fairly consistent and can be used to train a NeRF with our Inpaint DU method.

3D generation. 3D generation takes as input text or images and outputs 3D content. The Infinite Nature line of work [118, 112, 106, 253, 179], takes as input a single image and generates immersive fly-through content using a 2D inpainting model queried in an autoregressive manner [118, 112]. Cai et. al. [21] follow this path with a diffusion model, however, none of these approaches can recover a global 3D scene representation. Persistent

Nature [25] and related work [32, 41] maintain a latent scene but are completely generative and not conditioned on input image sets. SceneScape [50] and Text2Room [78] use text prompts and 2D inpainters to create a 3D mesh by using an inpainter and depth predictor to successively stitch a mesh. These approaches cannot fix a mistake in the scene if a bad inpaint is made during the successive stitching because no global optimization is performed.

Other methods create 3D content via a global optimization strategy. DreamFusion [165] and related works [242, 283] use the NeRF framework to optimize a 3D volume given a text prompt. Others train models to have 3D consistent properties [283, 248, 216, 110, 125]. Follow-up works leverage 2D diffusion models techniques [184, 54] to create 3D content conditioned on real images [169, 134, 26]. These approaches are not designed for the inpainting task.

3D inpainting. Unlike most 3D generation methods, we ground our inpaints with an actual 3D scene or object that has missing regions (Figure 4.1). Casual capture [135, 19, 221, 72, 71] or NeRFs [138, 100] is a use-case as they often contain artifacts when rendered from novel views [246, 61]. Most relevant to scene the completion setting is the object removal setting (see Figure 4.2). These works remove foreground objects from NeRF captures [232, 140, 251]. They do this by inpainting each image in a NeRF dataset once and training with various losses including patch-based perceptual losses and depth regularization. [139] enables inpainting from a reference image. A variety of these methods are evaluated on the SPIn-NeRF dataset, which is in the forward-facing LLFF [135] format and has small parallax. Our work uses datasets with a significantly larger baseline.

Our focus is on the more general scene completion setting, which is related to editing. IN2N [65] edits a scene using InstructPix2Pix [17], but it cannot hallucinate new geometry. The video editing literature is also relevant, with techniques such as extended attention from Tune-A-Video [255, 59] to encourage consistency in edited video frames. However, editing 2D images does not guarantee consistency when lifted to 3D. In our method, we encourage 2D inpaints to converge via iterative NeRF optimization and dataset updates. Moreover, we achieve this with *off-the-shelf* 2D generative models without the need for expensive purpose-trained diffusion models or model fine-tuning.

4.3 Preliminaries

Neural Radiance Fields (NeRFs)

Neural radiance fields (NeRFs) [138] represent the 3D geometry and radiance of a scene with neural networks. NeRFs take as input an 3D position (x, y, z) and a viewing direction (θ, ϕ), and output a color and density (c, σ). To train a NeRF f_Θ, a set of calibrated, posed images are used to construct a set of 3D rays $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ for each pixel with known color $C(\mathbf{r})$. During training, these rays are sampled and rendered via volumetric rendering to obtain a color estimate $\hat{C}(\mathbf{r})$. Rays are sampled from training images and the field is optimized

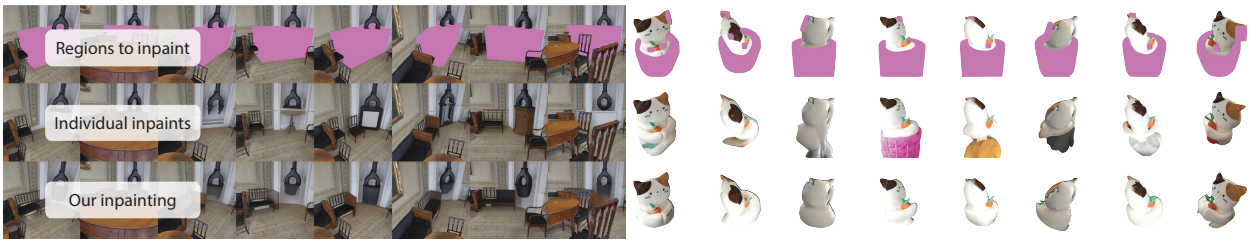

Figure 4.5: **Joint Multi-View Inpainting Examples.** The top images are inpainted with SD [181] without any text conditioning (middle) and with our Joint Multi-View Inpainting method (bottom). Our joint inpaints are more multi-view consistent.

with photometric losses $\mathcal{L}_{nerf}(C(\mathbf{r}), \hat{C}(\mathbf{r}))$, e.g., MSE or LPIPS [278]. During inference, a full image is rendered with all rays of the desired camera.

2D Diffusion Models

Diffusion models consist of two processes: a forward process q that gradually adds noise to a data sample $z_0 \sim p_{data}(z)$, and a learned reverse process to iteratively denoise a pure Gaussian noise sample $z_T \sim \mathcal{N}(0, 1)$ into a clean image z_0. An intermediate noisy z_t can be obtained from the clean image by adding noise ϵ with scaling $\bar{\alpha}_t$, where $z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$. The diffusion model ϵ_θ predicts noise $\hat{\epsilon}$ present in the image z_t as $\hat{\epsilon} = \epsilon_\phi(z_t, t, c)$. t is a time indicating how much noise is in the sample, and c is a general form of conditioning (e.g., images, masks, or text). During training, random noise ϵ and t are sampled and the objective $\mathcal{L}_{diff} = ||\hat{\epsilon} - \epsilon||^2$ is minimized. With the prediction $\hat{\epsilon}$ at time t, a reduced noise z_{t-1} can be obtained by $z_{t-1} = z_t - \hat{\epsilon}$ (where we omit the scaling of $\hat{\epsilon}$ for simplicity). Repeating this until z_0 yields a fully denoised sample. Stochastically training with c (conditionally) and without c (unconditionally) enables classifier-free guidance (CFG) [75] during inference time. In practice, z is latents since we are using SD (Stable Diffusion) [181], but in general we can map from z to higher resolution pixels x with an encoder $\mathcal{E}(x)$ and decoder $\mathcal{D}(z)$.

Using a diffusion model as a prior. Diffusion models have advantages over other models (e.g., GANs and deterministic inpainters [211]) because they can be used a a prior to optimize underlying variables such as the parameters of a 3D NeRF f_Θ with methods like score distillation sampling (SDS) [165, 234]. When used as a prior for NeRFs, the objective is to find the best Θ such that a rendered image x has high likelihood under the diffusion model prediction ϵ_ϕ. SDS involves rendering an image, adding partial noise, and updating the NeRF such that the diffusion model can predict the added noise. IN2N [65] introduced a variant of this method coined Dataset Update (DU). Instead of backpropping based on the diffusion model prediction, DU renders an image, adds partial noise, and takes multiple steps to recover an estimated clean image x_0. The clean image is added to the dataset and used to supervise the NeRF. Every S iteration, another image is replaced. The DU supervision signal will be slightly delayed since images are cached for several iterations, while SDS provides

immediate gradients corresponding to the current render. However, we use the DU method in our work because it has a few advantages over SDS in terms of implementation: We can obtain higher-resolution supervision (albeit slightly delayed) with less GPU memory and we can update a large batch of images simultaneously (e.g., 40 images). We find that large batch updates are important for our inpainting task since we are changing the NeRF geometry, unlike prior works that focus purely on modifying appearance [65, 150].

4.4 Method

Our method, NeRFiller, aims to complete a missing region within a 3D scene by using an inpainting prior from a generative 2D diffusion model. This problem statement poses a number of challenges. First, the inpainted estimates from a 2D diffusion model are diverse, and may vary from sample to sample. This requires a consolidation mechanism to ensure that the completed 3D scene contains one salient inpainted result, as opposed to the average of all possibilities. Second, 2D inpainting models are not trained for 3D consistency, and will therefore provide estimates that cannot be explained by a single 3D scene, even if they correspond to the same approximate style or content. In the following, we describe our approaches to tackle these problems. In Section 4.4, we describe how we encourage the inpainted outputs from a diffusion sampling process to be 3D consistent. In Section 4.4, we describe an iterative 3D scene optimization method that uses these inpainted images to optimize for a globally consistent inpainted 3D scene. NeRFiller builds on an observation that inpainting a grid of images encourages the outputs to have similar appearance, and we extend this idea to an arbitrarily large collection of images through a joint sampling approach.

Multi-view consistent inpainting

A core challenge in 3D inpainting with a 2D generative model is getting the outputs of the 2D model to be consistent across views. This challenge stems from the multimodality of the output distribution: in most cases, there are many plausible inpaintings, and sampling multiple consistent images remains an open research problem.

Grid Prior. While inpainting multiple viewpoints independently may produce inconsistent results, one interesting discovery is that consistency can be achieved by tiling the input images into a grid and treating the *grid* of images (and their corresponding masks) as a single inpainting target. This grid-based prior can produce more 3D consistent views, both in coarse appearance and approximate scene structure (illustrated in Figure 4.3). We hypothesize that this phenomenon results from similarly structured examples in Stable Diffusion's training dataset: sets of observations depicting the same scene or object organized as a grid (e.g., screenshots of online product photos). Similar properties were also explored in visual-prompting [9].

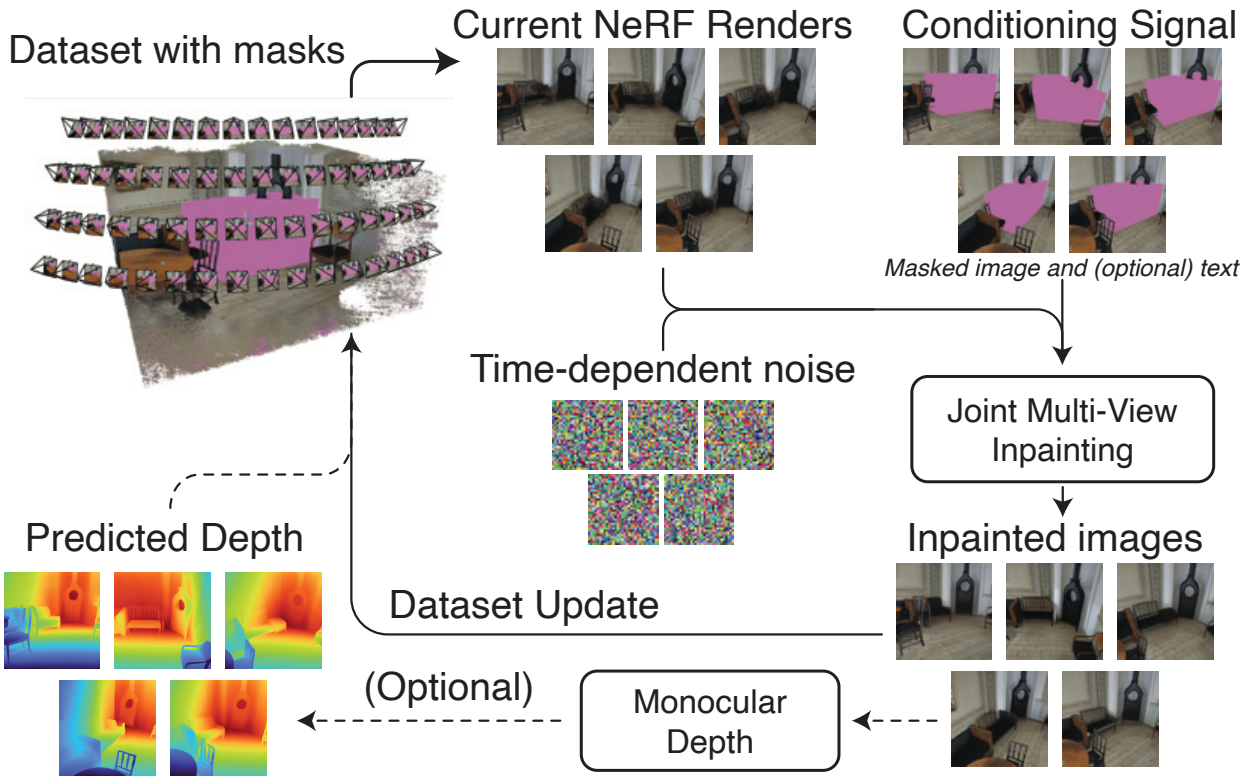

Figure 4.6: **Inpaint Dataset Update.** Every S iterations, we update the unknown pixels of the NeRF training images. We render N images, add partial noise, and jointly inpaint with a conditioning signal. We (optionally) predict the depth and update the dataset.

More specifically, in order to inpaint four images consistently, one can downsample them and their corresponding inpainting masks to quarter-resolution and tile them as a 2×2 grid. This grid is fed through the 2D inpainting model (as a single image would) to get as output four inpainted images with consistent content. More formally, let $\mathcal{G}$ be the downsampling and grid operation for four images and let $\mathcal{G}^{-1}$ undo this. We can grid four images latents $z_t^1, z_t^2, z_t^3, z_t^4$ as follows:

$$\{\hat{\epsilon}_t^1, \hat{\epsilon}_t^2, \hat{\epsilon}_t^3, \hat{\epsilon}_t^4\} = \mathcal{G}^{-1}(\epsilon_\phi(\mathcal{G}(\{z_t^1, z_t^2, z_t^3, z_t^4\}))) \tag{4.1}$$

and take a denoising step with $z_{t-1}^i = z_t^i - \hat{\epsilon}_t^i$. In principle, this approach is similar to recent methods that use *extended attention*, i.e., shared keys and values in the attention operations across a set of parallel sampling processes [255]. Our approach does not share attention features but instead shares context with other images via the diffusion U-Net receptive field that sees 4 tiled images at a time. In our experiments, we compare to *extended attention* and demonstrate that our grid prior more effectively inpaints 3D consistent content when used with our Joint Multi-View Inpainting method.

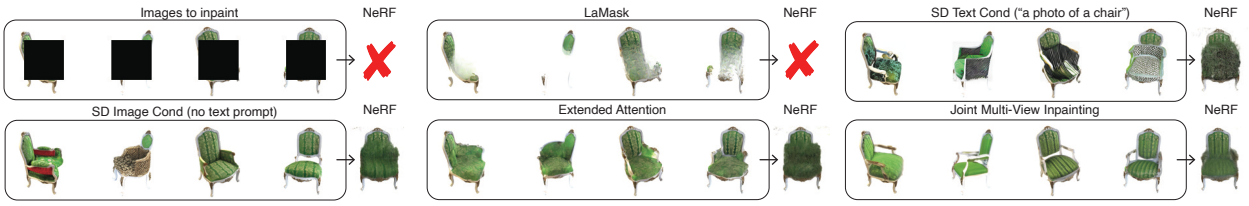

Figure 4.7: **Inpainting methods.** Inpainting methods produce inconsistent inpaints. We show various inpainting methods (boxed) and use a collection of them to train a NeRF. A resulting render is shown on the right of each method. Using our Grid Prior and Joint Multi-View Inpainting creates reasonably consistent inpaints and a plausible NeRF. $\mathcal{X}$ means the NeRF failed and resulted in white everywhere.

Table 4.1: **Multi-view consistent inpainting.** We inpaint images and train a NeRF for the 8 scenes of the NeRF synthetic dataset. Better metrics indicate more consistency of the NeRF 3D reconstruction with the 2D inpaints. Note that LaMask achieves the best results as it often copies the white background into the hole (see Figure 4.7) and results in a failed NeRF that is totally white.

	PSNR ↑	SSIM ↑	LPIPS ↓
Masked NeRF	7.76	0.71	0.37
LaMask	19.58	0.89	0.20
SD Text Cond	12.56	0.73	0.32
SD Image Cond	14.15	0.76	0.28
Extended Attention	14.57	0.77	0.27
Grid Prior	14.43	0.80	0.25
Joint Multi-View Inpainting	15.89	0.82	0.23

Joint Multi-View Inpainting. While effective at inpainting a set of images consistently, applying the grid prior to a larger set of images poses additional challenges. Increasing the number of images in the grid proportionally decreases the output resolution of each image (e.g., arranging a set of 2×2 images in a grid reduces each image's resolution by 4, a 3×3 grid by 9, and so on). For the purpose of producing high-quality inpainting results, we would like to minimize any loss in image detail. Therefore, we propose a method that uses the above grid prior in a joint sampling process, inspired by MultiDiffusion [10]. In each sampling step, we shuffle all the input images into a set of 2×2 grids. We repeat this M times and before taking a sampling step, the score estimate for each image is combined across all the grid combinations in which it was seen. This causes the inpainting estimates to be gradually shared across the entire dataset, effectively increasing the grid size without further reducing effective resolution.

Figure 4.4 describes this procedure. More formally, for a batch of N images, we randomly permute their order, construct $N/4$ grids, and predict the noise for M iterations ($j \in [1, M]$),

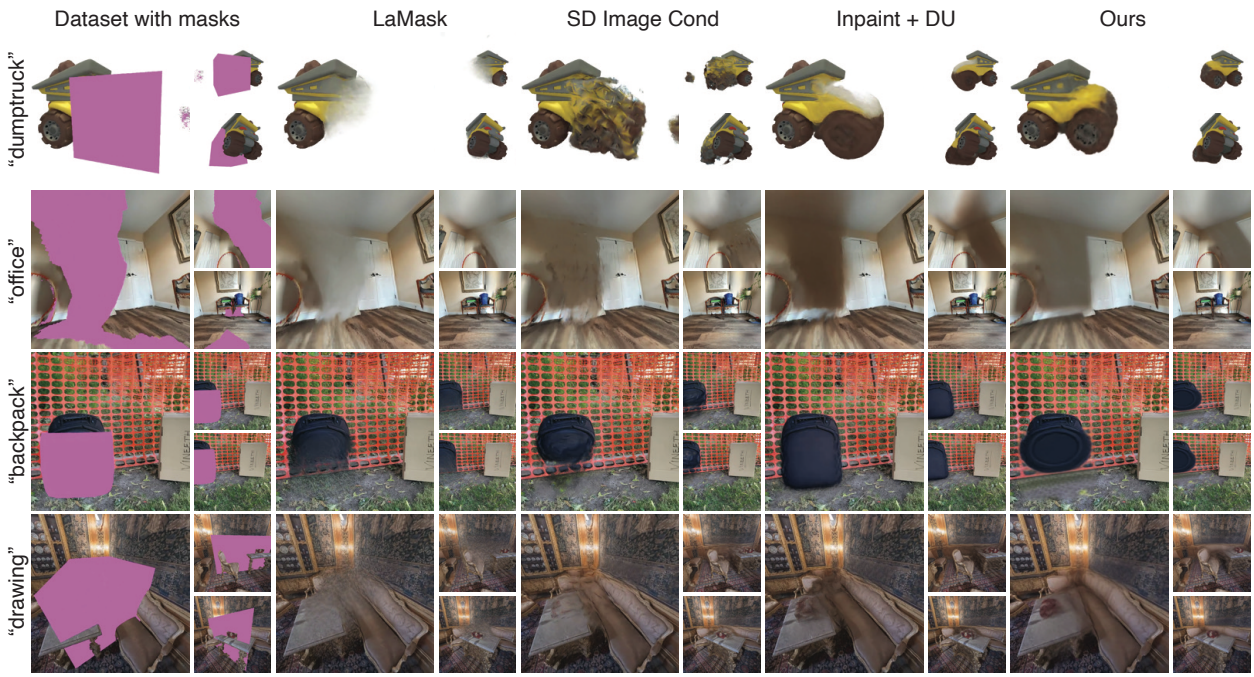

Figure 4.8: **Qualitative NeRF results.** On the left, we show various scenes with pink regions to be completed. We compare NeRFiller (far right) against baselines adapted to our scene completion setting. The "office" scene is missing parts of the wall, floor, and under the chairs.

as follows:

$$\{\hat{\epsilon}_t^{1j} \dots \hat{\epsilon}_t^{Nj}\} = \mathcal{G}^{-1}(\epsilon_\phi(\mathcal{G}(\{z_t^{1j} \dots z_t^{Nj}\}))) \tag{4.2}$$

and step with $z_{t-1}^i = z_t^i - \sum_{j \in M} \hat{\epsilon}_t^{ij}$ from z_T to z_0. Qualitative results are shown in Figure 4.5.

Completing 3D Scenes

The proposed Joint Multi-View Inpainting enables inpainting images in a more 3D consistent manner than other inpainting methods. Next, we describe how to distill these 2D inpainting results into a single 3D reconstruction. We refer to our method as Inpaint Iterative Dataset Update (Fig. 4.6), or **Inpaint DU**, as it derives from IN2N's [65] Iterative DU method (see Sec. 4.3). In contrast to IN2N, which begins with a complete NeRF reconstruction and uses a model conditioned on complete 2D observations, our task requires us to train a complete NeRF from images with masked unknown regions. As in IN2N, we begin training with a dataset of original (known) pixels, and update the dataset over training by adding or replacing the set of initially unknown pixels with the inpainted estimates. Fig. 4.6 illustrates this procedure. Specifically, every S steps, we render the set of N training views, encode

them into latents z_0^i and then partially noise them before feeding them to SD. We sample from these partially noised inputs using the proposed Joint Multi-View Inpainting strategy, then use the resulting images to replace the corresponding images in the dataset. This process is both prefixed and suffixed by an encode and decode operation, since the base model is a latent diffusion model. We repeat this process many times while linearly annealing t from full noise $t = 1$ to $t = t_{\text{min}}$. In practice, we set $t_{\text{min}} = 0.4$, since we find that low noise values result in quality degradation. We observe that over the course of optimization, our inpainted images become gradually more consistent (Fig. 4.9) as the added geometry and texture begin to take form. Annealing t helps encourage the inpainting to converge to a single result rather than making large changes late in training.

Depth regularization. Inpaint DU optionally incorporates depth supervision to improve inpainted scene geometry. After each dataset update, we predict the depth for all images with ZoeDepth [14]. We use a relative depth ranking loss [233] in the inpainted regions (but not on the known pixels). We use a ranking loss because it's a softer constraint than metric depth supervision, where errors in scale-and-shift alignment could more easily harm the 3D scene geometry. *We only apply depth supervision for our main method to indoor scenes and not objects, since we empirically noticed that [14] performs less consistently when the background is a solid color (e.g., white or black).*

4.5 Experiments

We compare NeRFiller for 3D scene completion to various inpainting baselines. We first investigate various inpainting strategies on multi-view synthetic scenes to gauge how effective a deterministic inpainter [211] is compared to SD [181], sampled in various ways. After establishing that our *Joint Multi-View Inpainting* demonstrates multi-view inpainting properties, we evaluate our full method on 10 scans with missing regions. We compare NeRFiller to various object-removal baselines, *adapted to our setting*, to complete missing regions. Finally, we analyze the parameters of our method and show an application to reference-guided scene completion. We conduct experiments with Nerfstudio [214] and provide implementation specifics in the appendix.

3D consistent image inpainting

Our goal is to evaluate various 2D inpainting models and strategies to quantify their 3D consistency.

Setting and evaluation. For these experiments, we take the testing split of the NeRF synthetic dataset [138] (8 scenes of 200 images each). We resize each image to 512×512 resolution and mask out the center of each image with a 256×256 region to inpaint (see Figure 4.7 top left). We inpaint all images with various methods and train a Nerfacto NeRF model [214] on 180 equally spaced images and evaluate metrics on the remaining 20 images. We use the standard NeRF metrics because they capture how similar the 3D reconstruction

Table 4.2: **Quantitative NeRF results.** We report various metrics averaged over our 10 scenes to quantify consistency of the 3D NeRF with the dataset inpaints (left), as well as novel-view metrics (right), such as "Corrs" (number of high-quality correspondences between random pairs of frames) for geometry.

	PSNR ↑	SSIM ↑	LPIPS ↓	MUSIQ ↑	Corrs ↑
Masked NeRF	14.71	0.78	0.26	3.71	675
LaMask	27.39	0.90	0.05	3.76	643
SD Image Cond	22.03	0.86	0.11	3.68	665
Inpaint + DU	26.60	0.89	0.08	3.76	660
Ours w/o depth	28.41	0.92	0.06	3.72	682
Ours	28.28	0.91	0.06	3.73	696

is to the 20 hold-out evaluation images. When rendering for evaluation (right of inpainted images in Figure 4.7), we push the near plane slightly forward to avoid including any floaters hiding in front of the cameras.

Baselines. Our baselines are the following:

- *Masked NeRF* - No inpainting, only train on known pixels.
- *LaMask* - LaMa [211] inpainting model.
- *SD Text Cond* - SD using a text CFG with prompt "a photo of {description}". See the appendix for text prompts.
- *SD Image Cond* - SD with only image CFG.
- *Extended Attention* - SD with only image CFG and extended attention [255].

For *Extended Attention* and *Grid Prior*, we inpaint in batches of 5 and 4, respectively, and for *Joint Multi-View Inpainting*, we inpaint 40 images simultaneously with $M = 8$ diffusion averaging steps. To inpaint additional batches of 40 images, we set 20 in the batch as known. This enables fitting within the memory constraints of a 16 GB GPU.

Results. Our results are shown in Tab. 4.1, where we see that *Joint Multi-View Inpainting* achieves the most favorable metrics in multi-view consistent inpainting. We note that *LaMask* has good metrics too, but this is due to failure cases described in the caption. Our results show that we have achieved some level of multi-view consistency. Our images look the most consistent in Figure 4.7 (bottom right), and the trained NeRF looks plausible. The other methods yield significant blur in the NeRF reconstruction. We provide videos on the project page showing the NeRF results for each method.

