

ABSTRACT

Title of Dissertation: **FLEXIBLE INPUT FOR
NOVEL VIEW SYNTHESIS**

Geng Lin
Doctor of Philosophy, 2025

Dissertation Directed by: **Professor Matthias Zwicker**
Department of Computer Science

Synthesizing novel views of a scene from a limited number of input images is a long-standing problem. It has enormous practical applications, such as virtual museums, online meetings, and sports event streaming. The emergence of virtual reality technology has also made it easier for people to observe and interact with virtual environments, highlighting the needs in robust and efficient view synthesis.

In recent years, this field of research has seen large advances in neural radiance fields (NeRFs) and 3D Gaussian Splatting (3DGS). Although these methods achieve efficient high-quality view synthesis, they require careful and dense capturing in controlled environments. Therefore, much research effort has been made to relax the requirements, like allowing dynamic scenes and handling variance in lighting. This thesis presents our work along this path of enabling novel view synthesis from challenging inputs. In particular, we focus on three scenarios.

First, we tackle the problem of unwanted foreground objects, like moving people or vehicles present in front of a building. As these objects cast shadows and reflections, naively masking

the objects leaves artifacts in background reconstruction. We propose a method to decompose foreground objects with their cast effects into separate 2D layers and a clean 3D background layer.

Second, we address view synthesis from very few inputs. With as few as three input views, we leverage recent developments in large image and video generation priors to interpolate in-between views for better supervising scene reconstruction methods. To improve both efficiency and quality, we use a feedforward geometry foundation model to obtain a dense point cloud that serves as condition images to the image priors. In addition, we introduce optimizable image warps and a robust view sampling strategy to deal with inconsistencies in generated images.

Lastly, we consider an extended problem of inverse rendering which decomposes the scene into geometry, material properties, and environment lighting. It not only enables synthesis of novel views via rendering, but also provides extended capacities like scene editing and relighting. We propose a simple capturing method by rotating the object several times when taking photos. With this setup, we show that the artifacts caused by ambiguity can be drastically reduced. We model the scene with 2D Gaussian primitives for computational efficiency and make use of a proxy geometry as well as a residual constraint to further improve handling of global illumination.

The works presented in this thesis improve the quality and robustness of novel view synthesis with challenging input data. Further research efforts can be made along the lines to enable casual capturing and lower the bars for creating and sharing digital content from the real world.

FLEXIBLE INPUT FOR
NOVEL VIEW SYNTHESIS

by

Geng Lin

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2025

Advisory Committee:

Professor Matthias Zwicker, Chair/Advisor
Professor Uzi Vishkin, Dean's Representative
Professor Thomas Goldstein
Professor Christopher Metzler
Professor Abhinav Shrivastava

© Copyright by
Geng Lin
2025

Acknowledgments

I would like to express my deepest gratitude to my advisor, Dr. Matthias Zwicker, for his continued guidance and support throughout my pursuit of research. We had countless meetings and discussions on research ideas and experiments. I benefited enormously from his knowledge and experience, and my achievements would not have been possible without his advice.

My appreciation also goes to the members of my dissertation committee, Dr. Uzi Vishkin, Dr. Thomas Goldstein, Dr. Abhinav Shrivastava, and Dr. Christopher Metzler, for their insightful comments and suggestions on this dissertation.

Big thanks to my internship mentors Xuejian Rong, Ayush Saraf, Jia-Bin Huang, Changil Kim, and Chen Gao. They broadened my view and shaped my good habits in research and communication.

Lastly, I am deeply grateful to my family for always being supportive and patient. I would not be as optimistic and mentally strong without their warm words. To my friend Shuhao Tan, for being a great roommate for years. To my collaborators and colleagues, for their intelligence and participation in my research.