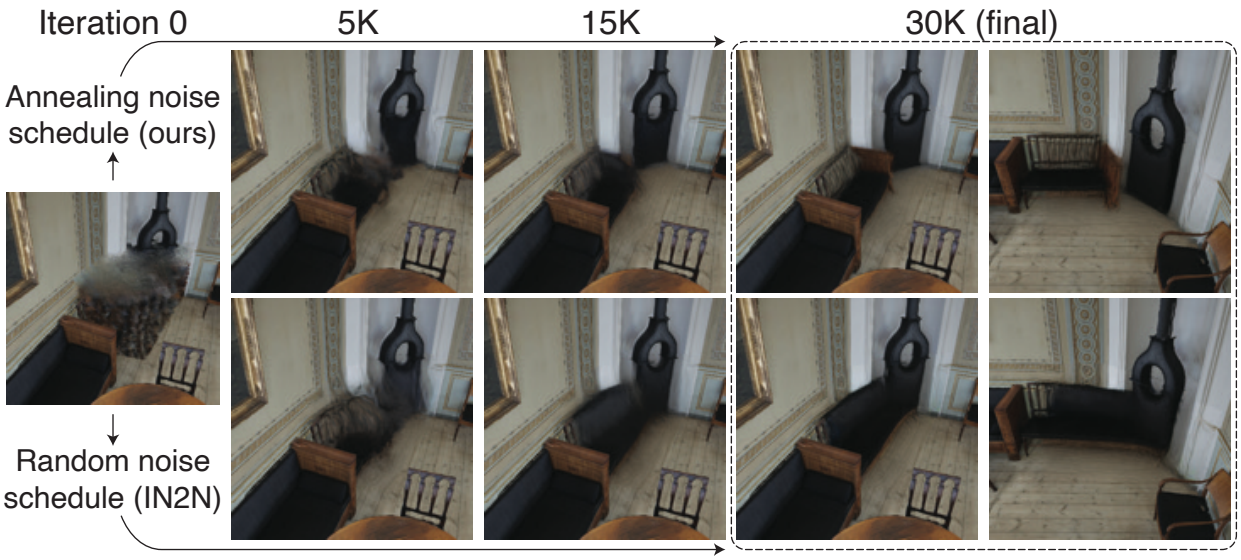

Figure 4.9: **Noise schedule.** We anneal the amount of noise we add to the NeRF renders when making a dataset update, while I-N2N chooses random noise each time. Annealing the noise produces sharper results (top) while I-N2N's noise schedule introduces significant blur (bottom).

Completing large unknown 3D regions

In this setting, our goal is to complete missing regions in 3D content. We construct a set of 10 datasets consisting of various 3D content. For some scenes e.g., the backpack from [140], we modify their provided mask to include part of the object (Fig. 4.2) to convert it to the scene completion setting. For other scenes, we simply want to fill in any missing details (e.g., parts of walls). Some of the meshes are missing vertices after multi-view stereo reconstruction, e.g., "bear" and "office". For others, we place a large 3D occluder in the scene to simulate the scene completion setting. The pink regions in Fig. 4.8 (left) shows the areas to complete. We create the datasets by rendering ∼60 novel views looking at the occluded region. Importantly, the rendered images have enough known pixels to provide context to the inpainting model that the images observe the same scene from different camera viewpoints. Our datasets have much more parallax than the forward-facing scenes of [140]. Our appendix provides details on our data, including where we obtained our 3D content, mostly from Objaverse [38, 37] and Sketchfab.

Evaluation. The evaluation is similar to Sec. 4.5 for the dataset images. However, in this case our task is to construct a scene for good novel-view synthesis, so we use all images for both training and evaluation. We compute NeRF metrics on the entire images, where we compare the final rendered images with the latest version of the inpainted region. For methods that inpaint once without DU (e.g., *LaMask*), we compare against the first and only inpaints. For DU methods, we compare against the latest round of inpaints. *Our metrics*

Figure 4.10: **Reference-based completion.** Given a reference inpaint (top row), we propagate it into a 3D NeRF (bottom row).

are against inpainted images which serves to evaluate the consistency of the scene because there is no ground-truth solution. We also report novel-view metrics, computed on a custom 10 second 30 FPS camera path novel views that moves around the scene. We also report an image quality metric MUSIQ [99] and a geometry metric. For geometry, we report the number of high-quality LoFTR [209] correspondences between 100 randomly sampled pairs of frames. More high-quality matches should correlate with better multi-view consistency and fewer extreme view-dependent effects that destroy realism. Please see the Appendix for more details.

Baselines. We implemented the following baselines:

- *Masked NeRF* - no inpainting, where we train a Nerfacto model [214] only in the known pixel locations,
- *LaMask* - Inpaint once with LaMa [211] and train with patch-based perceptual losses. This is our adaptation of SPIn-NeRF [140].
- *SD Image Cond* - Inpaint once with SD. This is similar to InpaintNeRF360 [232] but without text since we find in Sec. 4.5 that text CFG produces very inconsistent inpaints.
- *Inpaint + DU* - An adaption of IN2N [65] for our setting, which inpaints one image at a time with the SD inpainting model and our annealed noise schedule.

Results. Some qualitative results are shown in Fig. 4.8 and full videos for all 10 scenes are provided in the appendix. *LaMask* and *SD Image Cond* are both inpaint-once methods and therefore create large blurry regions in the NeRF, but between the two, *LaMask* is smoother since its deterministic inpainter [211] is less creative than SD in its outputs. LaMa [211]

tends to copy background textures into the mask region. From certain views, *Inpaint + DU* looks sharp due to inpainting individually at full resolution; however, it has geometric inconsistencies and view-dependent effects which are crisp from some angles and blurry in others. Our method looks the most consistent with plausible outputs, although its ability to create consistent high-frequency texture details may be improved. We provide quantitative results showing that our final renders are most similar with the latest round of inpaints (Tab. 4.2 left). For novel-view metrics, we obtain the most correspondences (Tab. 4.2 right).

Reference-based inpainting

In some situations it is desirable to have control over the content used to complete the scene. NeRFiller can be easily adapted to 3D inpaint with respect to a user-provided reference inpaint. To do this, we first inpaint from one view and use it to prompt our Grid Prior update method. We ensure that each grid has the one reference inpaint when passed through SD. This ensures that all U-Net predictions are influenced by the reference inpaint so that new inpaints are more likely to be consistent with the reference. Figure 4.1 and Figure 4.10 have examples.

Parameter choices

Noise schedule. It is important to anneal the amount of noise added the rendered images. We start by adding full noise (1.0) and decrease it to 0.4 over the 30K iterations of training. IN2N [65], in contrast, uses a random schedule of choosing between 0.98 and 0.02 each update. This likely works because the InstructPix2Pix model is image conditioned and geometry of the NeRF does not change much. In our case, we are changing the geometry and using their schedule leads to blurry results, shown in Figure 4.9.

Depth regularization. We find that adding depth ranking supervision [233] in *Ours* improves geometry but it hardly changes the quantitative or visual results. Our method without depth supervision is still favorable compared to the baselines. In Figure 4.11, we see that the geometry is significantly improved but the RGB NeRF renderings are nearly indistinguishable. Consequently, we use depth supervision on indoor scenes since having better geometry is favorable for downstream applications such as mesh export.

4.6 Limitations

Low resolution and blur. Our method recovers coarse geometry quite well but struggles to recover high-resolution detail in regions far way from the training cameras. We suspect this is because our *Joint Multi-View Inpainting* method downsamples images to construct the 2×2 grids. Perhaps a post-processing method to fine-tune SD [210] or a GANeRF [180] like optimization could improve the fidelity, but recovering high-frequency details remains a fundamental issue for 3D generative methods, e.g., DreamFusion [165].

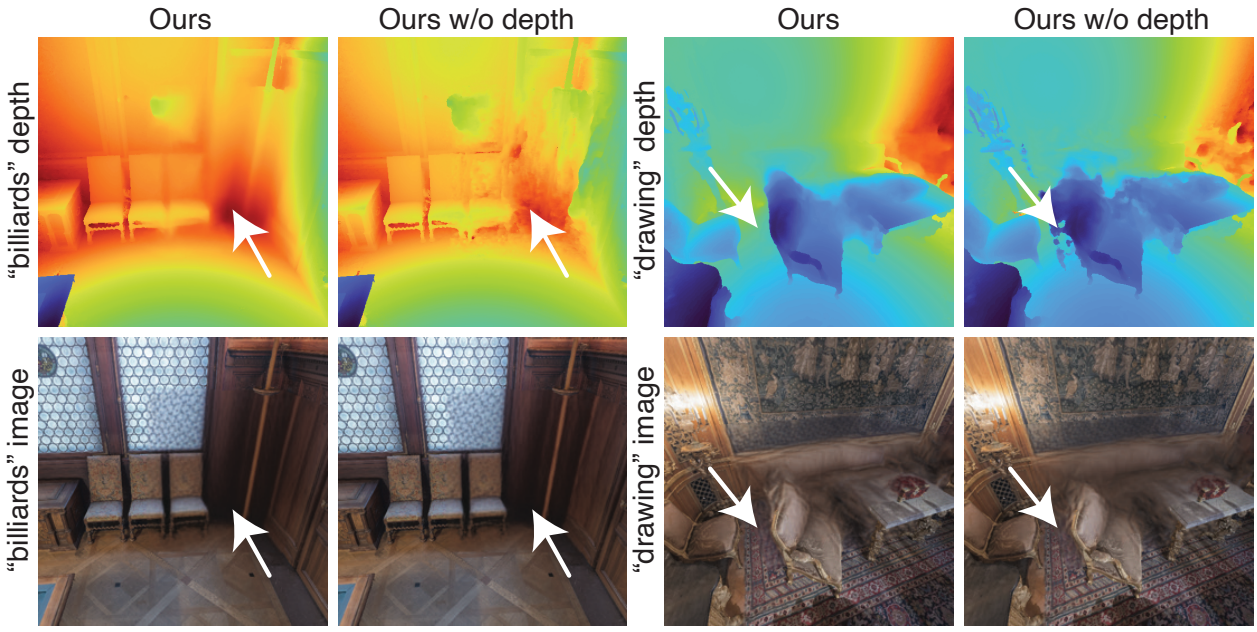

Figure 4.11: **Relative depth supervision.** Relative depth supervision cleans up geometry (top) without affecting visual quality (bottom).

Inpainting NeRF casual captures. An application of our approach would be to inpaint deleted content from Nerfbusters [246] or Bayes' Rays [61]. However, these masked-out regions are large (limiting scene context to an inpainter) and furthermore, their mask patterns cause SD [181] to fail without multiple iterations of dilation, as pointed out in Text2Room [78] and in our appendix. One could retrain SD with these mask distributions, but this is out of scope of our method which uses an *off-the-shelf* model.

4.7 Conclusion

In this paper, we propose a generative 3D inpainting method called NeRFiller, which leverages an *off-the-shelf* 2D inpainting model [181] to complete missing parts of 3D scenes and objects. We discover a unique property of these models where tiling four images into a 2×2 grid produces more consistent inpaints than inpainting them independently. We exploit this property and propose *Joint Multi-View Inpainting*, which enables inpainting many images simultaneously with more consistency by averaging noise predictions. We show how to use it in the NeRF setting by performing iterative dataset updates. We evaluate against relevant state-of-the-art baselines adapted to our problem setting on a variety of 3D captures. Our approach also enables users to specify how to fill in the missing regions. Many 3D captures are incomplete with holes, and our work presents a framework for completing these missing regions.

Acknowledgements

The authors of this paper are Ethan Weber, Aleksander Hołyński, Varun Jampani, Saurabh Saxena, Noah Snavely, Abhishek Kar, and Angjoo Kanazawa. This project is supported in part by IARPA DOI/IBC 140D0423C0035. The views and conclusions contained herein are those of the authors and do not represent the official policies or endorsements of IARPA, DOI/IBC, of the U.S. Government. We would like to thank Frederik Warburg, David McAllister, Qianqian Wang, Matthew Tancik, Grace Luo, Dave Epstein, Riley Peterlinz for discussions and technical support. We also thank Ruilong Li, Evonne Ng, Adam Rashid, Alexander Kristoffersen, Rohan Mathur, Jonathan Zakharov for proofreading drafts and providing feedback.

Chapter 5

Fillerbuster: Better scene completion

We present Fillerbuster, a method that completes unknown regions of a 3D scene by utilizing a novel large-scale multi-view latent diffusion transformer. Casual captures are often sparse and miss surrounding content behind objects or above the scene. Existing methods are not suitable for handling this challenge as they focus on making the known pixels look good with sparse-view priors, or on creating the missing sides of objects from just one or two photos. In reality, we often have hundreds of input frames and want to complete areas that are missing and unobserved from the input frames. Additionally, the images often do not have known camera parameters. Our solution is to train a generative model that can consume a large context of input frames while generating unknown target views and recovering image poses when desired. We show results where we complete partial captures on two existing datasets. We also present an uncalibrated scene completion task where our unified model predicts both poses and creates new content. Our model is the first to predict many images and poses together for scene completion. We open-source our framework for integration into popular reconstruction platforms like Nerfstudio or Gsplat.

5.1 Introduction

Photogrammetry has been around for decades [207] but only recently has become mainstream with novel-view synthesis techniques becoming high fidelity, such as NeRF [138] and Gaussian Splatting [100]. Widely used apps like Polycam [164] or Flythroughs [128] mean that everyday people can go out and easily capture content. Many such captures are done casually, which means the data is collected rather quickly and may miss large portions of the scene where the camera never looked. Sometimes, the capture is just a handful of sparse photos, which makes obtaining camera poses challenging.

Reconstructing casually captured scenes is challenging because there is missing content to complete and it is not predictable where the missing content will be from capture to capture. In contrast, the object-centric setting is much simpler as one can assume a canonical coordinate frame and sample missing views looking inward on a sphere. Instead, we highlight

Figure 5.1: **Completing casual captures.** Fillerbuster takes an incomplete casual capture which has many images (left) and conditions on these to create many consistent novel views, shown on the right with arrows. The original images and the new ones enable novel-view synthesis (right) that is much more complete compared to vanilla Gaussian Splatting trained on only the incomplete casual capture (left). The project page is `https://ethanweber.me/fillerbuster/`.

the challenges of scenes and focus on this more general setting, where the input camera poses can be incredibly diverse. Our goal is to fill in the missing information to enable an immersive view of the scene that feels complete, and where the rendered content can go beyond what is seen in training images, as illustrated in fig. 5.1. To address this problem setting, we propose Fillerbuster for recovering unknown 3D information from casually captured content. This content may be casual videos, where a user quickly scans their phone through a scene, or this may be a sparse set of photos with unknown poses, e.g. from a vacation. Given this data capture as input, our unified model can jointly complete the unobserved content and recover poses.

To improve casual captures, our key insight is to jointly model the image and camera distribution of existing casual captures by using a multi-view aware diffusion model. Our approach is made possible by the large influx of captured data being recorded and uploaded online.

We design our model to handle a large and variable number of input and output frames. This is in contrast to existing generative novel-view-synthesis (NVS) methods that are typically autoregressive, meaning the next generations are conditioned on previous generations. More specifically, our problem setting is very different from the common settings of (1) generation from text only (no images) [78, 242, 196] (2) from just one image [122, 187], or (3) from two images with the goal of interpolating between them [92, 271]. We present an overview of related work in fig. 5.2. Our model can take in many images and camera poses, e.g. 50 images, and the user can specify which content is known, which is unknown, and which should be completed. The model can also take in partially complete images, unlike the current paradigm of assuming the input images are fully intact and known [122,

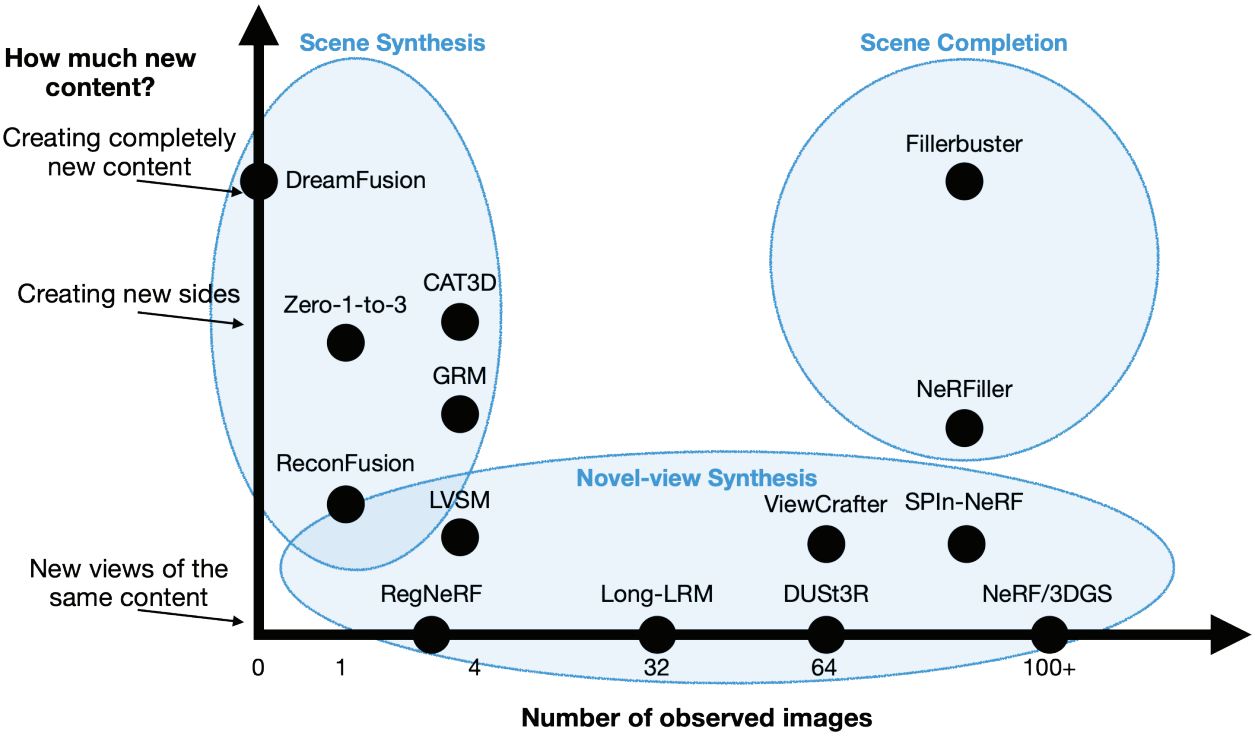

Figure 5.2: **Problem setting.** We illustrate our problem setting with respect to a non-exhaustive set of related work. Many works focus on scene synthesis (left) where one generates data from text or from a single image. Similarly many tackle novel-view synthesis (bottom) to synthesize new views of the input image content. Fewer works focus on scene completion where the task is to complete missing content in captures (top right).

57]. Concretely, we train a large-scale diffusion transformer with a flow-matching loss for inpainting in latent space [181], conditioned on known images and camera poses (represented as raymaps) with the task of recovering the missing content, as illustrated in fig. 5.3.

We demonstrate our problem setting and the usefulness of our Fillerbuster model on multiple tasks. First, we show our model can complete casual captures by hallucinating large unknown regions. Second, we introduce the task of "uncalibrated scene completion", where the goal is to recover both the image poses and completed novel views. Notably, we perform both tasks with our unified model. Third, we show the multi-view inpainting task on the NeRFiller dataset [250], where we surpass prior work in quality and consistency. Finally, we present an ablation of our modeling decisions and show our model's ability to gracefully handle different numbers of input images.

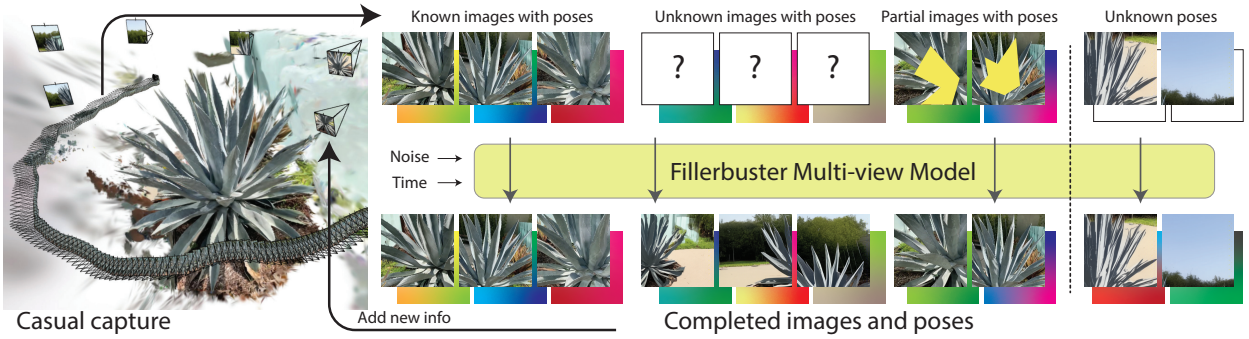

Figure 5.3: **Model overview.** Fillerbuster is trained on a large collection of multi-view images and poses (top and bottom of stacked images, respectively), which makes it useful for completing casual captures at inference time. More specifically, we are interested in four primary uses of the model: (1) conditioning on known images which have pose, (2) predicting new views where poses are provided, (3) predicting partial images where some pixels are known, or (4) recovering the camera poses when its unknown. Our model is a latent DiT trained to jointly model images and poses for any mixture of the input. In practice, our poses are 6-channel raymaps encoding ray origins and directions.

5.2 Related work

Few-view reconstruction Our problem setting is not few-view, but we highlight the differences here. These methods focus on deleting reconstruction artifacts [246, 186, 61], using sparse-view losses [152], leveraging depth and normal priors [222], or using generative models [256, 124, 96, 257], to complete sparse captures. In contrast, our goal is to complete more realistic casual captures, with more input images, and to look outside of the captured training images.

Multi-view generative models Many single-view to single new-view models exist where an input image is known and a target image is unknown [122, 187, 195, 218, 235]. A few methods have increased the input context to multiple input images but still generate just one output view [256, 92]. Even fewer methods have increased both the number of inputs and the number of outputs. CAT3D [57] uses 1 or 3 input images and generates 7 or 5 images, but never goes beyond a total sequence size of 8 (described in their supplemental material). It also remains closed-source, which limits its impact. Our model, in contrast, supports a larger and flexible number of input and output images, and will be open-sourced. We emphasize the importance of large sequence sizes in order to fit the entire casual capture in the context to make new content consistent with the observations. Some video models have been fine-tuned for camera control [69, 243, 226] or use geometry conditioning [271, 119, 143, 216], but these models generate smooth temporal videos, so can neither condition on the entire capture nor generate the many well-distributed views typical for the 3D reconstruction setting. [123, 254] clean up 3DGS artifacts in captures using diffusion priors.

LRMs conditioned on cameras Large reconstruction models (LRMs) that predict 3D have become popular to directly predict Gaussians [261, 288, 177]. Most methods assume camera poses as input, which may come from traditional methods like COLMAP [192, 154] or data-driven methods [240, 109]. LRMs are excellent at predicting pixel-aligned geometry but cannot inpaint unobserved areas of the scene. Furthermore, they rely on camera poses, which may be unknown in casual captures. We present a unified model for both tasks, such that when the camera is unknown, we can perform the "uncalibrated scene completion" task of making a camera fly-through of the scene from a set of sparse unposed photos. We model our camera pose prediction inspired by other data-driven approaches [275], but use a raymap latent space, with ray origins and directions instead of Plücker coordinates.

3D inpainting Current 3D inpainting methods such as SPIn-NeRF [140] or NeRFiller [250] rely on using 2D inpainting models within the NeRF 3D reconstruction framework to complete scenes [197]. Most methods [22, 30, 116, 29, 139] focus on the SPIn-NeRF dataset, which is forward-facing and has much less camera motion than a typical casual capture. NeRFiller has more challenging camera movement, so we consider this dataset for experiments. However, none of these methods are conditioned on camera views when inpainting. This makes it impossible for them to complete scenes with large unknown content. This is because some of the generated views will be completely unknown, and without having context of the existing scene, it is unclear how to fill in the image. In contrast, our approach is camera-pose conditioned.

5.3 Method

We first explain our model's details for jointly modeling image completion and poses (section 5.3), and then explain how to use Fillerbuster for casual scene completion with our model being helpful for 3D reconstruction (section 5.3).

Fillerbuster Model

We propose a latent diffusion transformer that denoises multiple input images and calibrated camera poses with masks indicating known and unknown regions. There are N elements in a sequence with images $I_i \in \mathbb{R}^{H \times W \times 3}$, raymaps $R_i \in \mathbb{R}^{H \times W \times 6}$ with origin and direction per pixel, valid image masks $\mathcal{M}_i^{\mathrm{I}} \in \mathbb{R}^{H \times W}$, and valid ray masks $\mathcal{M}_i^{\mathrm{R}} \in \mathbb{R}^{H \times W}$, where 1 indicates known conditioning information and 0 indicates unknown pixels. Our goal is to predict all images and raymaps given only the known information, i.e., $p(I, R \mid I \odot \mathcal{M}^{\mathrm{I}}, R \odot \mathcal{M}^{\mathrm{R}})$. We use "sequence" to refer to multiple images and cameras from the same capture, and "sequence size" for how many images are denoised together.

Model architecture The architecture is designed for latent inpainting [181], taking in any combination of known and unknown images and raymaps, and predicting the missing values. We use a DiT architecture [162] and train with the flow matching objective [117]. We train separate VAEs for images and poses encoded as raymaps, where $\mathcal{E}^{\mathrm{I}}$ denotes the

Figure 5.4: **Model samples.** Here we show generations from our model. For this setting, we provide pose input for all images. The "Masked" rows indicates which pixels are known, with yellow indicating unknown regions. This is the model conditioning signal. The "Inpainted" rows show the inpainted images after passing the entire sequence of size 16 (top rows) into the model for 24 denoising steps. The "GT" rows show the ground truth, but note that this is not necessarily the only correct solution if the newly generated pixels are unobserved according to the masks. Notice that in the top example, the generations are self-consistent but different than the GT, which is entirely plausible.

image encoder and $\mathcal{E}^{\mathrm{R}}$ denotes the raymap encoder. Both encoders compress the spatial resolution by a factor of 8× and output a d-dimensional representation. We set $d = 16$ for both encoders. Let $z_i^{\mathrm{I}} = \mathcal{E}^{\mathrm{I}}(I_i)$ and $z_i^{\mathrm{R}} = \mathcal{E}^{\mathrm{R}}(R_i)$ denote the compressed latent image and raymap, respectively. Let $\mathcal{D}$ denote a downscaling operation that reduces the spatial resolution by the same factor as the encoders. We add noise to z_i as $\tilde{z}_{i,t} = (1-t)z + t\epsilon$, then prepare the sequence as

$$\begin{aligned} s_{i,t} = \tilde{z}_{i,t}^{\mathrm{I}} \oplus \mathcal{E}^{\mathrm{I}}(I_i \odot \mathcal{M}_i^{\mathrm{I}}) \oplus \mathcal{D}(\mathcal{M}_i^{\mathrm{I}}) \oplus \\ \tilde{z}_{i,t}^{\mathrm{R}} \oplus \mathcal{E}^{\mathrm{R}}(R_i \odot \mathcal{M}_i^{\mathrm{R}}) \oplus \mathcal{D}(\mathcal{M}_i^{\mathrm{R}}), \end{aligned} \tag{5.1}$$

where $\oplus$ denotes concatenation of the noisy latents, known image and ray latents, and the masks themselves. The noisy sequence $s_t \in \mathbb{R}^{N \times H \times W \times (4d+2)}$ is patchified, and positionally embedded (described later), then passed through the transformer model $\mathcal{F}$ to predict the denoised latent images and raymaps as $\{z^{\mathrm{I}}, z^{\mathrm{R}}\} = \mathcal{F}(s)$. Our VAEs have a convolutional architecture [181] and train with KL [102], adversarial [62], and L1 reconstruction losses. Our transformer architecture is "DiT-L/2" [162] with a latent patch size of 2×2 and 24 layers of multi-head self-attention. Our model only has 650M parameters – small enough to fit on most GPUs and fast enough to use with open-source 3D reconstruction tools.

Raymap coordinate convention Raymaps comprise per-pixel ray origins and world-space unit directions. At training time, we randomly choose one camera from our sequence to be at the origin and oriented upright. We also randomly rotate and rescale the cameras for augmentation, and ensure that origins are always within the cube $[-1, 1]^3$.

Masking out regions During training, we mask out information from the images and/or the raymaps; the task is to predict the denoised sequence from partial noise. We apply masking at the pixel level before VAE encoding (eq. (5.1)) to enable precise control over which pixels are known or unknown. We use a mixed masking strategy: some images are known, some are unknown, and some are partially unknown with randomly rotated rectangles, as illustrated in fig. 5.4. We dropout image and raymap masking with a 10% chance to enable classifier-free guidance [75].

Token positional embeddings We use two forms of positional embeddings to enable varying sequence lengths to be generated at inference time. *1) 2D layout embeddings* encode the layout of the image with fixed sinusoidal embeddings. *2) Index embeddings* are more unique for our setting, where we add an unordered index descriptor to each token coming from the same image. More specifically, the full sequence s is first patchified and projected into patches p. It is then augmented with positional embeddings as $p' = \psi_{\text{2D}}(p) + \psi_{\text{Idx}}(p)$, where ψ_{2D} is sinusoidal embeddings to encode the 2D layout of each patch within the image itself [227], and ψ_{Idx} to encode which index in the sequence the patch is from. During training, $\psi_{\text{Idx}}(p)$ randomly samples a frequency for each image and then adds that value to each patch of the same image. During inference, the frequencies are chosen with uniform spacing and applied in the same way so each image has a unique identifier. This helps support generating longer sequences at inference time beyond the training lengths, which we show in section 5.4. Prior multi-view diffusion transformer models do not incorporate this, and we show that this is useful for generating longer sequences.

Training and inference details We train our model *from scratch* on a collection of datasets including ScanNet++ [267] and a corpus of Shutterstock data including 2D images and 3D asset renderings. We train our final model on 64 A100 GPUs for approximately a month. We first train at 256×256 resolution for 1M iterations, and then fine-tune for 100K iterations with resolutions varying from 64×64 to 1024×1024 with sequence lengths between 20 and 2, depending on how many images fit in GPU memory for a given resolution. See our supplement for additional details. For inference, we apply classifier-free guidance (CFG) by dropping out both image and raymap conditioning for an unconditional prediction. We use spatially varying CFG weights of 7 for the unknown regions and 1.1 for the known regions to avoid saturation artifacts since the task of copying the conditioning is much easier than predicting new information [15].

Multi-View Scene Completion

Here we explain how to use our model to complete scenes.

Variable sequence lengths Our index embedding enables changing the sequence length at inference time. We leverage this property to generate many images at the same time for

Figure 5.5: **Completing casual captures.** Here we demonstrate our ability to complete casual captures from the training splits of the Nerfbusters dataset [246]. On the left, we show the input captures and some representative images. 3DGS (Splatfacto) cannot add missing details so the capture remains incomplete. Our CAT3D baseline conditions on 3 images and generates 6 images at a time, so it cannot produce consistent content. Fillerbuster conditions on 16–40 images to generate 24 novel views, and obtains the most consistent results.

inpainting incomplete scenes, since NeRF and Gaussian splatting typically require many views to create a scene.

Multi-view inpainting for scene completion We complete scenes by generating novel views and adding them to our existing dataset, then optimizing 3DGS [100]. We avoid the need for an SDS-like optimization approach [165, 65] because our model can generate many consistent images with a large sequence size. To complete scenes with large camera

movement, we add new views to the scene that look in all directions. We first inpaint many (∼25) "anchor" frames, and then condition on these frames to generate more novel views, as in CAT3D [57]. The key difference is that we can handle much larger sequence sizes than CAT3D, which operates on at most 3 images for conditioning. Furthermore, unlike CAT3D, we can also complete scenes with partial masks. To complete these scenes, we inpaint the images themselves and update the dataset with the new pixels.

Normal regularization We find that regularizing Gaussian splat geometry towards the end of optimization can help improve results. Specifically, we apply a total-variation smoothness loss on rendered normals [52], and we also align our depth-derived surface normals with rendered normals (similar to [228] but using Gaussians instead of NeRF). This second loss is $\mathcal{L}_{\text{align}} = \|\text{sg}(N_{\text{r}}) - N_{\text{d}}\|_2^2 + \|\text{sg}(N_{\text{r}}) - N_{\text{d}}\|_2^2$, where N_{r} are rendered normals from 3D Gaussians, oriented towards the camera, and N_{d} are normals derived from rendered depth maps. We apply our normal regularizations after the initial geometry has taken form, at approximately 10K steps. The supplement has more info.

5.4 Evaluation

We first show our casual scene capture completion results on the Nerfbusters dataset, and then demonstrate the "uncalibrated scene completion" task on data captured ourselves. Next, we show results on the NeRFiller dataset, where we surpass prior work in quality and consistency. Finally, we evaluate our model design choices. Note that we choose to use 3D Gaussian splatting [100] for our reconstruction experiments rather than NeRF [138] because 3DGS is fast to train and thus gaining popularity among casual capture users.

Completing Casually Captured Scenes

Setting Here we show results for completing casually captured scenes. We choose the Nerfbusters dataset [246] for this setting because it mimics the casual captures of an inexperienced user. Our goal is to take these partial captures and to complete them – either by completing geometry or adding context to the capture. We compare the following methods:

1. *3DGS* (Splatfacto [214], which uses the gsplat library [265], with no inpainting),
2. *NeRFiller* [250] (NeRFiller inpainting, which is not suitable for this setting where the new views do not have partial masks),
3. *CAT3D-sequence-size* (ours, with CAT3D-sized conditioning—conditioning on 3 images and generating 6 images at a time, further described in the appendix), and
4. *Fillerbuster* (our complete method, conditioning on 16 views and generating 24 images at a time).

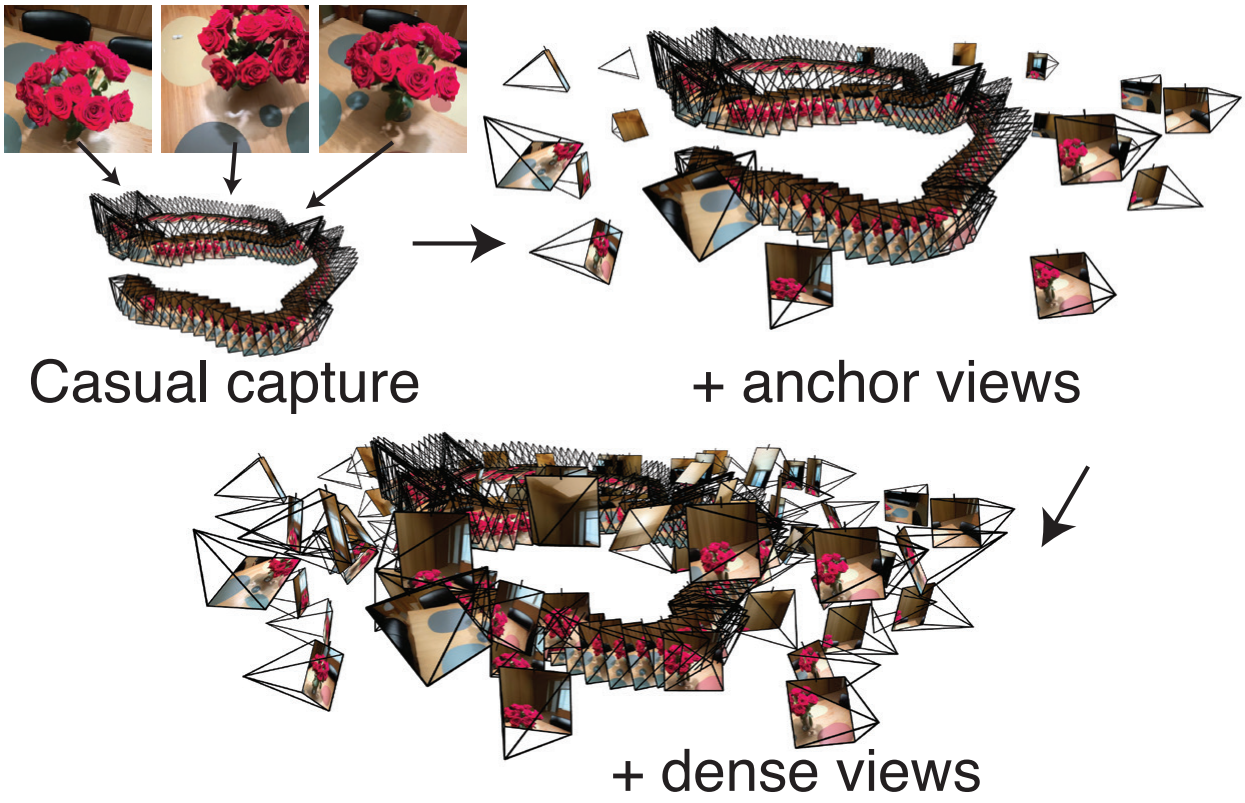

Figure 5.6: **Novel-view sampling.** We start with a casual capture (top left) and condition on 16 of the images to generate 24 anchor views simultaneously (top right). We then condition on the casual capture and anchors to densify views (bottom). We repeat the dense stage for multiple rounds to reach ∼100 novel views in total.

We perform multiple rounds of inpainting to reach ∼100 new views that are added to the scene. We show this procedure in fig. 5.6, where we sample cameras on a cylinder looking at random directions. Unfortunately, CAT3D [57] is not open-sourced, so we cannot compare with it directly.

Results We show qualitative results in fig. 5.5. We find that *3DGS* cannot add any additional detail, leading to large unknown regions when rendering novel views away from the training images. Naïvely adapting *NeRFiller* to this challenging setting fails drastically because the inpainting is not conditioned on pose and are therefore random, adding random colors to the scene. *CAT3D-sequence-size* is more consistent but introduces artifacts due to the limited context size. Our proposed method *Fillerbuster*, with large sequence sizes for conditioning and generation, of the most consistent. We design a new metric, to evaluate this task since we do not have ground-truth when hallucinating novel scene content. We are inspired by previous work that measures reconstruction error [50] or distance to epipolar

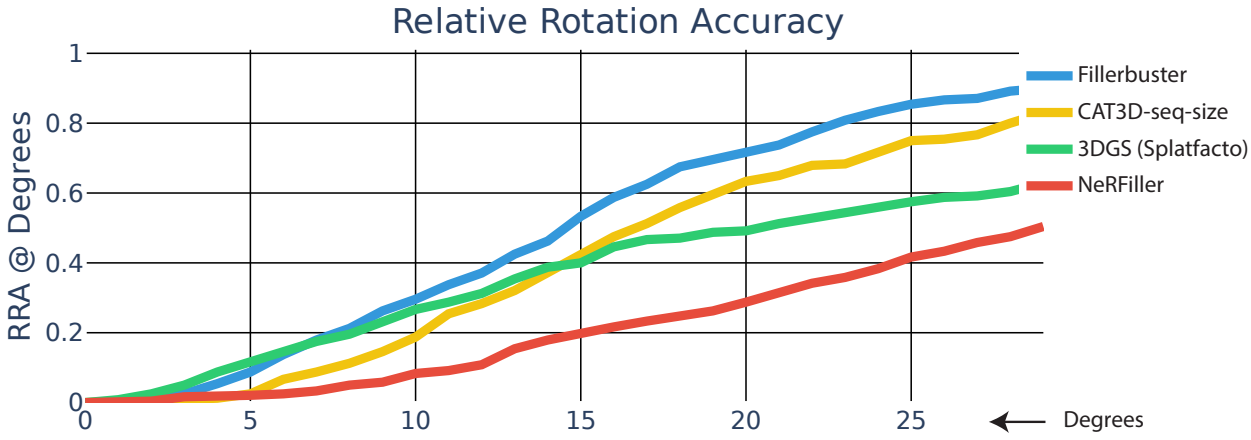

Figure 5.7: **Completing casual captures metrics.** We report relative rotation accuracy for nearby frames in a novel-view video. We use off-the-shelf correspondences [209] to estimate camera rotation and compare with the ground truth. Fillerbuster produces the most consistent videos from a pose-estimation perspective.

lines [143, 270]. Specifically, we render a novel-view camera trajectory and estimate the poses between nearby frames. We use off-the-shelf correspondences [209] and classical methods [67] to obtain a relative rotation, which we compare with the ground truth. We report relative rotation accuracy in fig. 5.7 and find the qualitative results to be consistent with camera-pose estimation accuracy.

Uncalibrated Scene Completion

Here we consider a new task of "uncalibrated scene completion", starting from a collection of 16 unposed photos. Our unified image-and-pose model supports such casual captures by predicting camera poses and then generating a fly-through of the scene, completing unknown content where missing.

Setting Given the set of images, we can denoise the raymaps conditioned only on the images. We use joint denoising tiling with a window size of 8 images and average 8 times per denoising step. This is similar to MultiDiffusion [10] or NeRFiller's joint denoising [250] (see appendix). Then, we solve for the pinhole camera parameters that match backprojected rays to the denoised rays, taking only 5 seconds to converge for 16 images. Next, we condition on our predicted rays to generate novel views to complete the scenes. We create a camera path by fitting a 2D ellipse to our posed images and point cameras inward. Here we use a sequence size of 48: 16 input images with generated poses, plus 32 generated images with specified poses, but note that this decision is flexible.

Results Our joint modeling of poses and images is convenient because we do not rely on external structure-from-motion; instead, our unified model can handle both tasks gracefully.

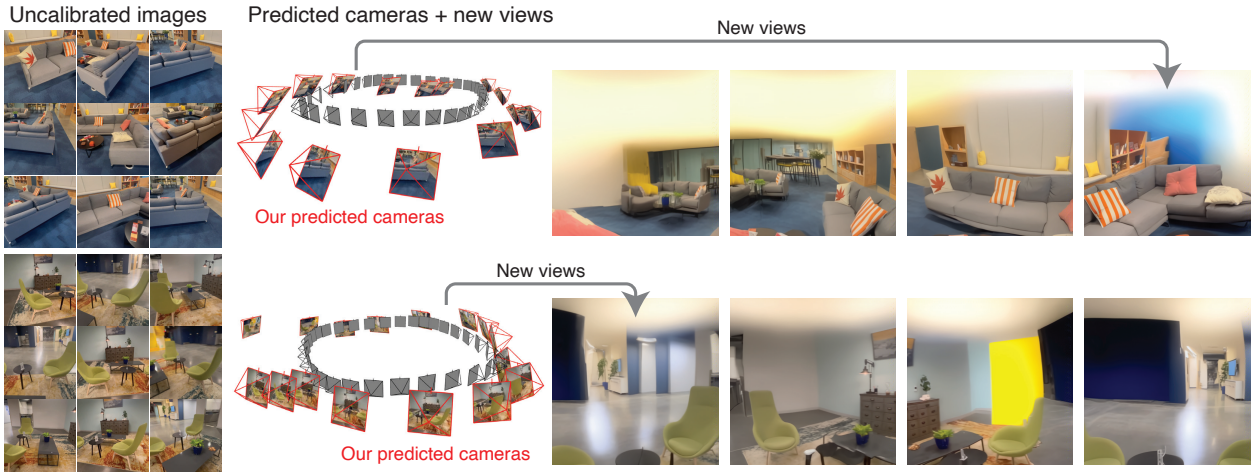

Figure 5.8: **Uncalibrated scene completion.** We capture some scenes with an iPhone 14 Pro and run our framework. We start from 16 uncalibrated and unposed images (left), and we use our model to both predict camera pose (middle) and generate completed views (right). We show our predicted cameras in red compared and unknown views we will sample in black. Our cameras are plausible and useful for conditioning on to generate new views. We show just 4 views here and the full videos in the supplement.

Table 5.1: **Completing masked 3D regions.** On the NeRFiller dataset [250], we report novel-view synthesis metrics where we compare the rendered images with the inpainted images. In parentheses, we report numbers without using our normal regularizations. No normal regularization lets the network cheat to explain inconsistencies, leading to slightly improved but misleading metrics. Overall, we find Fillerbuster is much more consistent than NeRFiller.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
NeRFiller	25.57 (25.94)	0.89 (0.88)	0.182 (0.194)
Fillerbuster	**29.60** (30.65)	**0.92** (0.93)	**0.096** (0.069)

We show qualitative results of our video poses in fig. 5.8. Please see our supplement for video results. We do not compare with COLMAP or SfM methods because they do not complete scenes, meaning they don't infill the missing areas outside the provided input images context.

Completing Masked 3D Regions

Figure 5.9 shows results where we inpaint scenes from the NeRFiller dataset and compare against the NeRFiller method [250]. For both NeRFiller and Fillerbuster, we inpaint 32 equally-spaced training images and then train 3D Gaussian splatting [100] with our normal

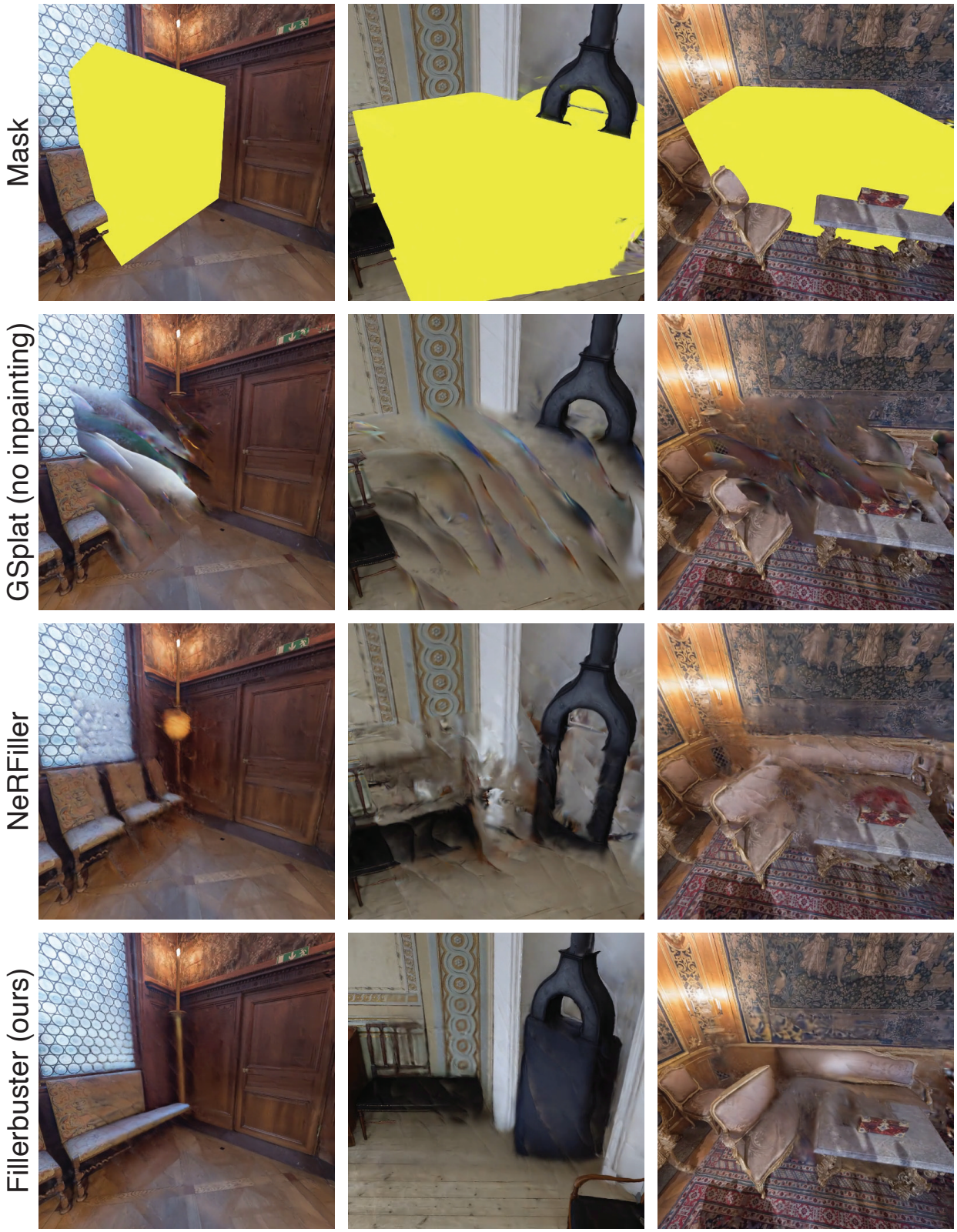

Figure 5.9: **NeRFiller dataset novel-views.** We complete NeRFiller scenes [250] with higher quality and control than their method.

Table 5.2: **Model design ablations.** We evaluate our model on posed images from the Nerfstudio Dataset [214]. We show a Fillerbuster prediction above the table, where we compare the generation vs. the ground truth for reconstruction metrics (PSNR/SSIM/LPIPS). For hallucination metrics, we report the conditional validation loss (VAL) as done by [45]. Notably, we focus on image generation rather than pose prediction but find that not predicting pose ("no-pose-pred") leads to worse results. See section 5.4 for detailed descriptions.

Method	Iters	8-views				16-views				32-views			
		PSNR ↑	SSIM ↑	LPIPS ↓	VAL ↓	PSNR ↑	SSIM ↑	LPIPS ↓	VAL ↓	PSNR ↑	SSIM ↑	LPIPS ↓	VAL ↓
no-index-emb	100K	11.10	0.431	0.456	0.2394	14.06	0.450	0.400	0.2417	12.38	0.467	0.422	0.2438
fixed-index	100K	10.46	0.353	0.520	0.2517	12.14	0.390	0.491	0.2546	12.52	0.416	0.448	0.2556
no-poses	100K	11.63	0.413	0.426	0.2386	14.73	0.461	0.384	0.2411	13.39	0.476	0.389	0.2431
random-poses	100K	11.82	0.431	**0.415**	0.2384	**16.18**	**0.487**	0.333	0.2409	**14.27**	0.483	0.367	0.2430
Fillerbuster	100K	**11.97**	**0.435**	**0.415**	**0.2383**	15.81	0.481	**0.329**	**0.2407**	14.21	**0.486**	**0.366**	**0.2426**
Fillerbuster	1M	12.77	0.442	0.381	0.2365	17.20	0.485	0.281	0.2388	14.13	0.498	0.352	0.2396

regularization. Notably, unlike in NeRFiller, we do not use depth supervision or iterative dataset updates. We instead inpaint once at the start of training to directly assess multi-view inpainting quality, regardless of any SDS-style optimizations that encourage consistency. We also report reconstruction metrics in table 5.1 by comparing our 32 inpainted images with the final renderings from the same 32 viewpoints. Our method is more consistent than NeRFiller, with and without our normal regularization.

Model Design Ablations

We evaluate our model choices with the Nerfstudio dataset [214] because it consists of well-captured static scenes that look in all directions. In table 5.2, we report novel-view synthesis metrics for 256×256 resolution images for varying sequence lengths. For each scene, we randomly sample a sequence of size N image crops of this resolution. We condition on $N/4$ full crops and $N/4$ partial crops, and generate all the missing information (see fig. 5.4 for examples). We repeat this procedure 50 times for sequences of length $N \in \{8, 16, 32\}$, and report averaged metrics. Each model is trained from scratch for 100K iterations with sequences of $N = 8$ images. Inspired by [45], we also report validation losses ("VAL") on these samples for 20 equally-spaced time steps. This metric captures generation quality unlike the other metrics that evaluate reconstruction.

Table 5.2 compares the following ablations (see fig. 5.10 for visual examples): *"no-index-emb"* does not use index embeddings and instead relies on the raymaps to understand the token relationships. This makes the task harder and the model performs worse. *"fixed-index"* uses a fixed number of index embeddings, preventing it from generalizing to more than 8 images. To go beyond 8 images, more index embeddings are introduced, which this model variant cannot handle. Notice the low PSNR of this setting for 16-views. *"no-poses"* does not denoise raymaps, and interestingly, we find that when it does not learn to predict camera pose, the model performs worse at image generation, indicating that image and pose

Figure 5.10: **Qualitative results for model ablations**. We provide pose for all images and perform completion in the unknown regions (yellow). Without index embeddings, the model fails to reason about which image the tokens are coming from, so the results are patchy and blurry. Without training for pose prediction, the generations are worse than training for pose prediction (Fillerbuster (100K)). The last rows show our final model and GT for reference.

predictions are complementary tasks. Finally, *"random-poses"* randomizes the poses instead of forcing them to be upright with one camera at the origin. Our final model is trained for much longer and is shown at the bottom of the table, obtaining the best "VAL" results. We also note that the metrics vary for 8, 16, and 32 views because each section in the table has different image subsets, making them non-comparable. The amount of known and unknown info also changes based on number of views, as described earlier.