Table of Contents

Acknowledgements	ii
Table of Contents	iii
List of Tables	vi
List of Figures	vii
List of Abbreviations	xii
Chapter 1: Introduction	1
1.1 Neural Radiance Fields	3
1.2 3D Gaussian Splatting	5
1.3 Inverse Rendering	7
1.4 Our Work	10
1.5 Contributions	14
1.5.1 Additional Works	15
Chapter 2: OmnimatteRF: Robust Omnimatte with 3D Background Modeling	18
2.1 Introduction	18
2.2 Related Work	21
2.3 Method	23
2.3.1 The OmnimatteRF Model	25
2.3.2 Optimizing the Model	26
2.3.3 Clean Background via Masked Retraining	29
2.4 Evaluation	32
2.4.1 The Movies Dataset	32
2.4.2 Experiment Setup	32
2.4.3 Implementation Details	34
2.4.4 Quantitative Evaluation	36
2.4.5 Qualitative Evaluation	36
2.4.6 Ablation Studies	40
2.4.7 Limitations	41
2.5 Conclusion	42
2.6 Additional Qualitative Results	42
2.7 Random Initialization	43
2.8 Additional Implementation Details	43

2.8.1	Mask Generation	43
2.8.2	Network Architecture	45
2.8.3	Hyper-parameters	45
2.8.4	Running Time Measurement	46
Chapter 3: Few View 3D Gaussians with Feed-Forward Geometry Model and Generative Priors		48
3.1	Introduction	48
3.2	Related Works	50
3.3	Method	54
3.3.1	Enhancing Sparse Input with Geometry Model and Generative Prior	55
3.3.2	Adaptive Warps	56
3.3.3	Color Adjustment Map	57
3.3.4	Losses	58
3.3.5	High Real View Ratio	59
3.4	Experiments	60
3.4.1	Implementation	60
3.5	SVD Model Training	61
3.5.1	Evaluation	61
3.5.2	SVD Model Comparison	61
3.5.3	Time Analysis	62
3.5.4	Ablation Study	63
3.5.5	Image Warp with Homography	63
3.5.6	Diffusion Model for Post Processing	65
3.6	Limitations	67
3.7	Conclusion	67
Chapter 4: Rotated Lights for Consistent and Efficient 2D Gaussians Inverse Rendering		70
4.1	Introduction	70
4.2	Related Works	72
4.2.1	Novel View Synthesis	72
4.2.2	Inverse Rendering	74
4.3	Background	75
4.3.1	3D Gaussian Splatting	75
4.3.2	2D Gaussian Splatting	77
4.3.3	Gaussian Splatting in Inverse Rendering	78
4.4	Method	79
4.4.1	RotLight Capturing	80
4.4.2	Neural Radiance Cache with Residual Constraint	81
4.4.3	Proxy Mesh for Incident Queries	84
4.4.4	Optimization Objective	85
4.5	Experiments	86
4.5.1	SynRotLight Dataset	86
4.5.2	Implementation	86
4.5.3	Material Recovery	86

4.5.4	Ablation Study	87
4.5.5	Residual Constraint	88
4.5.6	Mesh Proxy Geometry	89
4.5.7	Relighting	89
4.5.8	Real World Examples	89
4.6	Limitations	90
4.7	Conclusion	90
Chapter 6:	Conclusions	100
6.1	Limitations	101
6.2	Future Work	102
Bibliography		104