5.5 Conclusion

Many 3D casual captures are missing content because the camera does not look everywhere. To recover these missing details, we present Fillerbuster, a large-scale multi-view diffusion model, to complete missing regions or recover camera poses when they are not available. We show our model is useful at completing casual captures, and we introduce an "uncalibrated scene completion" task where we generate novel missing content from unposed images from our own mobile captures. We also outperform NeRFiller on its setting, where partial views are known and there are masks to complete. Lastly, we present important model design decisions to enable large sequence conditioning and generation.

The area of scene completion is incredibly exciting and there are many avenues to explore beyond what is presented here. For example, choosing camera paths can be challenging because cameras should not be sampled inside of objects or behind walls. A method that predicts where to sample next could be valuable. We also note that our model's generations become worse when the generated cameras are very far away from the conditioning views, e.g. when we step back too far from the input views. Incorporating more diverse training data, or specifically rendering distant viewpoints in simulations, may be useful. We proposed an initial step towards the challenging problem of casual capture scene completion. We expect the results will further improve with a larger model, bigger compute budget, and more diverse training data. We train our diffusion model from scratch due to legal constraints, but we expect starting from a pre-trained image or video diffusion model would lead to higher quality results with the same compute budget. We will open-source our model weights, training and processing code, hoping it aids in future 3D reconstruction and scene completion efforts.

Acknowledgements

The authors of this paper are Ethan Weber, Norman Müller, Yash Kant, Vasu Agrawal, Michael Zollhöfer, Angjoo Kanazawa, and Christian Richardt. We would like to thank Timur Bagautdinov, Jin Kyu Kim, Julieta Martinez, Su Zhaoen, Rawal Khirodkar, Nir Sopher, Nicholas Dahm, Alexander Richard, Bob Hansen, Stanislav Pidhorskyi, Tomas Simon, David McAllister, Justin Kerr, Frederik Warburg, Riley Peterlinz, Evonne Ng, Aleksander Holynski, Artem Sevastopolsky, Tobias Kirschstein, Chen Guo, Nikhil Keetha, Ayush Tewari,

Changil Kim, Lorenzo Porzi, Corinne Stucker, Katja Schwarz, and Julian Straub for helpful discussions, technical support, and/or sharing relevant knowledge.

Chapter 6

Sitcoms3D: Reconstructing TV shows

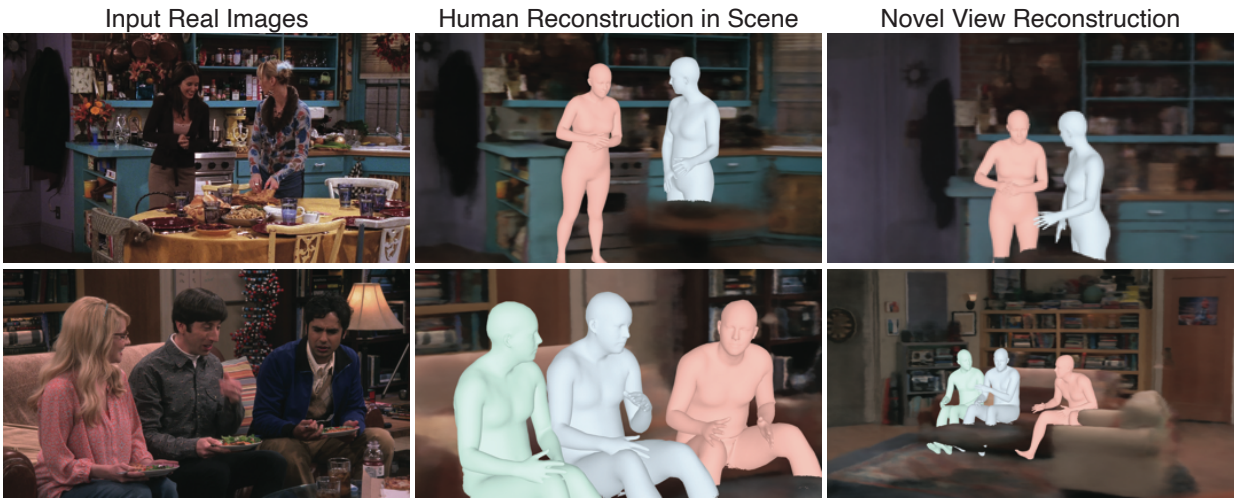

Figure 6.1: **Reconstruction of humans in TV show environments.** Given images across the whole season of a TV show, we present an approach that recovers the 3D scene context, which enables accurate estimation of every actor's 3D pose and location. We show the input (left), the mesh reconstructions of the actors in the camera view (center) and in a novel view (right). Human meshes are visualized against the reconstructed scene, which is represented by a Neural Radiance Field (NeRF). To appreciate the correct 3D localization of people, notice the position in the novel view and the occlusions. Readers are encouraged to watch video results in the project page: `http://ethanweber.me/sitcoms3D/`.

TV shows depict a wide variety of human behaviors and have been studied extensively for their potential to be a rich source of data for many applications. However, the majority of the existing work focuses on 2D recognition tasks. In this paper, we make the observation that there is a certain persistence in TV shows, i.e., repetition of the environments and the humans, which makes possible the 3D reconstruction of this content. Building on this

insight, we propose an automatic approach that operates on an entire season of a TV show and aggregates information in 3D; we build a 3D model of the environment, compute camera information, static 3D scene structure and body scale information. Then, we demonstrate how this information acts as *rich 3D context* that can guide and improve the recovery of 3D human pose and position in these environments. Moreover, we show that reasoning about humans and their environment in 3D enables a broad range of downstream applications: re-identification, gaze estimation, cinematography and image editing. We apply our approach on environments from seven iconic TV shows and perform an extensive evaluation of the proposed system.

6.1 Introduction

Remember that time when you binge-watched an entire season of your favorite TV show, e.g., "Friends", over a weekend? After that experience, you would know the layout of the rooms, the locations of the furniture, and even the relative height of the characters as they interact closely on screen. As a result, for any frame, you could tell where the room is viewed from, where the characters are situated, and how they relate to the rest of the scene, even the parts of the scene outside the frame. Essentially, as viewers, we aggregate all the visual information into a dynamic 3D world where the new observations are aligned to.

In this paper, we propose a method that can similarly aggregate 3D information over video collections and use it to perceive accurate 3D human pose and location of the actors. Although reconstruction of dynamic scenes is challenging from a single video clip, our insight is that in the context of TV shows, across many episodes, there are many video clips that *depict the same scene and people many times.* The repeated observations provide a strong multi-view signal of the underlying scene, enabling reconstruction of the camera and the dense structure. These serve as context to accurately recover the 3D pose and location of the people in the 3D environment. A representative result is shown in Figure 6.1. Although we demonstrate our method & results on TV shows, our insight is also applicable to other domains with repetition in the environment and the people, e.g., sports [79, 174, 285], late night shows [60, 149] and movies [160].

We operationalize our insight by focusing on an entire season from TV shows and collecting the sequences that correspond to a specific environment. These sequences are organized in shots [6], which are typically captured by different cameras. To collect a diverse set of images, we sample frames at the shot boundaries (cuts between cameras). This ensures a wide variety of viewpoints, while avoiding redundancy, making it practical to apply a structure-from-motion pipeline [192] to estimate the intrinsic and extrinsic camera parameters (calibration). Then we use a neural radiance field (NeRF-W [132]) to disentangle the static and transient components and obtain a dense 3D reconstruction (Figure 6.2, first block).

The 3D scene reconstruction offers rich 3D context - cameras and scene structure - enabling an in-depth study of humans (Figure 6.2, second block). First, for frames on the

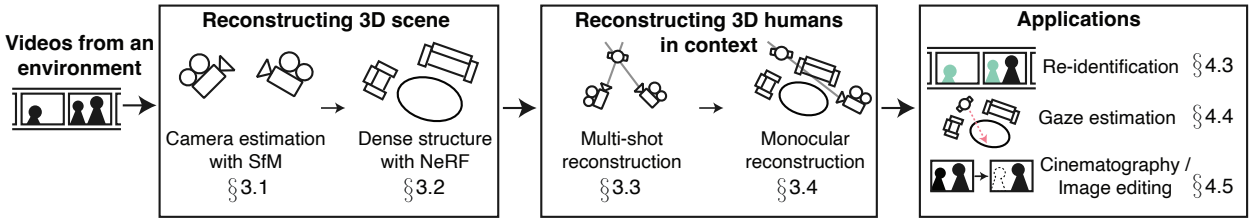

Figure 6.2: **Overview of our workflow.** First, we use a collection of videos from a TV show environment and reconstruct the 3D scene (cameras and dense structure). We then use this information to recover accurate 3D pose and location of people over shot boundaries and on monocular frames. The recovered 3D information is immediately useful for various downstream applications.

shot boundaries, the viewpoints from the two different shots act as effective multi-view (or multi-shot) information for human reconstruction [160]. We use the calibrated cameras and propose a multi-shot human reconstruction method, which jointly solves for body pose, body shape, identity and location. In this **calibrated multi-shot** method, camera information enables triangulation of people, which removes ambiguity and provides significant improvement upon the equivalent uncalibrated baseline [160]. Next, human reconstructions on the shot boundary inform us of the scale of each person relative to the scene. This is additional 3D context that is complementary to the cameras and scene structure. Since most frames are not on shot boundaries, we also formulate a monocular human reconstruction method that is explicitly guided by the extracted 3D context (camera, structure, body scale). The successful integration of the 3D context in our **contextual monocular** method leads to improvements over the state-of-the-art monocular baselines.

Our proposed "3Dification" of TV shows opens the door to many immediate applications (Figure 6.2, third block). First, our human reconstruction on the shot boundaries associates person detections, by incorporating geometric and anthropometric constraints. We show that this form of **re-identification** consistently outperforms traditional image-based baselines [53, 83, 84]. In parallel, from our reconstructed humans we can extract reliable **gaze information**, which can outperform specialized gaze estimation [172]. Moreover, our results provide estimates of the camera-to-person distance, which is relevant for **cinematography** applications [189, 190]. Finally, we illustrate the potential use in **image editing** applications, like object insertion or human deletion.

In summary, our contributions can be summarized as follows:

- We identify the significant amount of 3D context (cameras, structure and body shape) in domains with repetition in the environment and the people, e.g., TV shows, and propose a method to aggregate it from video sequences.
- We propose a formulation that integrates this context in 3D human estimation methods, which improves human reconstruction.

- We demonstrate how the aggregated 3D information can help a wide variety of downstream tasks: re-ID, gaze estimation, cinematography, image editing.
- We perform extensive qualitative and quantitative evaluation to validate the quality of our recovered 3D results.

6.2 Related work

Perceiving TV shows

The computer vision community has a long history of works on perceiving TV shows/movies. One of the most common tasks in this setting is studying the show characters with emphasis in face/character identification [5, 47, 148, 157, 201, 217], where different cues have also been explored, e.g., body, voice or gaze [18, 46, 130, 131]. TV show data has been used extensively to study human behavior. Ferrari et al. study 2D pose estimation [49] and perform pose-based analysis [48]. Patron et al. [159] and Hoai et al. [76] focus on human interactions, while Recasens et al. [172] and Marín-Jiménez et al. [130] use this data to study gaze. Vondrick et al. [230] use sequences from TV shows to learn activity forecasting, while Wang et al. [241] leverage it for affordance learning. Despite this attention, all the above methods reason in 2D, with only a few exceptions. Everingham and Zisserman [47] use a 3D head model for re-identification. Here, we demonstrate how 3D location information can significantly simplify the re-ID problem. Pavlakos et al. [160] reconstruct humans from videos with multiple shots. However, they operate without camera calibration, while we show the importance of recovering reliable cameras.

Scene reconstruction

Reconstruction of 3D scenes is a well studied problem, e.g., [3, 82, 192, 193], however, most methods assume static scenes. Related work focuses on dynamic reconstruction [8, 147], but requires capture from multiple wide-baseline synchronized cameras. Luo et al. [129] and Kopf et al. [108] present pipelines for recovering depth in monocular videos that include humans, but they assume that the underlying scene is static. View synthesis approaches like NeRF [138] and follow-ups [153, 264] can be used to solve multi-view stereo, however these also assume static scenes. Other extensions of NeRF focus on reconstructing 3D motion in the scene [55, 114, 156], but are often limited when handling changes in appearances and transient objects. NeRF in the wild [132] is the most relevant approach for the type of data we use (Figure 6.3), since it can deal with appearance and transient changes. In this paper, we find that when our data is properly curated we can use NeRF-W to recover dense 3D structure. We then show that this structure can be used to guide consistent 3D human reconstruction.

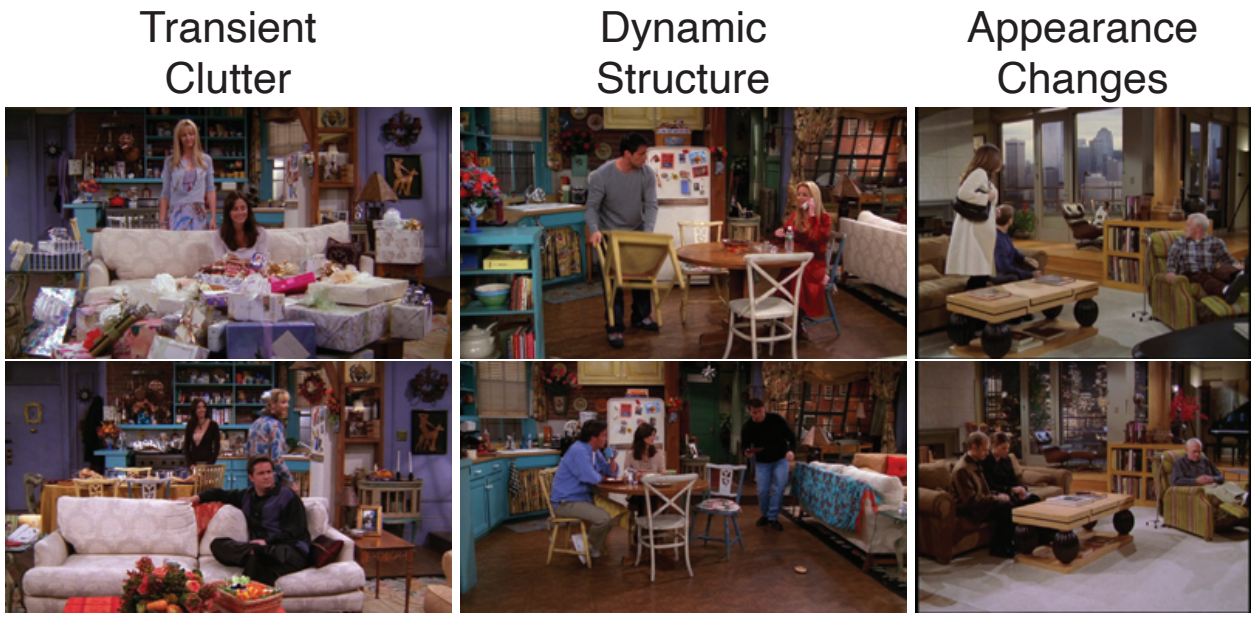

Figure 6.3: **Reconstruction challenges** of TV shows include transient and dynamic objects as well as appearance changes.

Humans in 3D scenes

Most works that reconstruct humans in context with the scene assume a static, pre-captured 3D environment. Savva et al. [191] is one of the first works that explore 3D human-scene interactions from RGB-D video, while Hassan et al. [68] study the recovery of 3D humans in context with their environment from monocular images. Many works incorporate environmental constraints for motion estimation from videos [176, 175, 198, 199, 258, 273, 279], by assuming known floor or contact points. Recently, Guzon et al. [64] proposed a system for localizing a person in a known environment and estimating their 3D pose. Again, the environment is reconstructed *a priori* and the approach also requires an egocentric sensor and IMUs for pose estimation. Liu et al. [121] propose a method that reconstructs the scene and the people together using egocentric video captured in static outdoor scenes. In this work, we reconstruct structure from much more challenging dynamic scenes, by aggregating 3D information over video content.

Some works [252, 276] have studied human reconstruction from single images, while also recovering aspects of the environment. PHOSA [276] recovers humans interacting with objects from in-the-wild images, and is followed by [252, 260] in other settings. While they focus on visible human-object interactions, we consider cases where the scene might not be fully visible. Knowing camera parameters is an integral part of scene perception. SPEC [105] regresses camera parameters from a single image. In contrast, we can recover more reliable context for cameras by leveraging the whole collection of images from a TV

show environment.

6.3 Technical approach

For the following discussion, we use the term *environment* to refer to a location, i.e., a room, kitchen, cafe, etc., that appears often in a TV show. Figure 6.4 visualizes the panoramic view of the environments we reconstruct in this paper. We use the term *shot* for an uninterrupted sequence captured by a camera. Shots are organized in *scenes*, which are typically captured in the same environment. Multiple scenes comprise an *episode* and multiple episodes are organized into a *season*. In this work we collect videos across the whole season of a TV show.

Camera estimation

For the first step of our workflow, we need to register the cameras in a common coordinate frame (i.e., computing intrinsics and extrinsics) for each environment. This amounts to hundreds of thousands of frames across the season. To keep the number of frames at a practical scale for Structure-from-Motion (SfM) pipelines, we sample frames at shot boundaries, which are automatically detected [84]. This helps to increase the variety of viewpoints - we only use two frames per shot, and inter-shot variety is typically larger than intra-shot variety.

On this reduced set of frames, we use DISK [223] to find correspondences. Since our data includes dynamic actors, we run Mask R-CNN [70] to detect human masks, and we reject correspondences on these regions. We use COLMAP [192] on the remaining feature matches and estimate the sparse 3D reconstruction and camera registration. We use a simple pinhole camera model, and allow each camera to have different focal length. For each frame t we get estimates of camera intrinsics $K_t \in \mathbb{R}^{3\times3}$ and extrinsics $R_t^{\mathrm{CW}} \in \mathbb{R}^{3\times3}, T_t^{\mathrm{CW}} \in \mathbb{R}^3$, where CW denotes camera to world transformation. This sparse reconstruction is used to register other frames (non shot-boundary images). Since we do not have access to 3D ground truth for TV show environments, the quality of our cameras is evaluated implicitly by the effect it has on the human reconstruction (see also Sup. Mat.).

Dense structure

Besides the camera registration returned from SfM, we also estimate the dense structure of the environment to help with human position estimation. Traditional dense reconstruction methods assume static scenes, but these assumptions are not satisfied in TV show environments which contain many images of extreme diversity (Figure 6.3). Instead, we use a NeRF-W network [132] for dense structure estimation. NeRF-W extends NeRF [138], to account for varying appearances and transient occluders. For efficiency, instead of training NeRF-W with all images, we use an automatic selection method to maximize viewpoint variety. We cluster the images based on camera location and viewing direction. For each cluster

Figure 6.4: **Panoramic views of the reconstructed TV show environments.** We obtain and render the static structure using NeRF-W [132]. The environments represent seven TV shows: "The Big Bang Theory", "Frasier", "Everybody Loves Raymond", "Friends", "Two And A Half Men", "Seinfeld" and "How I Met Your Mother".

we select the image with least percent of Mask R-CNN human pixels to use for training (i.e., maximum number of scene rays). After training, NeRF-W returns a volumetric 3D representation of the static structure of the scene.

Calibrated multi-shot human reconstruction

In movies and TV shows, scenes are filmed in consecutive shots. The shot changes within a scene correspond to consecutive time frames seen by different viewpoints. This serves as *effective multi-view* information, providing signal to recover the 3D location and pose of the actors [160]. However, doing so requires knowledge of the identity of the actors across the shot changes. Prior work utilizes a pre-trained recognition-based re-ID model to establish these correspondences, but this is not always reliable, for example when only the back of the character is visible. We make an observation that when camera information is available, the association can be solved jointly with the 3D human pose, shape, and location. We refer to this approach as *calibrated multi-shot optimization*.

Let us assume there are M actors in frame t, N actors in frame $t+1$, and a shot change happens from frame t to $t+1$. We need to solve a matching problem to associate the two sets of actors. We propose to use the objective of SMPLify fitting [16] to model the cost for this matching.

Formally, let us consider a detection of a person at time instance t, with detected 2D keypoints $J_{est,t}$ [23]. We denote with θ_t the pose parameters and with β_t the shape parameters

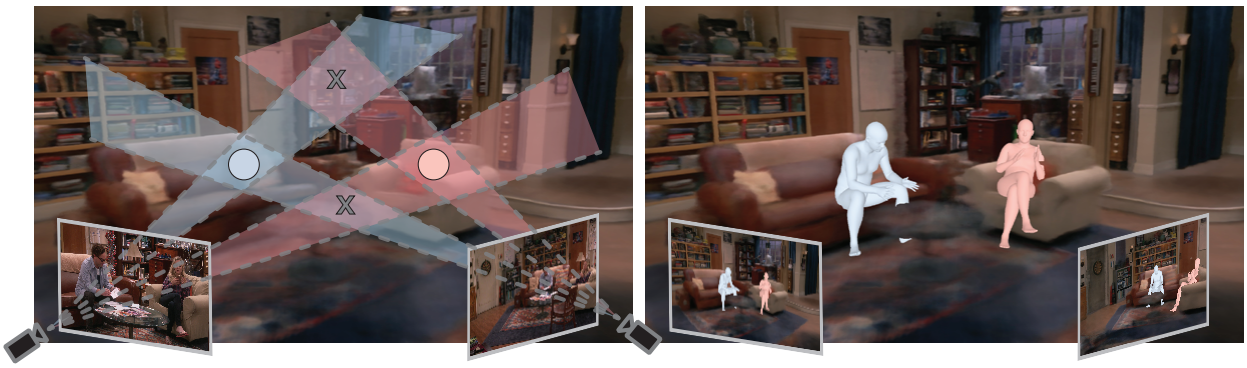

Figure 6.5: **Calibrated cameras for scale estimation and identity association.** Given calibrated cameras, we can use frames at a shot change to solve for the actors' pose, location, relative scale and association. The four overlapping regions (left) indicate possible locations triangulated by the cameras. Circles indicate correct matches after Hungarian matching. Reconstructed humans are visualized in a NeRF (right).

of the person in the SMPL format [126]. We use J_t for the joints and T_t^{C} for the translation of the body in the *camera frame*. Moreover, from SfM, we have access to the transformations $R_t^{\mathrm{CW}}, T_t^{\mathrm{CW}}$ from the camera frame to the world frame at time t. Given all of the above, we minimize the objective function with respect to $\{\theta_t, \theta_{t+1}, \beta_t, \beta_{t+1}, T_t^{\mathrm{C}}, T_{t+1}^{\mathrm{C}}\}$:

$$E = \underbrace{E_{J_t} + E_{J_{t+1}}}_{\text{2D reprojection}} + \underbrace{E_{\text{priors}_t} + E_{\text{priors}_{t+1}}}_{\text{anthropometric constraints}} + \underbrace{E_{\text{glob}_{t,t+1}}}_{\text{3D consistency}} \tag{6.1}$$

Here, $E_{J_t} = E_{J_t}(\beta_t, \theta_t, K_t, J_{est,t})$ is the joints reprojection term and E_{priors} are anthropometric priors similar to [16]. The key constraint is multi-shot consistency, which encourages the estimated bodies to be similar in the global frame:

$$E_{\text{glob}_{t,t+1}} = \|(R_t^{\mathrm{CW}} J_t^{\mathrm{C}} + T_t^{\mathrm{CW}}) - (R_{t+1}^{\mathrm{CW}} J_{t+1}^{\mathrm{C}} + T_{t+1}^{\mathrm{CW}})\|^2. \tag{6.2}$$

In contrast to prior work, we do not need to solve for the camera as we have access to reliable extrinsics and intrinsics (prior works [91, 160, 276] use a heuristic for focal lengths). This leads to more accurate human placement and constraints that allow for solving associations. Using this fitting cost E, we solve association by Hungarian matching. See Figure 6.5 for an illustration of this optimization.

Contextual monocular human reconstruction

Although shot changes provide effective multi-view information for free, the majority of the frames in the video only have monocular observations. Monocular human reconstruction is challenging, particularly so for TV shows with many close-up shots; however, in our case,

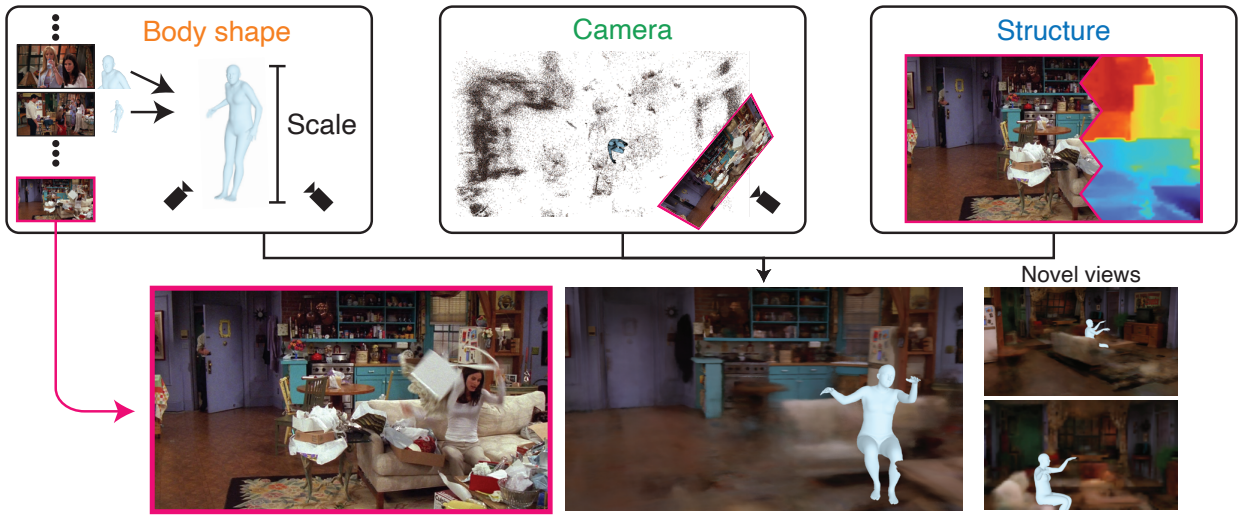

Figure 6.6: **Contextual monocular human reconstruction.** For an input frame, we can leverage (a) the **body shape** (scale) of the person from a neighboring shot change, (b) the **camera** registration, and (c) the static **structure** of the environment. This enables monocular reconstruction of the person in context with their environment.

we can capitalize on the contextual information we have recovered. In this subsection, we explain how we can make use of this 3D context in an effective way. We demonstrate this using a single-frame optimization approach, SMPLify [16], but other methods could also benefit from our context, e.g., we show representative results for HuMoR [176] in the Sup. Mat.

A high-level overview of this step is presented in Figure 6.6. First, given the sparse reconstruction of the environment, we can register the **camera** for a new frame. This gives us both extrinsics $R_t^{\mathbf{CW}}$, $T_t^{\mathbf{CW}}$ and intrinsics K_t for the camera via solving PnP with COLMAP [192]. We leverage these parameters for accurate projection. Moreover, we can employ the structure captured by our NeRF-W network. In general, it is not trivial to extract the structure from NeRF [153, 264]. The native representation used by NeRF is in the form of densities for each point. Here, we propose to use this density as a proxy for occupancy of the 3D space. With this in mind, we formulate an objective to discourage the human body vertices V from occupying areas with high density values:

$$E_{\textbf{structure}} = \rho\Big(\sum_{v \in V} \tilde{\sigma}(v)\Big), \tag{6.3}$$

where $\tilde{\sigma}$ samples values from the density field σ using trilinear interpolation, while ρ is the Geman-McClure robust error function [58]. Finally, we leverage the shape parameters $\hat{\beta}$ that capture the relative scale of the person with respect to the environment, and are recovered

from the nearest shot change with the calibrated multi-shot reconstruction. This value can be used explicitly in the optimization to resolve the scale ambiguity.

Eventually, our monocular fitting objective minimizes:

$$E_J(\beta = \hat{\beta}, \theta, K = \hat{K}, J_{est}) + E_{\text{priors}} + E_{\textbf{structure}}, \tag{6.4}$$

with respect to θ_t, T_t^{C}, where we employ the **camera** information and the **body shape** parameters of the person during the fitting, while also discouraging the body mesh from penetrating the static **structure** of the scene.

Applications

An important argument in favor of 3D reconstruction for people in TV show environments is that it can simplify many reasoning tasks in this domain. For example, the calibrated multi-shot optimization explicitly reasons about the identity of the detected humans, as part of the Hungarian matching. This enables reliable *re-identification* in the challenging case of shot changes where the viewpoint can change significantly (Section 6.4). Moreover, one can extract *gaze information* from our 3D humans by considering the 3D pose of the face/head. With knowledge of camera pose, we can easily estimate the gaze direction in the global space, and thus compute gaze targets by intersecting these rays with the 3D environment. We analyze various aspects of our data which could be useful for *cinematography* applications, and highlight the potential of *image editing* using our results (Section 6.4).

6.4 Experiments

In this section, we present the quantitative and qualitative evaluation of our approach. We use seven popular TV shows (Figure 6.4) and one season from each. We follow the procedure described in Section 6.3 to collect the images we use. Each environment has 1k-5k frames from shot changes. For evaluation, we select per TV show a set of 50 person identities present on these shot changes. We use these frames as a test set to evaluate our method qualitatively with a crowd-sourced perceptual evaluation on AMT and curate it with the information we require for quantitative evaluation, i.e., human-human associations, body keypoints, top-down location of the pelvis in the scene and gaze target across the shot change.

Calibrated multi-shot human reconstruction

For a proof of concept, we first evaluate our proposed calibrated multi-shot optimization in a controlled setting, with the Human3.6M dataset [85], where we have accurate 3D ground truth for pose. Since our focus is on the effect of having access to camera parameters, we compare with the equivalent uncalibrated baseline, which is similar to [160]. The results are presented in Table 6.1. The significant improvement when having access to camera information further motivates the importance of our calibrated multi-shot algorithm.

Figure 6.7: **Calibrated multi-shot and re-ID results.** Using input shot changes (left), we perform our calibrated multi-shot optimization which jointly solves for pose, shape, location and association (middle). Note that identity, illustrated with colors, is not available a priori, but is estimated jointly with the 3D reconstruction. The recovered humans can be rendered in novel views using the NeRF of the environment (right).

For our data from TV shows, we do not have access to 3D ground truth for humans, so we perform two evaluations for the human reconstructions. First, we perform a system evaluation by Amazon Mechanical Turk (AMT) workers. For each 3D human reconstruction, we task the annotators to select the rendered result video (our method vs. a baseline) where the human reconstruction is more accurate and consistent with the scene and shot boundary images. Each result video is 10 seconds and provides multiple viewpoints of the person in the scene. We test on our test set, resulting in 2100 human labels from 48 participants who went through quality control (please see Sup. Mat. for more details). We report the percent of choices where our method is preferred over the baselines in Table 6.1. The uncalibrated baseline (first row; without intrinsics or extrinsics) is very rarely preferred over our calibrated baseline (last row; with estimated intrinsics and extrinsics). Having access to estimated intrinsics can help with localization (middle row), but it is still preferred only 35% of the time. Besides the crowd-sourced evaluation, we also evaluate the location of each person quantitatively. In this case, we compute the mean metric distance error for the pelvis

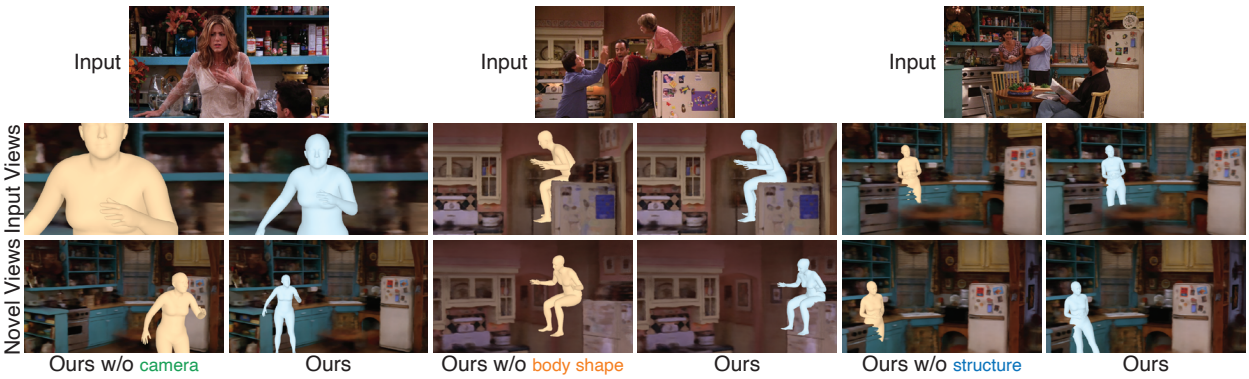

Figure 6.8: **Results for the contextual monocular reconstruction.** We ablate the basic components of the contextual reconstruction to demonstrate their effects. Our method uses all three forms of context. Without our estimated camera intrinsics (left) and without body shape (middle), the person is incorrectly placed in the scene due to scale ambiguity. Using structure (right) avoids interpenetration with the environment.

Table 6.1: **Evaluation of the proposed calibrated multi-shot optimization.** We ablate the effect of camera information in multi-shot optimization. On Human3.6M, we report results on the standard 3D pose metrics in mm [282]. On our TV show data, we perform a system evaluation on AMT and provide quantitative results based on the spatial localization of the reconstructed person in the scene.

Method	Camera information		Human3.6M		TV shows	
Multi-shot optimization	Intrinsics	Extrinsics	MPJPE	PA-MPJPE	% preferred vs. Ours ↑	Distance error ↓
Uncalibrated [160]	✗	✗	131.9	56.9	4%	889cm
Partial Calibration	✓	✗	123.8	56.3	35%	59cm
Calibrated	✓	✓	**65.8**	**47.1**	—	**38cm**

joint in the top-down projection, which is reported in Table 6.1. The conclusions are consistent with the AMT evaluation, highlighting the importance of camera information. Some representative results of our calibrated multi-shot optimization are presented in Figure 6.7, where we also indicate the estimated identity association (which we evaluate in more detail in Section 6.4).

Monocular contextual human reconstruction

Next, we investigate our proposed contextual monocular reconstruction. For this evaluation, we study the effect of each component separately – knowledge of the **cameras**, access to the

Table 6.2: **Ablation of the main components of our contextual reconstruction.** Cross-shot PCK @ $\alpha = 0.5$ is reported. Knowledge of the camera focal length is very important to get a good 3D location for the human. Information about body shape can have significant improvements, as it resolves the scale ambiguity. Structure helps to avoid the incoherent interpenetrations with the scene.

Method	cross-shot PCK
No context: ProHMR [107]	14.7%
No context: PARE [104]	14.2%
No context: SMPLify [16]	16.5%
Context w/o **camera** (intrinsics)	16.0%
Context w/o **body shape** (scale)	65.9%
Context w/o **structure**	87.5%
Context (full)	**88.7%**

person's **body shape**, and finally scene **structure** information. We present the results of this ablation in Table 6.2, where we report cross-shot PCK @ $\alpha = 0.5$ [160]. Effectively, we project the person to the view across the shot boundary and measure localization accuracy for the joints in that space (more details in the Sup. Mat.). First, we see that state-of-the-art monocular methods without context [16, 104, 107] perform similarly on this data. Then, we examine the effect of context, using the optimization baseline [16] as our starting point (third row). Access to camera intrinsics is important to estimate a rough location of the person, and without it the method performs as the baseline without context. Knowledge of the body scale of the person, can make our estimate even more accurate. Finally, structure gives a smaller quantitative improvement but has a more pronounced qualitative effect by placing the person coherently in the environment. See Figure 6.8 for qualitative results.

Re-identification

For the re-ID evaluation, we examine the challenging case of person association after a shot change. For our case, re-ID is directly estimated from our calibrated multi-shot optimization. We compare this result with two types of baselines for computing affinities/costs between instances for Hungarian matching. The first type is image-based re-ID networks for affinity estimation, where [53] achieves SOTA on standard re-ID benchmarks, while [83, 84] are trained on movies, a source of data similar to TV shows. The second type is a geometric baseline that uses our recovered cameras and is based on human keypoint triangulation, where the reprojection error is used as the cost for the Hungarian algorithm. Notice, that unlike SMPL fitting, this does not incorporate anthropometric constraints, i.e., it considers every keypoint match independently, without using human body shape priors or measuring the holistic result. We report re-ID F1 scores in Table 6.3 using the visible pairs of actors

Table 6.3: **Re-ID results for actors in shot boundary frames.** We use different methods to estimate matching costs for detections and we run Hungarian matching to establish associations. A geometric baseline using the reprojection error from person keypoint triangulation improves upon SOTA image-based baselines [53, 83, 84], but using our multi-shot fitting cost performs better because it also includes anthropometric constraints, i.e., the triangulated points should respect the human body priors.

Matching costs	Re-ID F1 ↑
Fu et al. [53] (Appearance)	0.78
Huang et al. [83] (Appearance)	0.79
Huang et al. [84] (Appearance)	0.80
Keypoint triangulation (Geometry)	0.86
Ours (Geometry + Anthropometric)	**0.91**

Table 6.4: **Gaze following results.** We report the Percentage of Correct Gaze Directions (see text for description). Our approach outperforms the baseline of [172].

Method	PCGD ($\alpha = 20^o$) ↑	
	all	w/ face
Recasens et al. [172]	16%	32%
Ours	**62%**	**67%**

before/after the shot change. Based on the results, our re-identification can consistently outperform these baselines.

Gaze estimation

For gaze estimation, we compare with the method of [172] that estimates the gaze target after the shot change. We evaluate the angular error in the gaze direction projected on the image plane. We report the Percentage of Correct Gaze Directions (similar to PCK [263]), using $\alpha = 20^o$ as threshold. Please see Sup. Mat. for details.

Results are reported in Table 6.4. Since [172] relies on face detection, we report results on our whole test set (column "all") and on the subset where face detection is successful (column "w/ face"). Our approach outperforms [172] in both cases. Since we rely on body detection, we are more robust even when the face is occluded. Moreover, our extracted camera poses allow us to follow gaze across shots in a more accurate way. Further improvements are expected by modeling eye pose and saliency estimation to detect the gaze target, similarly to [172]

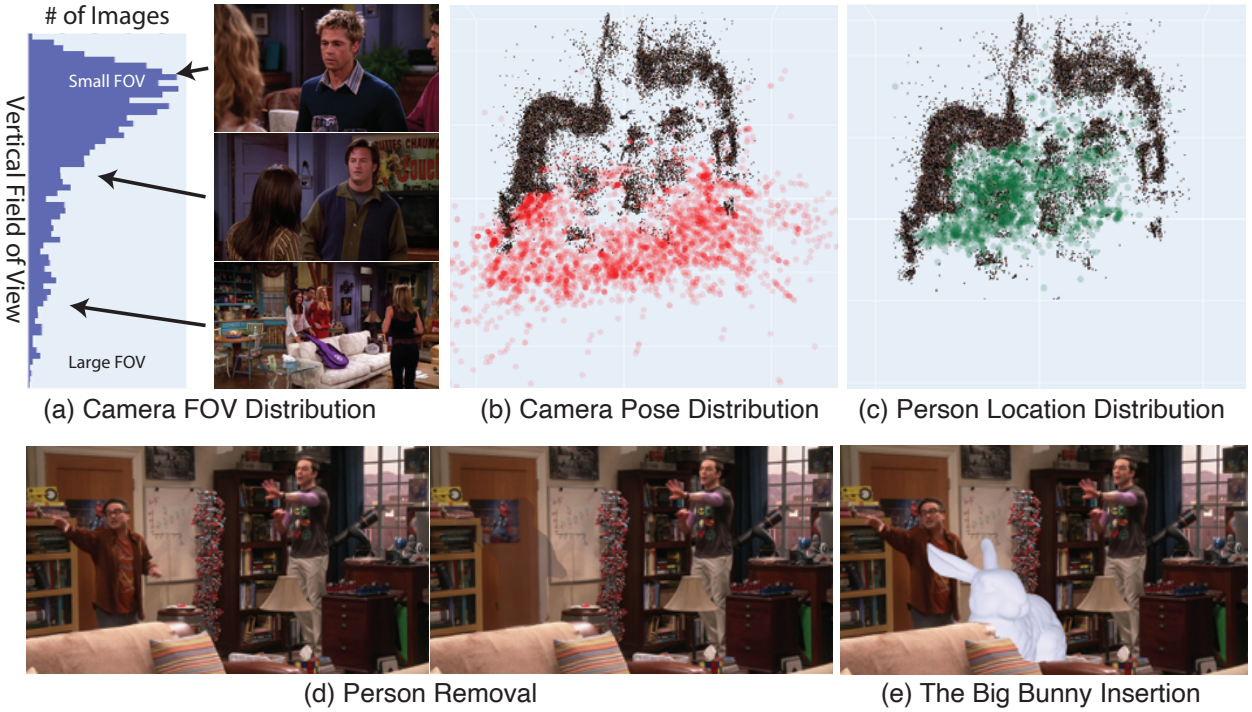

Figure 6.9: **Cinematography applications/Image editing.** We present analysis of our processed data, including distribution of field of view, camera pose distribution and person location distribution for Friends (top). Moreover, we present editing options after our processing, including person removal and object insertion (bottom).

Cinematography/Image editing applications

We provide an initial analysis of our results in Figure 6.9, and present an extended study in the Sup. Mat. First, we visualize the *distribution of the estimated field of view for the cameras*. Here we can see the long-tail distribution for the views with a large field of view, i.e., more informative viewpoints for 3D reconstruction. This justifies our insight to process data across the whole season, since the large majority of the views is typically close-ups. Moreover, we visualize the *locations of the cameras and the human actors*. The camera data could be useful for cinematography analysis [190], and the person data for behavior or affordance analysis [241]. Finally, we illustrate potential editing applications enabled by the reconstruction of humans and the environment: *person removal and object insertion*. More editing options are possible, given the 3D nature of our processing.

6.5 Conclusion

To the best of our knowledge, we are the first to reconstruct the people and the environment in TV shows and reason about them in 3D. We start with multi-shot video sequences associated with a specific environment and recover the camera, structure and relative human scale. We use this information as context to reconstruct humans even from a single frame, in a way that is consistent with their environment. We demonstrate our approach on seven different TV shows and present qualitative and quantitative results, as well as a wide variety of applications and analysis of the reconstructed data.

Our work has only scratched the surface of this extremely challenging and in-depth problem. Currently, we do not reconstruct the transient objects or dynamic objects that humans interact with (e.g., chairs that move around, fridge opening). Also, the recovered pose of the humans is completely dependent on the quality of the 2D keypoint detections. It would be an interesting direction to incorporate appearance models for pose fitting.

Acknowledgements

The authors of this paper are Georgios Pavlakos*, Ethan Weber*, Matthew Tancik, and Angjoo Kanazawa. * denotes equal contribution. This research was supported by the DARPA Machine Common Sense program as well as BAIR/BDD sponsors. Matthew Tancik is supported by the NSF GRFP.

Chapter 7

Toon3D: Reconstructing cartoons

We recover the underlying 3D structure from images of cartoons and anime depicting the same scene. This is an interesting problem domain because images in creative media are often depicted without explicit geometric consistency for storytelling and creative expression—they are only 3D in a qualitative sense. While humans can easily perceive the underlying 3D scene from these images, existing Structure-from-Motion (SfM) methods that assume 3D consistency fail catastrophically. We present Toon3D for reconstructing geometrically inconsistent images. Our key insight is to deform the input images while recovering camera poses and scene geometry, effectively explaining away geometrical inconsistencies to achieve consistency. This process is guided by the structure inferred from monocular depth predictions. We curate a dataset with multi-view imagery from cartoons and anime that we annotate with reliable sparse correspondences using our user-friendly annotation tool. Our recovered point clouds can be plugged into novel-view synthesis methods to experience cartoons from viewpoints never drawn before. We evaluate against classical and recent learning-based SfM methods, where Toon3D is able to obtain more reliable camera poses and scene geometry.

7.1 Introduction

Humans typically have little trouble inferring the relative camera poses and 3D structure from hand-drawn cartoons. However, current structure-from-motion (SfM) pipelines fail to reconstruct these scenes because (1) the images are not geometrically consistent, (2) the images do not obey physically plausible camera models, (3) the scenes are typically only drawn from a sparse set of views, and additionally, (4) many outlier correspondences from automatic methods. In this work, we overcome these challenges by proposing a piecewise-rigid deformable optimization framework that recovers camera poses and 3D scene from geometrically inconsistent images (see Fig. 7.1).

Our pipeline consists of a joint optimization to recover cameras and aligned geometry. It takes a set of correspondences as input, which we backproject into 3D using the depth from a monodepth network [98, 239]. We align these sparse correspondences in 3D to estimate

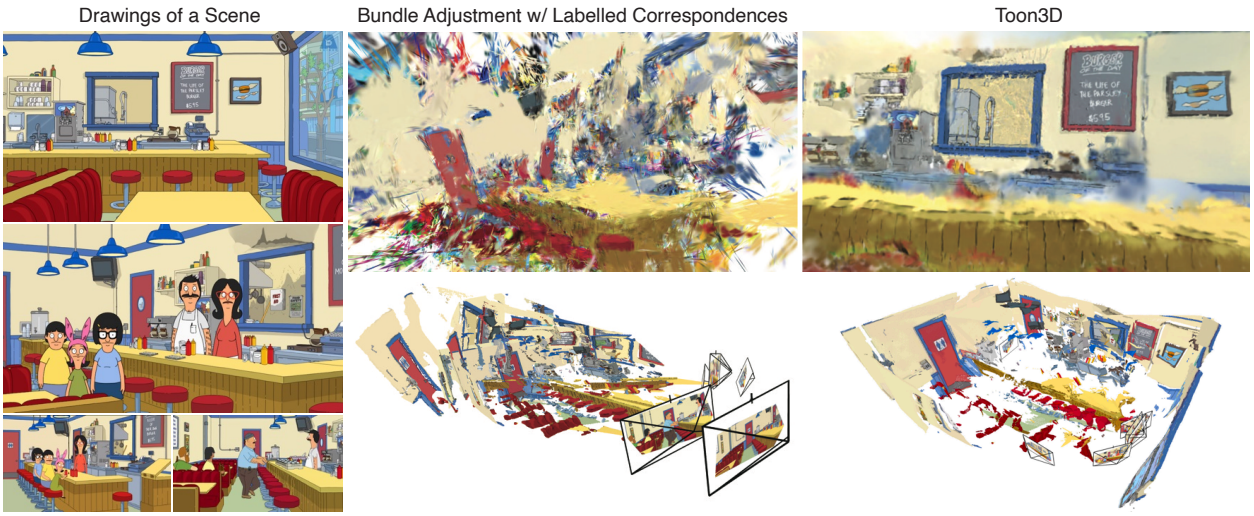

Figure 7.1: **Reconstructing a 3D scene from 3D inconsistent images.** Cartoons and animations often depict scenes that are not geometrically consistent by design (left), making it challenging for classical Structure-from-Motion (SfM) techniques to reconstruct these scenes as they assume 3D consistency (middle). However, humans can easily perceive the underlying 3D scene from these images. We introduce Toon3D, which addresses these challenges by deforming images during reconstruction to account for geometric inconsistencies and leveraging monocular depth priors. The middle column illustrates how Bundle Adjustment fails, even with manually labeled correspondences, resulting in scattered Gaussian splats (top) and misaligned camera reconstructions visualized by backprojected monodepths (bottom). The right column shows our Toon3D results, with more coherent Gaussian splats (top) and well-structured point clouds and camera views (bottom), demonstrating significantly improved 3D consistency. Our project page is `https://toon3d.studio/`.

the camera intrinsic and extrinsic parameters. Simultaneously, we also deform the image and the associated depth such that images satisfy 3D consistency. We regularize our warps with 2D and 3D rigidity losses to prevent degenerate solutions.

We also propose the Toon3D Dataset and the Toon3D Labeler which is a user-friendly annotation tool, where a user can label point correspondences between images while segmenting transient objects. The Toon3D Labeler is a hosted website with no installation, so anyone can get up and running with it easily. We intentionally highlight Toon3D Labeler as a contribution of our paper because artists work with cartoon drawings regularly, and this tool fits nicely into a human-in-the-loop framework for recovering 3D from these drawings. Our recovered 3D model may help artists draw novel viewpoints. We use our labeler to label 12 scenes from popular cartoons and anime, such as Sponge Bob and Spirited Away, and we release these as the Toon3D Dataset.

To the best of our knowledge, we are the first to present a pipeline for reconstruct-

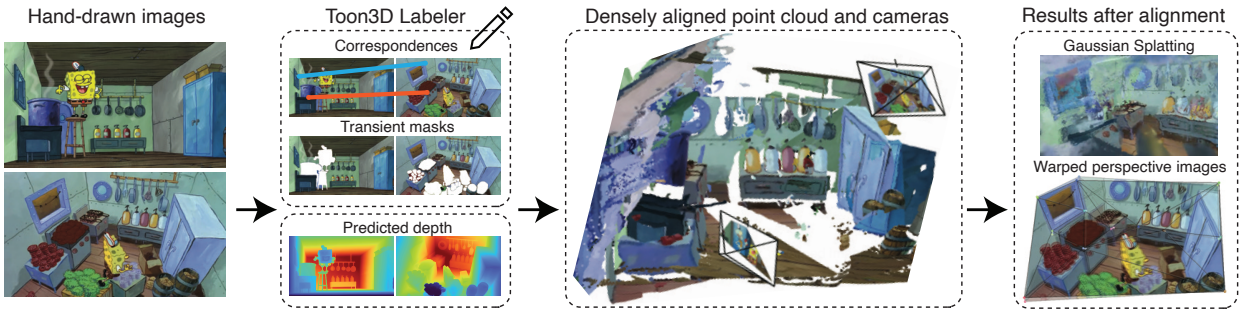

Figure 7.2: **Toon3D overview.** Our framework consists of labeling images with our interactive Toon3D Labeler tool, recovering camera poses and aligning a dense point cloud, and visualizing the dense reconstruction with Gaussians to create an immersive visual experience.

ing cartoon or hand-drawn scenes. Our pipeline yields reliable camera poses, whereas COLMAP [**schoenberger2016structure**] and DUSt3R [240] fail to reliably recover camera poses and 3D scene geometry (even with human-annotated correspondences) due to 3D inconsistencies in the input images. In contrast, our 2D image warpings of the original images enable us to reconstruct the full 3D geometry, while also visualizing geometrical inconsistencies in the drawings.

We evaluate our pipeline on 12 popular scenes (10 cartoon TV shows, 1 movie) to highlight the effectiveness of our pipeline in obtaining good camera poses and reconstructions. We show reconstructions of our recovered 3D point clouds and create an immersive visualization by rendering a 3D Gaussian Splatting [100] representation that are initialized from our aligned point cloud. We evaluate our proposed alignment objectives and losses qualitatively and quantitatively. We demonstrate that our warps can highlight geometric inconsistencies in hand-drawn images. We further validate the quality of Toon3D to estimate camera poses, when the scenes are in fact geometrically consistent. We show that we can obtain the 3D geometry of Airbnb rooms with sparse views. Finally, we show that Toon3D is also useful for reconstructing the 3D geometry from paintings depicting the same landmark from different views.