List of Tables

2.1	Quantitative evaluations. We present the background reconstruction comparison of our method and baselines on the <code>Kubrics</code> and <code>Movies</code> datasets. Best results are in bold and second place are <u>underlined</u> . Results marked - are the ones the method failed to give good separations (visuals in supplementary).	34
2.2	Hyper-parameters. We document the hyper-parameters (number of steps and weights of loss terms) in our experiments. The first row is the configuration shared by most videos. Remaining rows are videos with different configurations, and - means unchanged from the shared number.	46
2.3	Running Time Measurement. We measure and compare the time it takes to train <code>OmnimatteRF</code> and baseline methods. Top: <code>Movies</code> , <code>Wild</code> (480×270 , <code>DAVIS</code> has a similar resolution of 428×240). Bottom: <code>Kubrics</code> (256×256).	46
3.1	Sparse view synthesis results on the <code>DL3DV</code> dataset. Our method achieves on-par results with state-of-the-art. We additionally trains an SVD model following the setup in <code>3DGS-Enhancer</code> and report the result of using this model in our pipeline in the last row.	60
3.2	Sparse view synthesis results on the <code>Mip-NeRF 360</code> dataset.	60
3.3	Comparison of SVD models with different conditions. We optimize an SVD model using the code and settings of <code>3DGS-Enhancer</code> [71] and compare it with ours. The metrics are computed for all interpolated training views between SVD outputs and ground truths. In addition, the last row computes the metrics with our warped and color-adjusted images. The processed images are more faithful to ground truth, meaning that they are better training inputs than SVD outputs. . .	62
3.4	Time analysis. All numbers are seconds, averaged over the 9-view <code>DL3DV</code> scenes. Initial model is a coarse <code>3DGS</code> model for <code>3DGS-Enhancer</code> , and <code>VGGT</code> inference for ours. SVD inference is measured on an <code>H100</code> GPU, while all other steps are measured on a <code>RTX A5000</code> GPU.	62
3.5	Ablation study. We disable the components in our pipeline one at a time to demonstrate their effectiveness. In addition, we experiment with a homography-based warping instead of the proposed grid sampling offset method.	63
4.1	Quantitative comparison of material estimation. All experiments are conducted in linear color space. Top: our method outperforms prior works by a large margin thanks to the capturing setup. Bottom: ablation studies show that our method gains the most improvement from <i>RotLight</i>	87

List of Figures

1.1	NeRF pipeline illustration from the original work [79]. (a), (b) to render a ray, multiple locations are sampled along the ray. For each sample, a neural network is queried with its location (x, y, z) and ray direction (θ, ϕ) to produce radiance field value RGB and density field value σ . (c), (d) the sampled radiance and density are integrated with volumetric rendering to obtain ray (pixel) color. The color is compared with ground truth from input image to optimize the neural network’s parameters.	4
1.2	Left: visualization of projecting 2D Gaussians from 3D space to image (“splatting”). This can be implemented very efficiently for both 3D and 2D Gaussian primitives. Right: why 2D Gaussians is a more accurate depth representation than 3D Gaussians. Images taken from [51].	5
1.3	Pipeline of IRGS, a state-of-the-art two-stage method to use 2D Gaussians for inverse rendering. In the first stage, a 2DGS model is optimized for the scene to represent scene geometry and radiance. The 2D Gaussians are attached albedo and roughness values to be rendered with splatting. In the second stage, the rendering equation is used to render images from the geometry, radiance and material properties represented by the 2DGS model. The parameters are optimized to minimize the loss between renderings and ground truth images. Taken from IRGS [39].	8
1.4	In Chapter 2, we solve the problem of decoupling foreground objects with their associated effects like shadows from background.	10
1.5	In Chapter 3, we handle view synthesis from very sparse inputs. These are examples of having only 3 input views.	11
1.6	In Chapter 4, we reduce ambiguity in inverse rendering. In this example, IRGS appears more blue on the left and red on the right due to ambiguity of light. . . .	13
2.1	Video with parallax effects. Limited by their 2D image representation (a), previous works such as Omnimate fail to handle videos with parallax effects in the background. Their foreground layer (b) has to capture (dis)occlusion effects to minimize the reconstruction loss. In contrast, our method employs a 3D background (c), enabling us to obtain clean foreground layers (d).	19

2.2	Method overview. We propose a video matting method, named OmnimateRF, which combines 2D foreground layers with a 3D background layer. The foreground branch (f_{fg} , in green box) predicts an RGBA layer (C_t^i, α_t^i) for each object, and an auxiliary flow output ($\hat{F}_t^i$). The background branch (f_{bg} , in yellow box) produces a background layer with depths ($B_t, \hat{D}_t$). Optimization. During training, predicted colors ($\hat{I}_t$) and flow ($\hat{F}_t$) are alpha-composited, whose inputs have red and green borders respectively. The right most column illustrates the data terms in the loss function, and we omit the regularization terms in this illustration.	24
2.3	Background Layer Training Signals. We illustrate how the training signal to the background layer changes over time. It explains why the background captures some of the associated effects (in this example, shadows). We use the pixel circled in red as an example. (a) At the beginning of training, the foreground alpha value (in light green) does not include the shadow. Therefore, α is small and at this pixel, $\hat{I}_t(x, y) \approx B_t(x, y)$. The reconstruction loss $\mathcal{L}_{recons}$ encourages the background network f_{bg} to produce dark prediction at this location from this viewing angle. (b) As training progresses, α gets larger in the shadow region, and $\hat{I}_t(x, y) \approx C_t^i(x, y)$. This means that f_{bg} receives little to no supervision signals from this pixel. If it has modeled the shadow in some ways (in this case, a hole), it has little incentive to remove it, leaving the artifact in (c).	30
2.4	Background Reconstruction. We show examples of results presented in quantitative evaluations. For videos with parallax effects, 3D methods like D ² NeRF and ours reconstruct less distorted background than Omnimate and LNA.	35
2.5	Qualitative comparison (background). We compare results of our and baseline methods on videos from each dataset. The first two videos are synthetic from Kubrics and Movies, followed by three Wild videos. Omnimate fails to handle objects in 3D and produces distorted background. D ² NeRF works for videos with appropriate hyper-parameters, but does not generalize to new videos easily. Our method handles videos in many different settings. Due to space constraint we defer LNA results to the supplementary. Due to space constraint, the foreground layers are visualized in Figure 2.6.	37
2.6	Qualitative comparison (foreground).	38
2.7	Loss Term Ablations. Background of real-world videos without $\mathcal{L}_{depth}$ and foreground without $\mathcal{L}_{flow}$ can be degraded for real-world videos.	39
2.8	Clean Background Retraining. Background layers jointly trained can capture the shadows as a hole on the ground (a-c). After the joint training, the foreground omnimate provides a better mask that can be used to train a clean background (d-f).	40
2.9	Limitations. (a), (b): Background can have baked-in shadows when the region is covered by shadows for most frames of the video. (c): The foreground layer captures irrelevant object motions (middle left) in the background. Best viewed in videos.	41

2.10	Effect of random initialization. Top: for the <code>Wild/dogwalk</code> video, different seeds lead to different amount of hallucinated shadow of the person. Bottom: for the <code>Kubrics/cars</code> video, seeds influence how shadows are associated to the objects.	44
2.11	D²NeRF results for failed scenes.	47
3.1	We propose a scene reconstruction pipeline from sparse inputs. It leverages feed-forward geometry models and generative priors to densify the input views with pseudo ground truth images from interpolated views. With approaches to handle inconsistencies from generated views, we our method achieves high quality reconstruction.	49
3.2	Overview of our pipeline. Left: we first densify the sparse input images. A feed-forward geometry model is used to estimate a dense point cloud, which is then rendered from interpolated views. The point cloud renderings are used to condition a video diffusion model that generates dense pseudo ground truths. Right: we initialize 3DGS from the dense point cloud, and optimize it with both input and generated views. We employ a sampling strategy biased towards real views, as well as learnable image warps and color adjustments to handle the inconsistencies in generated views.	55
3.3	Adaptive warping illustration. The figure is an example of a warp grid with resolution $r_{\text{warp}} = 3$, in which $3 \times 3 = 9$ control points are optimized. Left: initialization with all offsets set to zeros. Right: a possible outcome with some control points moved and the image distorted.	56
3.4	Effects of proposed warp and color adjustment. We compare SVD output I_i (2 nd row) with the warped and color-adjusted image I_i^{adj} (4 th row) after training. For better visualization we also show their absolute error on the 3 rd and 5 th rows. The high error regions at the boundary of the buildings and objects due to geometric inconsistency in the SVD output are effectively mitigated in the processed images.	64
3.5	Real view ratio analysis. Increasing the ratio of real inputs improves reconstruction results until $\sigma_{\text{real}} = 0.8$. The smallest ratio $\sigma_{\text{real}} = 0.03$ corresponds to not using the proposed sampling strategy (i.e. the ratio between input and generated images is approximately 1 to 30), and $\sigma_{\text{real}} = 1$ means not using generated images at all. The results are from the DL3DV 9-view test set.	65
3.6	Qualitative results of 3 input views. The first two columns compare the two initial models: 3DGS from sparse views and dense point cloud from VGGT. With very sparse inputs, VGGT produces more coherent results than 3DGS. The 3 rd column shows our fine-tuned SVD model output conditioned with the VGGT point cloud renderings. Our 3DGS model trained with the densified images achieves high quality reconstruction (4 th column).	66
3.7	Qualitative results of 6 input views.	68
3.8	Qualitative results of 9 input views.	69