Humans routinely make successful 3D scene inferences from imagery (e.g. cartoons) which is 3D-inconsistent and/or not following perspective projection [73]. Toon3D is a step toward achieving this type of qualitative 3D understanding of cartoons. We validate our pipeline and will release all data, code, and tools to easily process any cartoon. We hope our contribution serves as a useful framework to build tools that, like humans, can reconstruct and understand qualitative 3D.

7.2 Related work

Multi-view geometry estimation. Structure-from-Motion (SfM) [66, 203] takes in images, detects and matches correspondences, and solves for camera parameters. COLMAP [**schoenberger20** is a popular SfM pipeline, but it fails for wide baseline images (few correspondences), images with a lot of moving objects, or geometric inconsistencies typically present in cartoons. Improvements in keypoint detection [40, 43], matching [209, 188] and optimizations [219] have been proposed to better handle wide baselines [224] and be robust to transient objects [13]. However, all these methods make a fundamental assumption that the input images are geometrically consistent. In contrast, we propose a method that accounts for such inconsistencies by explaining away the inconsistencies when possible via image deformation.

Reconstructing image collections. Facade [36], a seminal early work in image-based modeling and rendering, used a set of photographs of an architectural scene to recover a textured 3D model using structure-from-motion with human-specified volumetric constraints. Phototourism [203] and Building Rome in a Day [3] pioneered the use of large online photo collections for 3D reconstruction. Object-centric methods like CMR [95, 94] recover 3D models of animals through a learned deformation model. For non-rigid dynamic scenes, there exist methods which explain small variations in a video via a 3D model with a time-conditioned warp fied to be as rigid as possible [156, 220, 166]. With methods that require deformation, techniques such as As-Rigid-As-Possible (ARAP) [206] are useful. These problems are relevant in a sense that they need to reconstruct scenes with transient variations in each image. We propose a relevant but novel and under-explored problem setting where the input images are meant to depict the same 3D scene, through geometrically inconsistent multi-view imagery.

Paintings to 3D. Most attempts at recovering 3D from drawings and paintings have focused on the single view setting, with missing 3D information provided either manually by the user or via learning. Important early user-assisted approaches for generating 3D scenes from a single painting include Tour into the Picture [80], which assumed single-point perspective, and the more general Single View Metrology [34]. Automatic Photo Popup [77] replaced the manual parts of the reconstruction process with early machine learning techniques, and was able to generalize to paintings. Aubry et al. [7] is a rare attempt to connect different paintings of the same scene by using a 3D model. There has also been a few attempts to recover a 3D model from a set of sketches of the same object [39, 63]. Our approach similarly explores reconstructing creative expressions (i.e. drawings) but from multiple drawings of the same scene as seen in settings like cartoons instead of a single image.

Computer vision in TV and Film. Previous works have explored reconstructing TV shows and films. Pavlakos et al. [161] recover camera shot locations, 3D human poses, and gaze understanding, enabling applications such as post-production re-rendering with novel camera paths. MovieNet [84] proposes a large dataset of popular films annotated with bounding boxes, actions, and cinematic style for a holistic understanding of movies. Zhu et al. [286] align movies and books to obtain fine-grained descriptions of appearances of objects and characters, as well as high-level semantic understanding into how characters

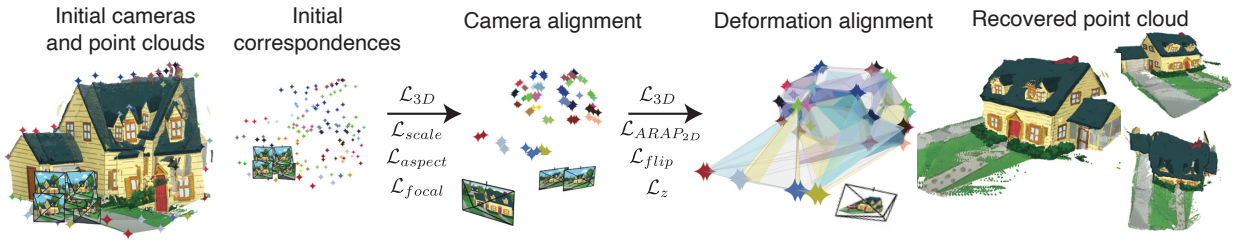

Figure 7.3: **Toon3D alignment.** The camera alignment objective aligns the point clouds while optimizing for camera intrinsics and extrinsics. Deformation alignment deforms the images to obey a perspective camera model. In practice, our method uses all the losses described here to obtain an aligned point cloud and posed images.

think and reason. Additionally, some works have looked into character reconstruction for cartoon characters [31, 86, 202] but none have looked at recovering camera poses and reconstructing full 3D environments. Our work is most similar to [161], but we tackle geometrical inconsistencies in cartoons and animation instead of video sequences in sitcoms.

7.3 Toon3D Dataset and Labeler

To study this unique problem, we introduce the Toon3D Dataset, which consists of 12 cartoon scenes (10 TV shows, 1 movie) each with 5-12 images depicting the same environment. An innate challenge in cartoons is that correspondences are difficult to obtain automatically. We tried several SOTA keypoint detectors [209, 188, 40, 4], but they often fail due to extreme viewpoint changes, the presence of transient objects such as characters, and the images' stylistic, low-texture expression. Since our focus is on reconstructing the underlying static 3D scene, we leave the automatic removal of foreground objects and estimation of 2D correspondences to future work. Instead, we develop the Toon3D Labeler, a human-in-the-loop tool for segmenting transient objects and annotating sparse 2D correspondences. The Toon3D Labeler is hosted online with no installation required, making it easily accessible. See the appendix or project page for a visualization of this tool. Next, we discuss how we use the Toon3D Labeler to curate our dataset.

Preprocessing. We start with a set of N images $\{I_i\}$ depicting the same scene in a cartoon. Each scene typically has $N \leq 10$ images with wide baselines. We preprocess these images by running a monocular depth network to obtain predicted depths $\{D_i\}$. We normalize the depth maps by dividing by the maximum depth of a labeled correspondence across all depth maps. We experiment with a variety of depth map predictors [98, 239, 262], while all quantitative evaluations are done with Marigold [98]. We also run Segment Anything (SAM) [103] to get a set of masks per image.

Labeling. We label these images using the Toon3D Labeler web interface. To annotate correspondences, the user clicks on corresponding points across all images. When the point

is not visible in an image, it is labeled as invisible. Our interface allows users to visualize the depth map, helping them avoid placing correspondences on depth discontinuities. Each annotated image has on average 18 sparse correspondences (see more details in appendix). To select SAM mask, the user simply hovers over a region, the mask will be highlighted, and it can be toggled on and off to discard transients within the image. After labeling, we have pixel correspondences $\mathcal{X} = \{x_{i,c}\}$ where i is the image index and c is the correspondence index. We also have a valid correspondences mask $m_{i,c} = \{0, 1\}$. When $m_{i,c} = 0$, the correspondence is not visible in that image. We denote the predicted depth of the correspondences with $d_{i,c} = D_i(x_{i,c})$.

7.4 Toon3D Method

We present Toon3D, a method to reconstruct scenes that are only 3D consistent in a qualitative sense, as opposed to existing SfM methods that requires a geometrical consistent scene. Toon3D takes as input multiple images of the same scene with point correspondences, mask annotations, and estimated monocular depth, and outputs camera poses for each image, a 3D point cloud, and a warping of the original images, such that they obey a perspective camera model. The output point cloud can be converted into a Gaussian Splatting [100] representation to create a more immersive novel-view experience. We optimize for cameras and geometry by aligning backprojected point correspondences and allowing the images to deform while still obeying a perspective camera model and multiview geometry. We will now explain our approach in more detail.

Camera Alignment

The first objective of our pipeline is to obtain camera poses. Since the images are not geometrically consistent, the standard bundle adjustment process that enforces a single 3D point for every corresponding points does not lead to correct camera poses. Instead, we make use of the monocular depth priors in each image and solve for camera poses that align the backprojected correspondences in 3D.

Specifically, we first backproject our sparse correspondences into 3D with

$$p(x_{i,c}) = R_i \cdot K_i^{-1} \cdot (s_i \cdot d_{i,c} + h_i)\,, \tag{7.1}$$

where the depth d of each point is estimated with a monocular depth network and we solve for camera rotations R, translations t, focal lengths f_x, f_y, depth scale s, and shift h that minimizes the 3D correspondence loss

$$\mathcal{L}_{3D} = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{N} \sum_{j<i}^{N} \sum_{c=1}^{M} m_{i,c} \cdot ||p(x_{i,c}) - p(x_{j,c})||_2^2, \tag{7.2}$$

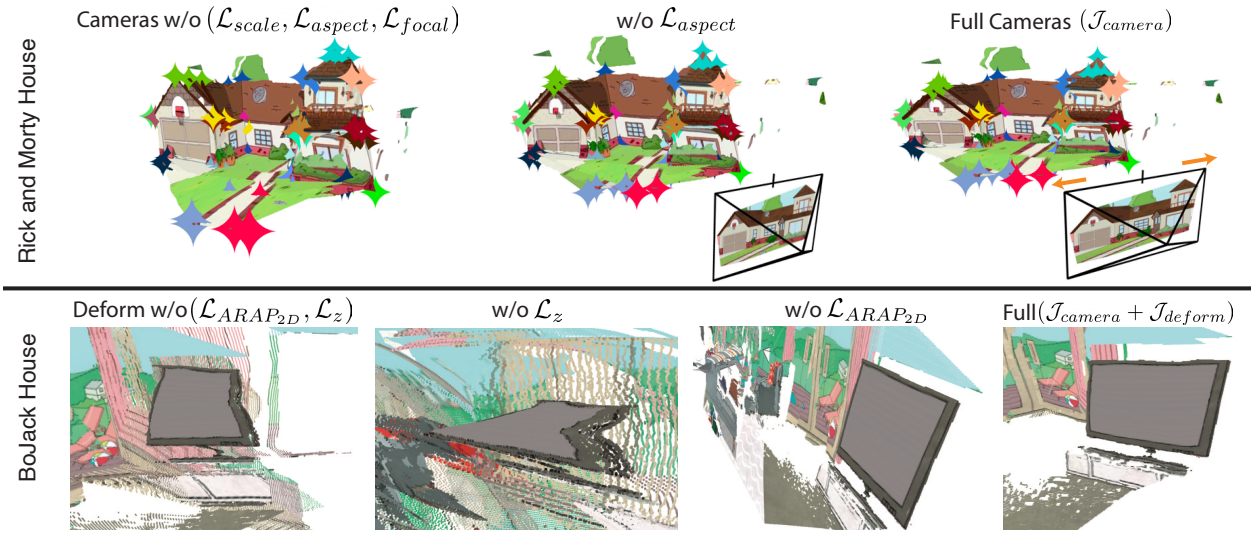

Figure 7.4: **3D alignment ablations.** Row 1 (Rick and Morty House) shows regularization's impact on scene shaping. Optimized shift and scale parameters can adjust point clouds to better align at correspondences. This is evident as the starred points converge. The aspect regularization keeps the optimized image close to its original aspect ratio. Row 2 (BoJack Horseman House) explores the effects of different warp regularizers ($\mathcal{L}_{ARAP_{2D}}$ and $\mathcal{L}_z$) on scene warping. Without any regularization, warping distorts scene geometry. ARAP alone results in poor 3D warps due to inaccurate depth. z regularization alone limits scene movement, maintaining rigid structures close to the original depth map. Using both strikes a good balance between correctly positioning geometry and preserving structural integrity.

which pulls the backprojected correspondences together in 3D. We found minimizing 3D distance rather than 2D reprojection error works better empirically, which we ablate in experiments.

Estimating these camera poses from just few sparse correspondences is a very under-constrained problem even with a strong depth prior. Therefore, we found that adding the following regularizers were necessary to reliably estimate camera poses across all scenes.

$$\mathcal{L}_{scale} = ||1 - \frac{1}{N}\sum_{i=1}^{N} s_i||^2 \tag{7.3}$$

$$\mathcal{L}_{aspect} = \sum_{i=1}^{N} ||\frac{f_{i,x}}{f_{i,y}} - \frac{h_i}{w_i}||^2 \tag{7.4}$$

$$\mathcal{L}_{focal} = \sum_{i=1}^{N} f_{i,x} + f_{i,y}, \tag{7.5}$$

where $\mathcal{L}_{scale}$ encourages a scale close to 1 such that the scene does not shrink, $\mathcal{L}_{aspect}$ balances

$f_{i,x}$ and $f_{i,y}$ to maintain aspect ratio of the camera with the original image's height h_i and width w_i, and $\mathcal{L}_{focal}$ penalizes large focal length to prefer wide-angle cameras over far away and zoomed in shots. We also have losses that penalize scales s_i and shifts h_i if they become negative with $\mathcal{L}_{neg}(x) = ||\frac{1}{N}\sum_{i=1}^{N}\max(0, -x_i)||^2$.

Thus, our final camera alignment objective is as follows

$$\begin{aligned}\mathcal{J}_{\text{camera}} = \mathcal{L}_{3D} + \lambda_{scale}\mathcal{L}_{scale} + \lambda_{aspect}\mathcal{L}_{aspect} \\ +\lambda_{focal}\mathcal{L}_{focal} + \lambda_{neg}(\mathcal{L}_{neg}(s) + \mathcal{L}_{neg}(h)),\end{aligned} \tag{7.6}$$

which gives us an coarse estimate of the 3D structure and the camera poses.

Deformation Alignment

Although the previous losses yield coarse estimates of the scene, they do not result in a coherent point cloud due to the geometric inconsistencies in cartoon images. To address this, we propose jointly deforming each image and its corresponding depth map to achieve geometric consistency. Our method introduces a set of dense alignment objectives, which, when optimized, refine the camera poses. This produces a densely aligned, warped 3D point cloud along with images that are geometrically consistent in 3D and adhere to a perspective camera model.

To do this, we use the same optimization objectives from Sec. 7.4 but now with more freedom as we also allow the input image to be warped to further minimize $\mathcal{L}_{3D}$. However, naively warping every pixel location with full degrees of freedom, without any constraints, results in degenerate solutions. To address this, we warp the image using a coarse 3D mesh that approximates the scene and apply a regularizer to ensure the deformation remains piece-wise rigid.

Specifically, we first transform each training image and predicted depth into a 3D mesh with vertices $V \in \mathbb{R}^{M\times 3}$ and faces $F \in \mathbb{R}^{K\times 3}$, where $V_{i,xy}$ is the initial 2D point for image i and $V_{i,z}$ is the initial depth. We use the labeled correspondences $x_{i,c}$ as the vertices of this mesh. We use Delaunay triangulation to create the mesh topology. See Fig. 7.3 for illustrations of this 3D mesh that represents the scene for each image.

We optimize the V of each image with various 2D and 3D regularizers to constrain the warps to be as-rigid-as-possible to prevent degenerate solutions. First, we regularize such that the optimized vertices are encouraged to follow a rigid transform in the 2D image plane via

$$\mathcal{L}_{ARAP_{2D}} = \frac{1}{N \times |\mathcal{F}|}\sum_{i=1}^{N}\sum_{f\in F_i}||\pi(V_i^{'}[f]) - A_{i\to j}\pi(V_i[f])||^2, \tag{7.7}$$

where π denotes the 2D projection with the current camera parameters, $V_i[f] \in \mathbb{R}^{2\times 3}$ are vertices indexed at face f, V' are the optimized 2D projected vertices, and $A_{a\to b}$ is the best

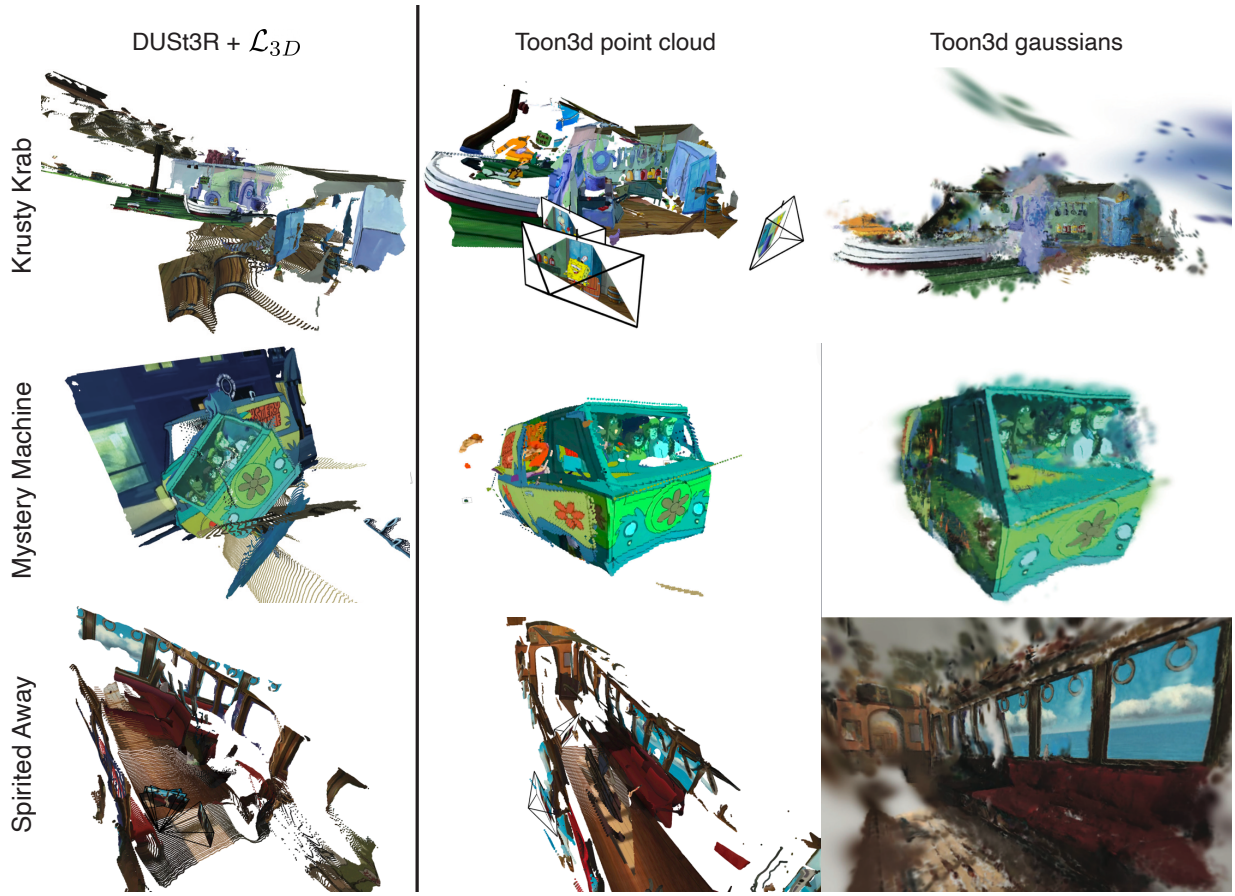

Figure 7.5: **3D reconstructions of cartoons.** Off-the-shelf methods like COLMAP fail completely. State-of-the-art learning based method DUSt3R [240] also fails catastrophically on many scenes even with labeled correspondences (left). Our method (middle), recovers reliable cameras and plausible pointclouds, which can be visualized with Gaussians for a more immersive experience. For the SpongeBob scene (top), we label point correspondences between walls to reconstruct two rooms together. Notably, our method works with different depth predictors. From top to bottom, we show results with MoGe [239], Depth Anything V2 [262], and Marigold [98]. Check the supplement for more comparisons with DUSt3R.

fit 2D rigid transform in the image plane that transforms vertices $V_i[f]$ to the new vertices $V_i^{'}[f]$.

Additionally, we use these two losses

$$\mathcal{L}_{flip} = \frac{1}{N \times |\mathcal{F}|} \sum_{i=1}^{N} \sum_{f \in F_i} || \min(0, t_{area} - \det(V_i[f]))||^2, \tag{7.8}$$

$$\mathcal{L}_z = \frac{1}{N \times |\mathcal{X}|} \sum_{i=1}^{N} \sum_{c=1}^{M} m_{i,c} \cdot ||d'_{i,c} - d_{i,c}||, \tag{7.9}$$

where $\mathcal{L}_{flip}$ penalizes if the triangle face gets too small or flips, and $\mathcal{L}_z$ encourages the warped depth to be close to the original predicted depth. t_{area} is the minimum area a face can be, and det gives the signed face area. We set t_{area} to 10% of the original face area.

Finally, we use barycentric interpolation to densely warp the RGB and depth maps according to our deformed vertices V'. We warp the RGB image with barycentric interpolation according to the original vertices V and the deformed mesh V'. Similarly, we compute a depth offset and apply it to the original depth images d_i to obtain d'_i.

Our deformation alignment objective becomes an extension of our camera alignment, where besides optimizing for poses, focal lengths, rotation, scale, and shift, we also optimize the mesh topology. Our final objective is

$$\underset{R,t,f,s,h,V}{\arg\min}\ \mathcal{J}_{\text{align}} = \mathcal{J}_{\text{camera}} + \mathcal{J}_{\text{deform}} \tag{7.10}$$

$$\mathcal{J}_{\text{deform}} = \lambda_{ARAP_{2D}} \mathcal{L}_{ARAP_{2D}} + \lambda_{flip} \mathcal{L}_{flip} + \lambda_z \mathcal{L}_z \tag{7.11}$$

Optimizing this objective results in an accurate 3D poses as well as an image and depth map that obey the perspective camera model as well as global 3D geometry consistency.

Gaussian visualization

At this point, we have aligned depth maps which are backprojected into a combined 3D point cloud. We could visualize the point cloud as-is, but we find that Gaussian Splatting can create a more immersive experience. Gaussian Splatting [100] is typically initialized by a sparse point cloud from COLMAP, but instead, we initialize it with our dense point cloud. We add a few sparse-view regularizers including the ranking loss from [233] (to reconstruct scenes to be consistent with the predicted depth) and a total variation [281] loss in novel views interpolated between pairs of training views. The transient regions are not ignored in the objective.

7.5 Experiments

First we show results on cartoon scenes, and then we evaluate our design choices and compare our method with DUSt3R [240], a state-of-the-art learning based 3D reconstruction method. We further test the correctness of our approach on a similar setup but with geometrically consistent photos from an AirBnB listing. We also evaluate our approach on paintings and finally, visualize which parts of the images need to warp to become consistent with each other.

Figure 7.6: **Sparse-view Reconstruction.** Our pipeline can reconstruct sparse-view image collections that are geometrically consistent as well (left). COLMAP by default only registers 2 out of 5 images and fails to recover structure (middle top). Using Toon3D Labeler correspondences, we get COLMAP to work (middle bottom) but it is initialized with a very sparse point cloud and cannot recover dense details properly. Using Toon3D, we can fully reconstruct the room.

Cartoon reconstruction

In Fig. 7.5 we show the results from our pipeline on multiple popular cartoon scenes. On the left, we show results using DUSt3R [240], which often fails catastrophically even with our labeled correspondences. The center column shows our point cloud reconstruction. The right column shows rendered novel views after the Gaussian visualization. We also show a traditional bundle adjustment (BA) baseline in Fig. 7.1 that optimizes a single 3D point for each labeled correspondence, which recovers inaccurate poses. For clarity, we visualize the dense result by backprojecting monocular depths. Approaches that don't account for geometrical inconsistencies result in poor camera poses. Please see our overview video for better visualization. From start to completion, our method takes on the order of minutes. Finding a few images of a cartoon scene and labeling points is quick due to the web-based viewer, and running our camera alignment and warping takes approximately 1 minute on an NVIDIA RTX A5000. Running Gaussian Splatting in Nerfstudio [214] with our additional losses takes $\sim$3 minutes.

Table 7.1: **Quantitative ablations.** We report reprojection error for 5 holdout points on our 12 scenes. PCC is evaluated using a threshold radius set to 3% of the image size ($\alpha = 0.03$). (* Includes $\mathcal{L}_{flip}$)

Method	PCC↑	Method	PCC↑
Camera Alignment		*Deformation Alignment*	
$\mathcal{J}_{\text{camera}}$	0.26	$\mathcal{J}_{\text{camera}} + \mathcal{J}_{\text{deform}}$	0.47
$-\mathcal{L}_{focal}$	0.26	$-\mathcal{L}_{z}$	0.42
$-\mathcal{L}_{aspect}$	0.24	$-\mathcal{L}_{ARAP_{2D}}$	0.42
$-\mathcal{L}_{scale}$	0.18	$-\mathcal{J}_{\text{deform}}{}^{*}$	0.36
Traditional BA	0.10	Switch $\mathcal{L}_{3D}$ to $\mathcal{L}_{2D}$	0.31

Qualitative ablations. For our default method, we have all parameters free (including scale and shift) with all regularization losses turned on. We show the qualitative trade-offs for our various losses in Fig. 7.4. We find our losses help align structure while maintaining an accurate aspect ratio, preventing degenerate warps, and favoring cameras inside walls rather than far away and zoomed in (see caption).

Quantitative evaluation Since our task and data have no ground truth cameras, we design a metric to evaluate 3D consistency, with results reported in Tab. 7.1. We randomly remove 5 labeled correspondence points from each image of our 12 Toon3D scenes and report the average percentage of correct correspondences (PCC) across all scenes on these held-out points. Similar to PCK [263], PCC considers a correspondence correct if the reprojected point lies within a radius defined as a percentage *alpha* of the image size. We run our method with various parameters and regularizations turned on and off for ablations and also compare against strong baselines like DUSt3R [240] adapted to our setting, all shown in Fig. 7.7. In order for a fair comparison, we also compare with a version of DUST3R that uses our correspondences via adding $\mathcal{L}_{3D}$ in their global optimization stage along with our labeled masks applied to the confidence map. Results show that our proposed approach obtains the best PCC across all methods. We find that DUST3R works well occasionally, but when it fails, it fails catastrophically (see supplement). Using our labels helps, but not significantly. Adding the dense alignment warp is necessary to significantly increase the performance. Our experiments validate the need for methods designed to deal with geometrically inconsistent input images.