4.1	Overview of our pipeline. RotLight : we capture the same object with $K = 3$ different rotations. Stage 1 : we pretrain a 2DGS model with a set of images under the same lights (e.g. ϕ_1) to obtain an initial geometry reconstruction of the scene. Stage 2 : we obtain pixel material, depth and surface normal by splatting Gaussians. The depth and normal are used to query incident lights, with direct illumination from the environment map $\mathbf{E}_{\phi_k}$ (Section 4.4.1), and indirect illumination from a radiance cache (Section 4.4.2). The radiance cache is further supervised with a residual constraint evaluated at random surface points. For accurate incident sampling, we also use a proxy mesh instead of ray tracing the Gaussians (Section 4.4.3).	80
4.2	Analysis of residual constraint. In the above example, consider the albedo estimation of point $\mathbf{x}$. (a) when querying incident radiance $L_i(x, \omega_i)$, we show three possibilities of the sampled directions. ω_3 does not hit scene geometry and queries the environment. ω_1 and ω_2 hits the scene at $\mathbf{y}_1, \mathbf{y}_2$ respectively, and will query the radiance cache $\mathbf{R}_k$. Note that $\mathbf{y}_1$ can be observed from several cameras, so the cached radiance $\mathbf{R}_k(\mathbf{y}_1, -\omega_1)$ may be relatively accurate, in contrast to $\mathbf{R}_k(\mathbf{y}_2, -\omega_2)$. (b) in this case, the residual constraint can be useful to correct $\mathbf{R}_k(\mathbf{y}_2, -\omega_2)$ by rendering $L_o(\mathbf{y}_2, -\omega_2)$ with Equation 4.13, which in turns queries incident radiance from more directly observable regions (like $\mathbf{x}$ and $\mathbf{y}_1$) towards $\mathbf{y}_2$. As a result, the cached value $\mathbf{R}_k(\mathbf{y}_2, -\omega_2)$ will be more accurate, which benefits the material estimation of $\mathbf{x}$	87
4.3	Examples from our <i>RotLight</i> dataset. The three rows are light rotations at 0, 120 and 240 degrees respectively.	88
4.4	Qualitative results of albedo estimation. Thanks to our <i>RotLight</i> setup, our method is able to better factor out environment illumination from albedo, resulting in more consistent and accurate estimations. Numbers below images are PSNR and SSIM metrics.	91
4.5	Qualitative results of roughness estimation. Our method is better in <i>counter</i> and similar to IRGS in other scenes. However, roughness estimation still remains challenging overall.	92
4.6	Qualitative results of environment light estimation. Top to bottom: <i>counter</i> , <i>toy</i> , <i>table</i> , <i>hotdog</i> . Note that the problem is still extremely ill-posed without direct observations from very specular surfaces. Our method give overall closer results, such as the color of the sky.	93
4.7	Ablation study. Even rows show error maps. Additional examples shown in Figure 4.8. Without residual constraint, the estimated albedo has more baked shadow where incident illumination presents from locations not directly visible from the camera. <i>counter</i> : under the chopping board, between the red bag and blue scale.	94
4.8	Ablation study of more scenes. Baked shadows in albedo can be found without residual constraint. <i>toy</i> : under the helicopter and dog. <i>table</i> : between the books. <i>hotdog</i> : under the front end of the hotdog.	95
4.9	Ambient occlusion rendering. With the proxy mesh, our ambient occlusion rendering is smoother and more accurate.	96