Figure 7.8: **Reconstructing paintings with Toon3D.** Our method enables reconstructing paintings. On the left, we show a few paintings of The Trevi Fountain. In the middle, we show the recovered point cloud and cameras (with warped and cropped images). On the right, we densify the point cloud with Gaussian Splatting.

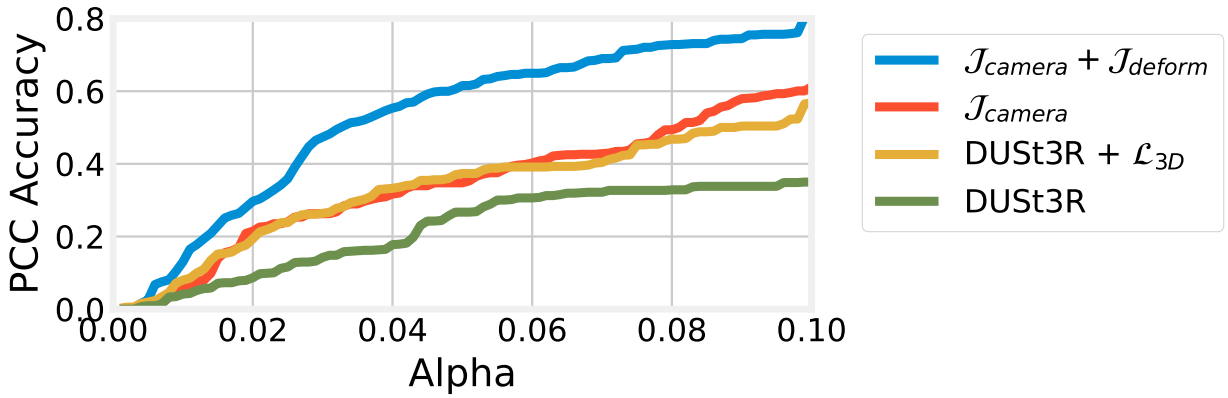

Figure 7.7: **Baselines evaluation.** We obtain best metrics for percent correct correspondences at image size % thresholds α. We beat DUSt3R and improve it with our labels & $\mathcal{L}_{3D}$.

Sparse-view Reconstruction Validation

In this section, we validate the correctness of our approach on image collections that are geometrically consistent. Airbnb listings provide suitable test cases, as their photos are often geometrically consistent but sparse with wide baselines. For this evaluation, we reconstruct sparse photo collections from two Airbnb rooms from a listing (8 photos of a bedroom, shown in the project page, and 5 photos of a living room, shown in Fig. 7.6). This task is very difficult because SfM pipelines like COLMAP fail to find enough correspondences to accurately recover all poses. Furthermore, even with accurate camera poses, the sparse-view

reconstruction setting is especially hard without priors or specialized methods like RegNeRF [152] or ReconFusion [256]. We tackle this sparse-view Airbnb setting with our method for two reasons: (1) to show that we can get COLMAP to work with labeled correspondences from the Toon3D Labeler and (2) to show that our approach works for real sparse photo collections, indicating applications beyond cartoons.

When running COLMAP on our Airbnb collections, default COLMAP only registers 46% of the images. This could be possibly improved with better correspondences, e.g. [40, 223, 43], but there is no guarantee of finding enough inlier correspondences if automated methods are used. With our Toon3D Labeler, however, we can manually label the images quickly and get COLMAP to succeed for all images. We compare the recovered COLMAP cameras with our correspondences with the cameras recovered from Toon3D. The mean relative rotation distance between corresponding pairs in our reconstructions vs. COLMAP's is quite low at only 8.29°, indicating our cameras are similar to ones recovered by COLMAP with human-labeled correspondences. We do not compare translations or focal lengths due to ambiguity between the two, but we note that our camera relative rotations match COLMAP quite well, suggesting that our camera pose estimation is accurate. We show qualitative results for sparse-view reconstruction on real images in Fig. 7.6 and videos on the project page.

Reconstructing paintings

We also show our pipeline can reconstruct paintings of the same scene. Fig. 7.8 shows results of Toon3D on paintings of The Trevi Fountain found in the Oxford Dataset [35]. This setup requires multiple paintings of the same scene from diverse viewpoints, which is uncommon. However, it presents an interesting problem, and notably, we are able to apply the Toon3D pipeline successfully without any modifications.

Visualizing inconsistencies

One unique aspect of Toon3D is that we keep the original images around rather than discarding them. They are warped in 2D to obey the global 3D consistency through a perspective camera model. This is fundamentally different than alternative sparse-view generative methods, e.g. Dreambooth3D [169] which fine-tunes on a collection of images and then hallucinates a scene. In Fig. 7.9 we show where the images deform the most to create a unified consistent 3D structure. Additionally, it provides insights into the artistic techniques used to convey 3D or to emphasize regions in drawings without strictly adhering to physical laws.

Figure 7.9: **Visualizing inconsistencies.** We show the most inconsistent regions in a few images from different scenes by overlaying the original image (left) on top of the deformed image (right) to construct a difference image (middle). Blurry regions indicate where warping occurs.

7.6 Conclusion

We present Toon3D, a pipeline for 3D reconstruction from geometrically inconsistent images of a scene found in settings such as cartoons and animations. This is an interesting setup as humans have no issue interpreting a hand-drawn scene in 3D, but existing 3D reconstruction methods struggle in various ways. We propose a method that takes advantage of labeled correspondences and predicted depth priors to reconstruct these scenes by explaining away their inconsistencies by deforming the images to obey perspective projection models with regularizations. Many exciting future directions remain, such as incorporating diffusion priors or data-driven methods to reconstruct cartoons end-to-end. Our recovered cameras

cameras and point clouds could even be useful to train supervised methods like DUSt3R. We encourage ethical and responsible use of our work.

Acknowledgements

The authors of this paper are Ethan Weber*, Riley Peterlinz*, Rohan Mathur, Frederik Warburg, Alexei A. Efros, and Angjoo Kanazawa. * denotes equal contribution. This project is supported in part by IARPA DOI/IBC 140D0423C0035. The views and conclusions contained herein are those of the authors and do not represent the official policies or endorsements of IARPA, DOI/IBC, of the U.S. Government. We would like to thank Qianqian Wang, Justin Kerr, Brent Yi, David McAllister, Matthew Tancik, Evonne Ng, Anjali Thakrar, Christian Foley, Abhishek Kar, Georgios Pavlakos, the Nerfstudio team, and the KAIR lab for discussions, feedback, and technical support. We also thank Ian Mitchell and Roland Jose for helping to label points.

Chapter 8

Conclusion

In this thesis, we have explored the challenges and opportunities of 3D reconstruction and generation from casual, in-the-wild data. Our work addresses the limitations of current tools in handling imperfect captures, missing regions, and diverse data sources. By developing robust frameworks for 3D reconstruction and generative scene completion, we take significant steps toward enabling machines to recover and create rich 3D experiences from the everyday visual world. We make progress on the title of this thesis, yet there is still much work to be done, which we highlight in this section.

8.1 Reflections

When I began my PhD, recovering cameras with COLMAP would take a long time (hours), and training a reasonably small NeRF required an entire day of training time. Generative methods for 3D were in their infancy, with little active research in the area. Since then, the fields of structure-from-motion (SfM), multi-view stereo (MVS), and novel view synthesis (NVS) have advanced at a remarkable pace. Today, camera recovery and NeRF or 3DGS training can be accomplished in under a minute, and generative 3D methods have become a vibrant and rapidly advancing field. COLMAP [192, 193] has long been the standard tool for SfM and camera recovery, but recent years have seen rapid progress with the advent of data-driven and feed-forward regression techniques such as DUSt3R [240], MAST3R [109], MegaSaM [113], CUT3R [237], and others. On the NVS front, methods like NeRF [138] have evolved toward more expressive and efficient representations, most notably 3D Gaussian Splatting [100], which continue to push the boundaries of quality and speed.

As these algorithms develop, we are increasingly able to reconstruct 3D scenes from more casual, unconstrained data sources—a central theme of this thesis. Data-driven, end-to-end approaches will play a crucial role in this progress. However, our work on Toon3D demonstrates that human-in-the-loop systems remain invaluable: when state-of-the-art automated methods like DUSt3R fail, targeted human intervention (e.g., adding sparse correspondences) can enable successful reconstruction where fully automatic pipelines break down.

Looking ahead, the trajectory of this field is promising. Continued improvements in 2D correspondence estimation, monocular depth prediction [14, 262], and 3D scene representations [138, 100] will further expand the range of reconstructable data. We are building tools for casual data in an era of rapid progress, and I am optimistic about future applications—such as reconstructing movies (which are even sparser than TV shows) or enabling localization across disparate viewpoints in scenarios like natural disasters, where aerial and ground views must be aligned.

We have benefited from all of these key advances, but perhaps the most significant enabler for future progress is the rise of generative methods powered by large datasets. These approaches allow us to hallucinate plausible content in unseen regions of a scene [78, 165]. As models continue to grow larger and more sophisticated—a trend we have only begun to explore with Fillerbuster—I expect generative methods to become even more powerful. Ongoing research in this area promises to further expand the boundaries of what is possible in 3D scene completion.

8.2 Outstanding Scene Completion Challenges

Despite the progress made in this thesis, scene completion remains a fundamentally challenging problem. Many real-world scenes contain large occlusions, ambiguous or missing regions, and complex interactions between objects that are difficult to infer from limited observations. Current generative models, while powerful, can struggle to produce globally consistent and semantically plausible completions, especially when the missing regions are large or when the available context is sparse or noisy. Furthermore, ensuring multi-view consistency—so that generated regions look correct from all possible viewpoints—remains a important research direction. There are also challenges in evaluating scene completion: without ground truth, it is difficult to measure the quality and realism of generated content. Addressing these issues will require advances in generative modeling, better integration of geometric and semantic priors, and new benchmarks and evaluation protocols. Enhancing the fidelity of generative scene completion has the potential to enable more immersive and realistic 3D experiences, such as in virtual reality, where the objective is to experience a space as if you are there. It also holds significant promise for applications in visual media entertainment and in training robots to interact effectively within complex environments.

8.3 Toward More Casual Data

A central theme of this thesis is the ambition to make 3D reconstruction and generation work on increasingly casual and unconstrained data. While we have demonstrated methods that succeed on mobile phone videos, TV shows, cartoons, and other challenging sources such as AirBnB photos, there remains a vast spectrum of visual data in the world that is even more diverse and unstructured. Examples include social media clips, surveillance

footage, historical archives, livestreams, artistic or abstract representations, old footage, low resolution data, and other data captured but not intended for reconstruction. Each of these domains presents unique challenges—ranging from extreme viewpoint variation and motion blur to non-photorealistic rendering and intentional geometric inconsistency. The insights and frameworks developed in this thesis lay the groundwork for tackling these frontiers. By continuing to push the boundaries of what constitutes "reconstructable" or "generatable" data, we move closer to a future where machines can understand and recreate the 3D world from any visual input, no matter how casual or unconventional.

Bibliography

[1] Martín Abadi et al. "{TensorFlow}: a system for {Large-Scale} machine learning". In: *12th USENIX symposium on operating systems design and implementation (OSDI 16)*. 2016, pp. 265–283.

[2] Michal Adamkiewicz et al. "Vision-Only Robot Navigation in a Neural Radiance World". In: *CoRR* abs/2110.00168 (2021). arXiv: 2110.00168. URL: https://arxiv.org/abs/2110.00168.

[3] Sameer Agarwal et al. "Building Rome in a day". In: *2009 IEEE 12th International Conference on Computer Vision*. 2009, pp. 72–79. DOI: 10.1109/ICCV.2009.5459148.

[4] Shir Amir et al. "Deep vit features as dense visual descriptors". In: *arXiv preprint arXiv:2112.05814* 2.3 (2021), p. 4.

[5] Ognjen Arandjelovic and Andrew Zisserman. "Automatic face recognition for film character retrieval in feature-length films". In: *CVPR*. 2005.

[6] Daniel Arijon. *Grammar of the film language*. Hastings House, 1976.

[7] M. Aubry, B. Russell, and J. Sivic. "Painting-to-3D Model Alignment Via Discriminative Visual Elements". In: *ACM Transactions on Graphics* (2013).

[8] Luca Ballan et al. "Unstructured video-based rendering: interactive exploration of casually captured videos". In: *ACM Transactions on Graphics (TOG)* 29.4 (2010), pp. 1–11.

[9] Amir Bar et al. "Visual prompting via image inpainting". In: *ANeurIPS*. 2022.

[10] Omer Bar-Tal et al. "Multidiffusion: Fusing diffusion paths for controlled image generation". In: *ICML*. 2023.

[11] Jonathan T Barron et al. "Mip-nerf 360: Unbounded anti-aliased neural radiance fields". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5470–5479.

[12] Jonathan T Barron et al. "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5855–5864.

[13] Berta Bescos et al. "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 4076–4083.

[14] Shariq Farooq Bhat et al. "Zoedepth: Zero-shot transfer by combining relative and metric depth". In: *arXiv preprint arXiv:2302.12288*. 2023.

[15] Andreas Blattmann et al. "Stable video diffusion: Scaling latent video diffusion models to large datasets". In: *arXiv preprint arXiv:2311.15127* (2023).

[16] Federica Bogo et al. "Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image". In: *ECCV*. 2016.

[17] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. "InstructPix2Pix: Learning to Follow Image Editing Instructions". In: *CVPR*. 2023.

[18] Andrew Brown, Vicky Kalogeiton, and Andrew Zisserman. "Face, Body, Voice: Video Person-Clustering with Multiple Modalities". In: *ICCVW*. 2021.

[19] Michael Broxton et al. "Immersive Light Field Video with a Layered Mesh Representation". In: *SIGGRAPH*. 2020.

[20] Arunkumar Byravan et al. *NeRF2Real: Sim2real Transfer of Vision-guided Bipedal Motion Skills using Neural Radiance Fields*. 2022. DOI: 10.48550/ARXIV.2210.04932. URL: https://arxiv.org/abs/2210.04932.

[21] Shengqu Cai et al. "DiffDreamer: Consistent single-view perpetual view generation with conditional diffusion models". In: *ICCV*. 2023.

[22] Chenjie Cao et al. "MVInpainter: Learning Multi-View Consistent Inpainting to Bridge 2D and 3D Editing". In: *NeurIPS*. 2024.

[23] Zhe Cao et al. "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields". In: *PAMI* (2019).

[24] Rohan Chabra et al. "Deep local shapes: Learning local sdf priors for detailed 3d reconstruction". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*. Springer. 2020, pp. 608–625.

[25] Lucy Chai et al. "Persistent Nature: A Generative Model of Unbounded 3D Worlds". In: *CVPR*. 2023.

[26] Eric R Chan et al. "Generative novel view synthesis with 3d-aware diffusion models". In: *arXiv*. 2023.

[27] Angel X Chang et al. "Shapenet: An information-rich 3d model repository". In: *arXiv preprint arXiv:1512.03012* (2015).

[28] Anpei Chen et al. "TensoRF: Tensorial Radiance Fields". In: *European Conference on Computer Vision (ECCV)*. 2022.

[29] Honghua Chen, Chen Change Loy, and Xingang Pan. "MVIP-NeRF: Multi-view 3D Inpainting on NeRF Scenes via Diffusion Prior". In: *CVPR*. 2024.

[30] Jiafu Chen et al. "Single-Mask Inpainting for Voxel-based Neural Radiance Fields". In: *ECCV*. 2024.

[31] Shuhong Chen et al. "PAniC-3D: Stylized Single-view 3D Reconstruction from Portraits of Anime Characters". In: *CVPR*. 2023.

[32] Zhaoxi Chen, Guangcong Wang, and Ziwei Liu. "SceneDreamer: Unbounded 3d scene generation from 2d image collections". In: 2023.

[33] Robert L Cook, Loren Carpenter, and Edwin Catmull. "The Reyes image rendering architecture". In: *ACM SIGGRAPH Computer Graphics* 21.4 (1987), pp. 95–102.

[34] Antonio Criminisi, Ian Reid, and Andrew Zisserman. "Single view metrology". In: *IJCV* (2000).

[35] Elliot J Crowley and Andrew Zisserman. "In search of art". In: *Computer Vision-ECCV 2014 Workshops: Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part I 13*. Springer. 2015, pp. 54–70.

[36] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. "Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach". In: *SIGGRAPH'96*. 1996.

[37] Matt Deitke et al. "Objaverse-xl: A universe of 10m+ 3d objects". In: *arXiv*. 2023.

[38] Matt Deitke et al. "Objaverse: A universe of annotated 3d objects". In: *CVPR*. 2023.

[39] Johanna Delanoy et al. "3d sketching using multi-view deep volumetric prediction". In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.1 (2018), pp. 1–22.

[40] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "Superpoint: Self-supervised interest point detection and description". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 224–236.

[41] Terrance DeVries et al. "Unconstrained scene generation with locally conditioned radiance fields". In: *ICCV*. 2021.

[42] Danny Driess et al. "Learning Multi-Object Dynamics with Compositional Neural Radiance Fields". In: *arXiv preprint arXiv:2202.11855* (2022).

[43] Mihai Dusmanu et al. "D2-net: A trainable cnn for joint detection and description of local features". In: *arXiv preprint arXiv:1905.03561* (2019).

[44] Alexei A Efros and Thomas K Leung. "Texture synthesis by non-parametric sampling". In: *ICCV*. 1999.

[45] Patrick Esser et al. "Scaling rectified flow transformers for high-resolution image synthesis". In: *ICML*. 2024.

[46] Mark Everingham, Josef Sivic, and Andrew Zisserman. ""Hello! My name is... Buffy" – Automatic Naming of Characters in TV Video". In: *BMVC*. 2006.

[47] Mark Everingham and Andrew Zisserman. "Identifying individuals in video by combining generative and discriminative head models". In: *ICCV*. 2005.

[48] Vittorio Ferrari, Manuel Marín-Jiménez, and Andrew Zisserman. "Pose search: retrieving people using their pose". In: *CVPR*. 2009.

[49] Vittorio Ferrari, Manuel Marín-Jiménez, and Andrew Zisserman. "Progressive search space reduction for human pose estimation". In: *CVPR*. 2008.

[50] Rafail Fridman et al. "Scenescape: Text-driven consistent scene generation". In: *NeurIPS*. 2024.

[51] Sara Fridovich-Keil et al. "K-planes: Explicit radiance fields in space, time, and appearance". In: *arXiv preprint arXiv:2301.10241* (2023).

[52] Sara Fridovich-Keil et al. "Plenoxels: Radiance fields without neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5501–5510.

[53] Dengpan Fu et al. "Unsupervised pre-training for person re-identification". In: *CVPR*. 2021.

[54] Rinon Gal et al. "An image is worth one word: Personalizing text-to-image generation using textual inversion". In: *arXiv*. 2022.

[55] Chen Gao et al. "Dynamic View Synthesis from Dynamic Monocular Video". In: *ICCV*. 2021.

[56] Hang Gao et al. "Monocular dynamic view synthesis: A reality check". In: *Advances in Neural Information Processing Systems*. 2022.

[57] Ruiqi Gao et al. "CAT3D: Create anything in 3D with multi-view diffusion models". In: *NeurIPS*. 2024.

[58] Stuart Geman and Donald E McClure. "Statistical methods for tomographic image reconstruction". In: *Bulletin of the International Statistical Institute* 4 (1987), pp. 5–21.

[59] Michal Geyer et al. "TokenFlow: Consistent Diffusion Features for Consistent Video Editing". In: *arXiv*. 2023.

[60] Shiry Ginosar et al. "Learning individual styles of conversational gesture". In: *CVPR*. 2019.

[61] Lily Goli et al. "Bayes' Rays: Uncertainty Quantification for Neural Radiance Fields". In: *CVPR*. 2024, pp. 20061–20070.

[62] Ian Goodfellow et al. "Generative adversarial nets". In: *NeurIPS*. Vol. 27. 2014.

[63] Benoit Guillard et al. "Sketch2mesh: Reconstructing and editing 3d shapes from sketches". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13023–13032.

[64] Vladimir Guzov et al. "Human POSEitioning System (HPS): 3D Human Pose Estimation and Self-localization in Large Scenes from Body-Mounted Sensors". In: *CVPR*. 2021.

[65] Ayaan Haque et al. "Instruct-NeRF2NeRF: Editing 3D scenes with instructions". In: *ICCV*. 2023, pp. 19740–19750.

[66] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[67] Richard I Hartley. "In defense of the eight-point algorithm". In: *IEEE Transactions on pattern analysis and machine intelligence* 19.6 (1997), pp. 580–593.

[68] Mohamed Hassan et al. "Resolving 3D human pose ambiguities with 3D scene constraints". In: *ICCV*. 2019.

[69] Hao He et al. "CameraCtrl: Enabling camera control for text-to-video generation". In: *ICLR*. 2025.

[70] Kaiming He et al. "Mask R-CNN". In: *ICCV*. 2017.

[71] Peter Hedman and Johannes Kopf. "Instant 3D Photography". In: *SIGGRAPH*. 2018.

[72] Peter Hedman et al. "Casual 3D Photography". In: *SIGGRAPH Asia*. 2017.

[73] Aaron Hertzmann. "Toward a theory of perspective perception in pictures". In: *Journal of Vision* 24.4 (2024), pp. 23–23.

[74] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[75] Jonathan Ho and Tim Salimans. "Classifier-free diffusion guidance". In: *NeurIPS Workshop on Deep Generative Models and Downstream Applications*. 2022.

[76] Minh Hoai and Andrew Zisserman. "Talking heads: Detecting humans and recognizing their interactions". In: *CVPR*. 2014.

[77] Derek Hoiem, Alexei A Efros, and Martial Hebert. "Automatic photo pop-up". In: *ACM SIGGRAPH 2005 Papers*. 2005, pp. 577–584.

[78] Lukas Höllein et al. "Text2room: Extracting textured 3d meshes from 2d text-to-image models". In: *ICCV*. 2023.

[79] Namdar Homayounfar, Sanja Fidler, and Raquel Urtasun. "Sports field localization via deep structured models". In: *CVPR*. 2017.

[80] Youichi Horry, Ken-Ichi Anjyo, and Kiyoshi Arai. "Tour into the picture: using a spidery mesh interface to make animation from a single image". In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 1997, pp. 225–232.

[81] Yuanming Hu et al. "Taichi: a language for high-performance computation on spatially sparse data structures". In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–16.

[82] Po-Han Huang et al. "DeepMVS: Learning multi-view stereopsis". In: *CVPR*. 2018.

[83] Qingqiu Huang, Wentao Liu, and Dahua Lin. "Person search in videos with one portrait through visual and temporal links". In: *ECCV*. 2018.

[84] Qingqiu Huang et al. "MovieNet: A holistic dataset for movie understanding". In: *ECCV*. 2020.

[85] Catalin Ionescu et al. "Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments". In: *PAMI* (2013).

[86] Eakta Jain et al. "Three-dimensional proxies for hand-drawn characters". In: *ACM Transactions on Graphics (ToG)* 31.1 (2012), pp. 1–16.

[87] Wenzel Jakob et al. *Mitsuba 3 renderer*. Version 3.1.1. https://mitsuba-renderer.org. 2022.

[88] Rasmus Jensen et al. "Large scale multi-view stereopsis evaluation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2014, pp. 406–413.

[89] Yoonwoo Jeong, Seungjoo Shin, and Kibaek Park. *NeRF-Factory: An awesome PyTorch NeRF collection*. 2022. URL: `https://github.com/kakaobrain/NeRF-Factory/`.

[90] Yangqing Jia et al. "Caffe: Convolutional architecture for fast feature embedding". In: *Proceedings of the 22nd ACM international conference on Multimedia*. 2014, pp. 675–678.

[91] Wen Jiang et al. "Coherent reconstruction of multiple humans from a single image". In: *CVPR*. 2020.

[92] Haian Jin et al. "LVSM: A Large View Synthesis Model with Minimal 3D Inductive Bias". In: *arXiv preprint arXiv:2410.17242* (2024).

[93] Yuhe Jin et al. "Image matching across wide baselines: From paper to practice". In: *International Journal of Computer Vision* 129.2 (2021), pp. 517–547.

[94] Angjoo Kanazawa et al. "Learning 3d deformation of animals from 2d images". In: *Computer Graphics Forum*. 2016.

[95] Angjoo Kanazawa et al. "Learning category-specific mesh reconstruction from image collections". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 371–386.

[96] Yash Kant et al. "iNVS: Repurposing Diffusion Inpainters for Novel View Synthesis". In: *SIGGRAPH Asia*. 2023.

[97] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson surface reconstruction". In: *Proceedings of the fourth Eurographics symposium on Geometry processing*. Vol. 7. 2006.

[98] Bingxin Ke et al. *Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation*. 2023. arXiv: `2312.02145 [cs.CV]`.

[99] Junjie Ke et al. "Musiq: Multi-scale image quality transformer". In: *ICCV*. 2021.

[100] Bernhard Kerbl et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering". In: *SIGGRAPH*. 2023.

[101] Justin Kerr et al. "Evo-NeRF: Evolving NeRF for Sequential Robot Grasping of Transparent Objects". In: *6th Annual Conference on Robot Learning*. 2022.

[102] Diederik P Kingma. "Auto-encoding variational Bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[103] Alexander Kirillov et al. "Segment anything". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 4015–4026.

[104] Muhammed Kocabas et al. "PARE: Part attention regressor for 3D human body estimation". In: *ICCV*. 2021.

[105] Muhammed Kocabas et al. "SPEC: Seeing People in the Wild with an Estimated Camera". In: *ICCV*. 2021.

[106] Jing Yu Koh et al. "Pathdreamer: A world model for indoor navigation". In: *ICCV*. 2021.

[107] Nikos Kolotouros et al. "Probabilistic modeling for human mesh recovery". In: *ICCV*. 2021.

[108] Johannes Kopf, Xuejian Rong, and Jia-Bin Huang. "Robust consistent video depth estimation". In: *CVPR*. 2021.

[109] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. "Grounding Image Matching in 3D with MASt3R". In: *ECCV*. 2024.

[110] Jialu Li and Mohit Bansal. "PanoGen: Text-Conditioned Panoramic Environment Generation for Vision-and-Language Navigation". In: *NeurIPS*. 2023.

[111] Ruilong Li, Matthew Tancik, and Angjoo Kanazawa. "NerfAcc: A General NeRF Accleration Toolbox." In: *arXiv preprint arXiv:2210.04847* (2022).

[112] Zhengqi Li et al. "Infinitenature-zero: Learning perpetual view generation of natural scenes from single images". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I*. Springer. 2022, pp. 515–534.

[113] Zhengqi Li et al. "Megasam: Accurate, fast, and robust structure and motion from casual dynamic videos". In: *arXiv preprint arXiv:2412.04463* (2024).

[114] Zhengqi Li et al. "Neural scene flow fields for space-time view synthesis of dynamic scenes". In: *CVPR*. 2021.

[115] Chen-Hsuan Lin et al. "Barf: Bundle-adjusting neural radiance fields". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5741–5751.

[116] Chieh Hubert Lin et al. "Taming Latent Diffusion Model for Neural Radiance Field Inpainting". In: *ECCV*. 2024.

[117] Yaron Lipman et al. "Flow matching for generative modeling". In: *ICLR*. 2022.

[118] Andrew Liu et al. "Infinite nature: Perpetual view generation of natural scenes from a single image". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 14458–14467.

[119] Fangfu Liu et al. "ReconX: Reconstruct Any Scene from Sparse Views with Video Diffusion Model". In: *arXiv preprint arXiv:2408.16767* (2024).

[120] Lingjie Liu et al. "Neural sparse voxel fields". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 15651–15663.

[121] Miao Liu et al. "4D human body capture from egocentric video via 3D scene grounding". In: *3DV*. 2021.

[122] Ruoshi Liu et al. "Zero-1-to-3: Zero-shot one image to 3D object". In: *ICCV*. 2023, pp. 9298–9309.

[123] Xi Liu, Chaoyi Zhou, and Siyu Huang. "3dgs-enhancer: Enhancing unbounded 3d gaussian splatting with view-consistent 2d diffusion priors". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 133305–133327.

[124] Xinhang Liu et al. "Deceptive-NeRF/3DGS: Diffusion-Generated Pseudo-Observations for High-Quality Sparse-View Reconstruction". In: *ECCV*. 2024.

[125] Yuan Liu et al. "SyncDreamer: Learning to Generate Multiview-consistent Images from a Single-view Image". In: *arXiv*. 2023.

[126] Matthew Loper et al. "SMPL: A skinned multi-person linear model". In: *ACM Transactions on Graphics (TOG)* 34.6 (2015), pp. 1–16.

[127] Andreas Lugmayr et al. "Repaint: Inpainting using denoising diffusion probabilistic models". In: *CVPR*. 2022.

[128] Luma AI, Inc. *Luma Flythroughs*. 2024. URL: `https://lumalabs.ai/flythroughs`.

[129] Xuan Luo et al. "Consistent video depth estimation". In: *ACM Transactions on Graphics (TOG)* 39.4 (2020), pp. 71–1.

[130] Manuel J Marín-Jiménez et al. "LAEO-Net++: Revisiting people looking at each other in videos". In: *PAMI* (2021).

[131] Manuel Jesús Marín-Jiménez et al. "Detecting people looking at each other in videos". In: *IJCV* (2014).

[132] Ricardo Martin-Brualla et al. "Nerf in the wild: Neural radiance fields for unconstrained photo collections". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 7210–7219.

[133] N. Max. "Optical models for direct volume rendering". In: *IEEE Transactions on Visualization and Computer Graphics* 1.2 (1995), pp. 99–108. DOI: `10.1109/2945.468400`.

[134] Luke Melas-Kyriazi et al. "RealFusion: 360 {\deg} Reconstruction of Any Object from a Single Image". In: *arXiv preprint arXiv:2302.10663* (2023).

[135] Ben Mildenhall et al. "Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines". In: *SIGGRAPH*. 2019.

[136] Ben Mildenhall et al. *MultiNeRF: A Code Release for Mip-NeRF 360, Ref-NeRF, and RawNeRF*. 2022. URL: `https://github.com/google-research/multinerf`.

[137] Ben Mildenhall et al. "Nerf in the dark: High dynamic range view synthesis from noisy raw images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16190–16199.

[138] Ben Mildenhall et al. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[139] Ashkan Mirzaei et al. "Reference-guided Controllable Inpainting of Neural Radiance Fields". In: *ICCV*. 2023.

[140] Ashkan Mirzaei et al. "SPIn-NeRF: Multiview Segmentation and Perceptual Inpainting with Neural Radiance Fields". In: *CVPR*. 2023.

[141] Paritosh Mittal et al. "Autosdf: Shape priors for 3d completion, reconstruction and generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 306–315.

[142] Norman Müller et al. "DiffRF: Rendering-Guided 3D Radiance Field Diffusion". In: *arXiv preprint arXiv:2212.01206* (2022).

[143] Norman Müller et al. "MultiDiff: Consistent Novel View Synthesis from a Single Image". In: *CVPR*. 2024, pp. 10258–10268.

[144] Thomas Müller. *tiny-cuda-nn*. Version 1.6. Apr. 2021. URL: `https://github.com/NVlabs/tiny-cuda-nn`.

[145] Thomas Müller et al. "Instant neural graphics primitives with a multiresolution hash encoding". In: *arXiv preprint arXiv:2201.05989* (2022).

[146] Krishna Murthy. *nerf-pytorch: A PyTorch re-implementation*. 2020. URL: `https://github.com/krrish94/nerf-pytorch`.

[147] Armin Mustafa et al. "Temporally coherent general dynamic scene reconstruction". In: *IJCV* (2021).

[148] Arsha Nagrani and Andrew Zisserman. "From Benedict Cumberbatch to Sherlock Holmes: Character identification in TV series without a script". In: *BMVC*. 2017.

[149] Evonne Ng et al. "Body2Hands: Learning to infer 3D hands from conversational gesture body dynamics". In: *CVPR*. 2021.

[150] Thu Nguyen-Phuoc, Feng Liu, and Lei Xiao. "Snerf: stylized neural implicit representations for 3d scenes". In: *SIGGRAPH*. 2022.

[151] Alexander Quinn Nichol and Prafulla Dhariwal. "Improved denoising diffusion probabilistic models". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8162–8171.

[152] Michael Niemeyer et al. "Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5480–5490.

[153] 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*. 2021, pp. 5589–5599.

[154] Linfei Pan et al. "Global structure-from-motion revisited". In: *European Conference on Computer Vision*. Springer. 2024, pp. 58–77.

[155] Jeong Joon Park et al. "Deepsdf: Learning continuous signed distance functions for shape representation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 165–174.

[156] Keunhong Park et al. "Nerfies: Deformable neural radiance fields". In: *ICCV*. 2021.

[157] Omkar M Parkhi et al. "Automated video face labelling for films and TV material". In: *PAMI* (2018).

[158] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[159] Alonso Patron-Perez et al. "Structured learning of human interactions in TV shows". In: *PAMI* (2012).

[160] Georgios Pavlakos, Jitendra Malik, and Angjoo Kanazawa. "Human Mesh Recovery from Multiple Shots". In: *CVPR*. 2022.

[161] Georgios Pavlakos* et al. "The One Where They Reconstructed 3D Humans and Environments in TV Shows". In: *ECCV*. 2022.

[162] William Peebles and Saining Xie. "Scalable diffusion models with transformers". In: *ICCV*. 2023, pp. 4195–4205.

[163] Ryan Po et al. "State of the art on diffusion models for visual computing". In: *arXiv preprint arXiv:2310.07204* (2023).

[164] Polycam. *LiDAR & 3D Scanner for iPhone & Android*. 2024. URL: `https://poly.cam`.

[165] Ben Poole et al. "Dreamfusion: Text-to-3d using 2d diffusion". In: *arXiv preprint arXiv:2209.14988* (2022).

[166] Albert Pumarola et al. "D-nerf: Neural radiance fields for dynamic scenes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10318–10327.

[167] Chen Quei-An. *Nerf_pl: a pytorchlightning implementation of NeRF*. 2020. URL: `https://github.com/kwea123/nerf_pl/`.

[168] Jonathan Ragan-Kelley et al. "Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines". In: *Acm Sigplan Notices* 48.6 (2013), pp. 519–530.

[169] Amit Raj et al. "DreamBooth3D: Subject-Driven Text-to-3D Generation". In: *ICCV*. 2023.

[170] Nikhila Ravi et al. "Accelerating 3d deep learning with pytorch3d". In: *arXiv preprint arXiv:2007.08501* (2020).

[171] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. "Generating diverse high-fidelity images with vq-vae-2". In: *Advances in neural information processing systems* 32 (2019).

[172] Adria Recasens et al. "Following gaze in video". In: *ICCV*. 2017.

[173] Jeremy Reizenstein et al. "Common Objects in 3D: Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction". In: *International Conference on Computer Vision*. 2021.

[174] Konstantinos Rematas et al. "Soccer on your tabletop". In: *CVPR*. 2018.

[175] Davis Rempe et al. "Contact and human dynamics from monocular video". In: *ECCV*. 2020.

[176] Davis Rempe et al. "HuMoR: 3D Human Motion Model for Robust Pose Estimation". In: *ICCV*. 2021.

[177] Jiawei Ren et al. "L4gm: Large 4d gaussian reconstruction model". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 56828–56858.

[178] Edgar Riba et al. "Kornia: an open source differentiable computer vision library for pytorch". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 3674–3683.

[179] Chris Rockwell, David F Fouhey, and Justin Johnson. "Pixelsynth: Generating a 3d-consistent experience from a single image". In: *ICCV*. 2021.

[180] Barbara Roessle et al. "GANeRF: Leveraging Discriminators to Optimize Neural Radiance Fields". In: *SIGGRAPH Asia*. 2023.

[181] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *CVPR*. 2022.

[182] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, pp. 234–241.

[183] Darius Rückert et al. "NeAT: Neural Adaptive Tomography". In: *ACM Trans. Graph.* 41.4 (July 2022). ISSN: 0730-0301. URL: https://doi.org/10.1145/3528223.3530121.

[184] Nataniel Ruiz et al. "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation". In: *CVPR*. 2023.

[185] Sara Sabour et al. "RobustNeRF: Ignoring Distractors with Robust Losses". In: *arXiv preprint arXiv:2302.00833* (2023).

[186] Sara Sabour et al. "SpotlessSplats: Ignoring Distractors in 3D Gaussian Splatting". In: *ECCV Workshops*. 2024.

[187] Kyle Sargent et al. "ZeroNVS: Zero-Shot 360-Degree View Synthesis from a Single Real Image". In: *CVPR*. 2024.

[188] Paul-Edouard Sarlin et al. "Superglue: Learning feature matching with graph neural networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 4938–4947.

[189] Mattia Savardi et al. "CineScale: A dataset of cinematic shot scale in movies". In: *Data in Brief* 36 (2021), p. 107002.

[190] Mattia Savardi et al. "Shot scale analysis in movies by convolutional neural networks". In: *ICIP*. 2018.

[191] Manolis Savva et al. "PiGraphs: learning interaction snapshots from observations". In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), pp. 1–12.

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

[193] Johannes L Schönberger et al. "Pixelwise view selection for unstructured multi-view stereo". In: *ECCV*. 2016.

[194] Johannes Lutz Schönberger et al. "A Vote-and-Verify Strategy for Fast Spatial Verification in Image Retrieval". In: *Asian Conference on Computer Vision (ACCV)*. 2016.

[195] Junyoung Seo et al. "GenWarp: Single Image to Novel Views with Semantic-Preserving Generative Warping". In: *NeurIPS*. 2024.

[196] Yichun Shi et al. "MVDream: Multi-view Diffusion for 3D Generation". In: *ICLR*. 2024.

[197] Meng-Li Shih et al. "ExtraNeRF: Visibility-Aware View Extrapolation of Neural Radiance Fields with Diffusion Models". In: *CVPR*. 2024.

[198] Soshi Shimada et al. "Neural Monocular 3D Human Motion Capture with Physical Awareness". In: *ACM Transactions on Graphics (TOG)* 40.4 (2021), pp. 1–15.

[199] Soshi Shimada et al. "PhysCap: Physically plausible monocular 3D motion capture in real time". In: *ACM Transactions on Graphics (TOG)* 39.6 (2020), pp. 1–16.

[200] Anthony Simeonov et al. "Neural Descriptor Fields: SE(3)-Equivariant Object Representations for Manipulation". In: *ICRA*. 2022, pp. 6394–6400. URL: https://doi.org/10.1109/ICRA46639.2022.9812146.

[201] Josef Sivic, Mark Everingham, and Andrew Zisserman. ""Who are you?" – Learning person specific classifiers from video". In: *CVPR*. 2009.

[202] Harrison Jesse Smith et al. "A Method for Animating Children's Drawings of the Human Figure". In: *ACM Trans. Graph.* 42.3 (June 2023). ISSN: 0730-0301. DOI: 10.1145/3592788. URL: https://doi.org/10.1145/3592788.

[203] Noah Snavely, Steven M Seitz, and Richard Szeliski. "Photo tourism: exploring photo collections in 3D". In: *ACM siggraph 2006 papers*. 2006, pp. 835–846.

[204] Jascha Sohl-Dickstein et al. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 2256–2265.

[205] Yang Song and Stefano Ermon. "Generative modeling by estimating gradients of the data distribution". In: *Advances in neural information processing systems* 32 (2019).

[206] Olga Sorkine and Marc Alexa. "As-rigid-as-possible surface modeling". In: *Symposium on Geometry processing*. Vol. 4. Citeseer. 2007, pp. 109–116.

[207] Peter Sturm. "A historical survey of geometric computer vision". In: *International Conference on Computer Analysis of Images and Patterns (CAIP)*. 2011. DOI: 10.1007/978-3-642-23672-3_1.

[208] Cheng Sun, Min Sun, and Hwann-Tzong Chen. "Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5459–5469.

[209] Jiaming Sun et al. "LoFTR: Detector-free local feature matching with transformers". In: *CVPR*. 2021.

[210] Jingxiang Sun et al. "DreamCraft3D: Hierarchical 3D Generation with Bootstrapped Diffusion Prior". In: *arXiv*. 2023.

[211] Roman Suvorov et al. "Resolution-robust Large Mask Inpainting with Fourier Convolutions". In: *WACV*. 2022.

[212] Towaki Takikawa et al. *Kaolin Wisp: A PyTorch Library and Engine for Neural Fields Research*. https://github.com/NVIDIAGameWorks/kaolin-wisp. 2022.

[213] Matthew Tancik et al. "Block-NeRF: Scalable Large Scene Neural View Synthesis". In: *arXiv* (2022).

[214] Matthew Tancik et al. "Nerfstudio: A Modular Framework for Neural Radiance Field Development". In: *arXiv preprint arXiv:2302.04264* (2023).

[215] Luming Tang et al. "Realfill: Reference-driven generation for authentic image completion". In: *arXiv*. 2023.

[216] Shitao Tang et al. "MVDiffusion: Enabling Holistic Multi-view Image Generation with Correspondence-Aware Diffusion". In: *NeurIPS*. 2023.

[217] Makarand Tapaswi, Marc T Law, and Sanja Fidler. "Video face clustering with unknown number of clusters". In: *ICCV*. 2019.

[218] Ayush Tewari et al. "Diffusion with forward models: Solving stochastic inverse problems without direct supervision". In: *NeurIPS*. Vol. 36. 2023, pp. 12349–12362.

[219] Javier Tirado-Garín, Frederik Warburg, and Javier Civera. "DAC: Detector-Agnostic Spatial Covariances for Deep Local Features". In: *arXiv preprint arXiv:2305.12250* (2023).

[220] Edgar Tretschk et al. "Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12959–12970.

[221] Richard Tucker and Noah Snavely. "Single-view View Synthesis with Multiplane Images". In: *CVPR*. June 2020.

[222] Matias Turkulainen et al. "DN-Splatter: Depth and Normal Priors for Gaussian Splatting and Meshing". In: *WACV*. 2025.

[223] Michał J Tyszkiewicz, Pascal Fua, and Eduard Trulls. "DISK: Learning local features with policy gradient". In: *NeurIPS*. 2020.

[224] Andrea Vallone et al. "Danish airs and grounds: A dataset for aerial-to-street-level place recognition and localization". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 9207–9214.

[225] Aaron Van Den Oord, Oriol Vinyals, et al. "Neural discrete representation learning". In: *Advances in neural information processing systems* 30 (2017).

[226] Basile Van Hoorick et al. "Generative Camera Dolly: Extreme Monocular Dynamic Novel View Synthesis". In: *ECCV*. 2024.

[227] Ashish Vaswani et al. "Attention is all you need". In: *NeurIPS*. 2017.

[228] Dor Verbin et al. "NeRF-Casting: Improved View-Dependent Appearance with Consistent Reflections". In: *SIGGRAPH Asia*. 2024.

[229] Dor Verbin et al. "Ref-nerf: Structured view-dependent appearance for neural radiance fields". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2022, pp. 5481–5490.

[230] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. "Anticipating visual representations from unlabeled video". In: *CVPR*. 2016.

[231] Can Wang et al. "Clip-nerf: Text-and-image driven manipulation of neural radiance fields". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 3835–3844.

[232] Dongqing Wang et al. "InpaintNeRF360: Text-Guided 3D Inpainting on Unbounded Neural Radiance Fields". In: *arXiv*. 2023.

[233] Guangcong Wang et al. "Sparsenerf: Distilling depth ranking for few-shot novel view synthesis". In: *ICCV*. 2023.

[234] Haochen Wang et al. "Score Jacobian Chaining: Lifting Pretrained 2D Diffusion Models for 3D Generation". In: *CVPR*. 2023.

[235] Peng Wang and Yichun Shi. "Imagedream: Image-prompt multi-view diffusion for 3d generation". In: *arXiv preprint arXiv:2312.02201* (2023).

[236] Peng Wang et al. "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction". In: *arXiv preprint arXiv:2106.10689* (2021).

[237] Qianqian Wang et al. "Continuous 3D Perception Model with Persistent State". In: *arXiv preprint arXiv:2501.12387* (2025).

[238] Qianqian Wang et al. "Ibrnet: Learning multi-view image-based rendering". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4690–4699.

[239] Ruicheng Wang et al. "MoGe: Unlocking Accurate Monocular Geometry Estimation for Open-Domain Images with Optimal Training Supervision". In: *arXiv preprint arXiv:2410.19115* (2024).

[240] Shuzhe Wang et al. "DUSt3R: Geometric 3D vision made easy". In: *CVPR*. 2024, pp. 20697–20709.

[241] Xiaolong Wang, Rohit Girdhar, and Abhinav Gupta. "Binge watching: Scaling affordance learning from sitcoms". In: *CVPR*. 2017.

[242] Zhengyi Wang et al. "ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation". In: *NeurIPS*. 2023.

[243] Zhouxia Wang et al. "MotionCtrl: A unified and flexible motion controller for video generation". In: *SIGGRAPH*. 2024.

[244] Zirui Wang et al. "NeRF–: Neural radiance fields without known camera parameters". In: *arXiv preprint arXiv:2102.07064* (2021).

[245] Frederik Warburg, Michael Ramamonjisoa, and Manuel López-Antequera. "SparseFormer: Attention-based Depth Completion Network". In: *arXiv preprint arXiv:2206.04557* (2022).

[246] Frederik Warburg et al. "Nerfbusters: Removing ghostly artifacts from casually captured NeRFs". In: *ICCV*. 2023, pp. 18120–18130.

[247] Frederik Warburg et al. "Self-Supervised Depth Completion for Active Stereo". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 3475–3482.

[248] Daniel Watson et al. "Novel view synthesis with diffusion models". In: *arXiv*. 2022.

[249] Katherine Watson et al. *Creating workflows for NeRF Portraiture*. 2022. URL: `https://rd.nytimes.com/projects/creating-workflows-for-nerf-portraiture`.

[250] Ethan Weber et al. "Nerfiller: Completing scenes via generative 3D inpainting". In: *CVPR*. 2024, pp. 20731–20741.

[251] Silvan Weder et al. "Removing Objects From Neural Radiance Fields". In: *CVPR*. 2023.

[252] Zhenzhen Weng and Serena Yeung. "Holistic 3D Human and Scene Mesh Estimation from Single View Images". In: *CVPR*. 2021.

[253] Olivia Wiles et al. "Synsin: End-to-end view synthesis from a single image". In: *CVPR*. 2020.

[254] Jay Zhangjie Wu et al. "Difix3D+: Improving 3D Reconstructions with Single-Step Diffusion Models". In: *arXiv preprint arXiv:2503.01774* (2025).

[255] Jay Zhangjie Wu et al. "Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation". In: *ICCV*. 2023.

[256] Rundi Wu et al. "ReconFusion: 3D reconstruction with diffusion priors". In: *CVPR*. 2024, pp. 21551–21561.

[257] Jamie Wynn and Daniyar Turmukhambetov. "Diffusionerf: Regularizing neural radiance fields with denoising diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 4180–4189.

[258] Kevin Xie et al. "Physics-based Human Motion Estimation and Synthesis from Videos". In: *ICCV*. 2021.

[259] Yiheng Xie et al. "Neural fields in visual computing and beyond". In: *Computer Graphics Forum*. Vol. 41. 2. Wiley Online Library. 2022, pp. 641–676.

[260] Xiang Xu et al. "D3D-HOI: Dynamic 3D Human-Object Interactions from Videos". In: *arXiv preprint arXiv:2108.08420* (2021).

[261] Yinghao Xu et al. "GRM: Large Gaussian reconstruction model for efficient 3D reconstruction and generation". In: *ECCV*. 2024.

[262] Lihe Yang et al. "Depth Anything V2". In: *arXiv preprint arXiv:2406.09414* (2024).

[263] Yi Yang and Deva Ramanan. "Articulated human detection with flexible mixtures of parts". In: *PAMI* (2012).

[264] Lior Yariv et al. "Volume rendering of neural implicit surfaces". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 4805–4815.

[265] Vickie Ye et al. "gsplat: An open-source library for Gaussian splatting". In: *Journal of Machine Learning Research* 26.34 (2025), pp. 1–17.

[266] Lin Yen-Chen. *NeRF-pytorch*. `https://github.com/yenchenlin/nerf-pytorch/`. 2020.

[267] Chandan Yeshwanth et al. "ScanNet++: A high-fidelity dataset of 3D indoor scenes". In: *ICCV*. 2023, pp. 12–22.

[268] Alex Yu et al. "pixelnerf: Neural radiance fields from one or few images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4578–4587.

[269] Alex Yu et al. "Plenoctrees for real-time rendering of neural radiance fields". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5752–5761.

[270] Jason J Yu et al. "Long-term photometric consistent novel view synthesis with diffusion models". In: *ICCV*. 2023, pp. 7094–7104.

[271] Wangbo Yu et al. "ViewCrafter: Taming video diffusion models for high-fidelity novel view synthesis". In: *arXiv preprint arXiv:2409.02048* (2024).

[272] Zehao Yu et al. *SDFStudio: A Unified Framework for Surface Reconstruction*. 2022. URL: `https://github.com/autonomousvision/sdfstudio`.

[273] Ye Yuan et al. "SimPoE: Simulated Character Control for 3D Human Pose Estimation". In: *CVPR*. 2021.

[274] Yan-Pei Cao Yue Luo. *ArcNerf: Nerf-based object/scene rendering and extraction framework*. 2022. URL: `https://github.com/TencentARC/arcnerf/`.

[275] Jason Y Zhang et al. "Cameras as rays: Pose estimation via ray diffusion". In: *ICLR*. 2024.

[276] Jason Y Zhang et al. "Perceiving 3D human-object spatial arrangements from a single image in the wild". In: *ECCV*. 2020.

[277] Kai Zhang et al. "Arf: Artistic radiance fields". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXI*. Springer. 2022, pp. 717–733.

[278] Richard Zhang et al. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *CVPR*. 2018.

[279] Siwei Zhang et al. "Learning Motion Priors for 4D Human Body Capture in 3D Scenes". In: *ICCV*. 2021.

[280] Yinda Zhang and Thomas Funkhouser. "Deep depth completion of a single rgb-d image". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 175–185.

[281] Chao Zhou et al. "Unsupervised learning of stereo matching". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1567–1575.

[282] Xiaowei Zhou et al. "MonoCap: Monocular human motion capture using a CNN coupled with a geometric prior". In: *PAMI* (2018).

[283] Zhizhuo Zhou and Shubham Tulsiani. "Sparsefusion: Distilling view-conditioned diffusion for 3d reconstruction". In: *CVPR*. 2023.

[284] Jun-Yan Zhu, Yong Jae Lee, and Alexei A Efros. "Averageexplorer: Interactive exploration and alignment of visual data collections". In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), pp. 1–11.

[285] Luyang Zhu et al. "Reconstructing NBA players". In: *ECCV*. 2020.

[286] Yukun Zhu et al. "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 19–27.

[287] Zihan Zhu et al. "NICE-SLAM: Neural Implicit Scalable Encoding for SLAM". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022.

[288] Chen Ziwen et al. "Long-LRM: Long-sequence Large Reconstruction Model for Wide-coverage Gaussian Splats". arXiv:2410.12781. 2024.

Appendix A

Project Websites and Supplementary Material

Below is a list of project websites and supplementary material relevant to the chapters in this thesis. These pages contain additional results, videos, code, datasets, and interactive demos.

- **Nerfstudio (Chapter 2):** `https://nerf.studio`
- **Nerfbusters (Chapter 3):** `https://ethanweber.me/nerfbusters`
- **Nerfiller (Chapter 4):** `https://ethanweber.me/nerfiller`
- **Fillerbuster (Chapter 5):** `https://ethanweber.me/fillerbuster`
- **Sitcoms3D (Chapter 6):** `https://ethanweber.me/sitcoms3D`
- **Toon3D (Chapter 7):** `https://toon3d.studio`

You can also find more information and references to all of these projects on my personal website: `https://ethanweber.me`.