4.10 Relighting comparison. We demonstrate relighting results using our recovered scene parameters and compare them with IRGS. Our method produces colors that are closer to ground truth.	97
4.11 Examples of the captured real-world videos, <i>box</i> and <i>desk</i>	97
4.12 Albedo recovery of real-world data. Thanks to RotLight capture, our recovered albedo maps have less baked shadows.	98
4.13 Known geometry experiments. The first two columns are experiments run with known (GT) geometry, with single light and RotLight setups respectively. Column 1, 2: we show that the RotLight capture setup and other proposed components is effective with known geometry. Column 2, 3: the performance of our method is limited by inaccuracies in geometry, such as under the onions in <i>counter</i> , under the dog in <i>toy</i> , the top of the book pile in <i>table</i> , and under the hotdogs in <i>hotdog</i>	99

List of Abbreviations

MLP	Multi-layer Perceptron
MVS	Multi-view Stereo
NeRF	Neural Radiance Fields
2DGS	2D Gaussian Splatting
3DGS	3D Gaussian Splatting

Chapter 1: Introduction

Replicating the physical world in the digital space is one of the many ultimate goals of computer graphics and vision. As virtual reality headsets become increasingly affordable and popular, the applications of digital representations are expanding to unprecedented landscapes. For example, in the future, museums may be able to create copies of their collections and welcome virtual visitors. Imagine being able to visit a museum thousands of miles away by putting on a pair of VR glasses, and immediately ancient artifacts appear in front of us and can be observed from all angles just like the real world. We can even touch and move them, which would be prohibited in physical reality, to get better close-up views. At the end of a tour, we can bring virtual copies of the objects back to decorate our own virtual space. On another day, we may invite our foreign friends to the space and show them the art and design, while everyone has a realistic avatar that makes the virtual gathering feel as natural as a physical one. In addition to virtual tourism and gathering, similar technologies can also enable remote medical operations, engineering collaborations, online sport events, and much more.

The key technology challenge to enable this future is novel view synthesis. It focuses on the problem of generating new pictures of a scene from some given observations like images and videos. In the above scenario, the museum would create digital representations of their collections that can be easily transmitted to virtual visitors. The VR device will then *synthesize* images with

the models based on the user’s interaction. For using the object in another virtual space, the models need to support *relighting* to properly change their appearances based on the lighting environment. These applications pose a need for efficient and accurate novel view synthesis methods.

Being a long-standing problem, novel view synthesis remains an active field of research. Traditionally, view synthesis is achieved by multi-view stereo (MVS) [35, 98, 101]. The key idea is *correspondence*, where pixels from different images that correspond to the same 3D location are identified to collectively provide information for that location. With sufficient observations, novel view images can be produced by inferring the features at the 3D points that project to the view of interest. With the advancements of deep learning, image priors have been applied to image feature extraction, correspondence determination, and feature-to-image conversion [21, 40, 137, 138, 148, 154]. However, the problem remains difficult as it is extremely challenging to identify enough correspondences for complete reconstruction of objects. In addition, aggregated information from the corresponding points alone does not always ensure that the synthesized view matches the observations.

The greatest advances in recent years’ view synthesis research come from *differentiable rendering*. On a high level, the idea is to represent a scene with some parameters for a certain rendering algorithm that is differentiable. For every input camera pose, the parameters are rendered to produce an image. Then, the rendered result is compared with the input image, and the differences can be measured by a distance function. Finally, the gradient of the distance with respect to the scene parameters can be used to update them with gradient descent. This optimization process can be repeated for many iterations until the parameters converge to produce images that are close to all input images. The converged parameters can then be rendered from novel

camera poses and generate novel views of the scene.

Naturally, it is crucial to select a suitable parameterization of the scene along with a rendering method. Recent approaches mostly decompose a scene into a geometry field and a radiance field, with a differentiable volumetric rendering model for these fields. A geometry field can be evaluated anywhere in 3D and produces a scalar value that represents the density of the scene at this location. Similarly, a radiance field is evaluated with arbitrary combination of 3D location and view direction, and the result is the radiance in the scene from the location towards the queried direction. The density and radiance values are converted to opacity and color along the rays to produce pixel colors. Early explorers used discrete plane-based geometry representations, where the radiance of the scene is recorded on the planes [46, 78, 155]. Later, neural radiance fields (NeRFs) [79] and 3D Gaussian Splatting [57] proposed two different approaches to represent a continuous geometry field. They brought significant improvements in quality and speed of view synthesis and have had a huge impact in the field.

1.1 Neural Radiance Fields

NeRF [79] models a scene as a density field and a direction-dependent radiance field, encoded by neural networks, typically multi-layer perceptrons (MLPs). It composes an image by volumetric rendering. For every pixel, a ray is traced from the camera into the scene and a fixed number of sampled locations are chosen on the ray. An illustration borrowed from NeRF [79] can be found in Figure 1.1. At every location, the density MLP is queried to determine its volume density, and the radiance MLP is queried to find its color towards the camera. The colors of all samples are then composited with front-to-back alpha blending, where the alpha values are calcu-

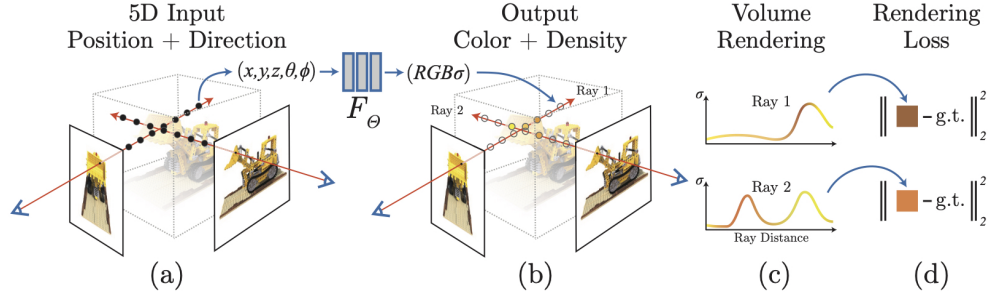


Figure 1.1: NeRF pipeline illustration from the original work [79]. **(a), (b)** to render a ray, multiple locations are sampled along the ray. For each sample, a neural network is queried with its location (x, y, z) and ray direction (θ, ϕ) to produce radiance field value RGB and density field value σ . **(c), (d)** the sampled radiance and density are integrated with volumetric rendering to obtain ray (pixel) color. The color is compared with ground truth from input image to optimize the neural network’s parameters.

lated from the densities. The optimization objective is to minimize image loss, which is defined as the mean squared distance between the rendered pixels and the ground-truth input images. The integration process is differentiable, and the gradients of the loss can be backpropagated to the MLPs, allowing stochastic gradient descent to update their parameters. In implementation, the sampling of the locations happens twice: a uniform, coarse sampling to find density values along the ray, followed by a dense sampling weighted by those values, so that more samples are used at high-density regions.

At the time, NeRF demonstrated a new level of performance for view synthesis. The density and radiance fields make it possible to represent view-dependent effects with high quality. The neural networks also interpolate well in locations and directions, resulting in impressive novel view renderings. However, it was limited to scene sizes on an object scale with dense input under constant lighting. Also, NeRF models are slow to render because the volumetric rendering method requires many evaluations of neural networks along every ray. Many follow-up works have been proposed to improve NeRF, such as handling input images with different appearances [23, 64, 76, 135], using few inputs [52, 146], avoiding aliasing [8, 9], and reducing computational

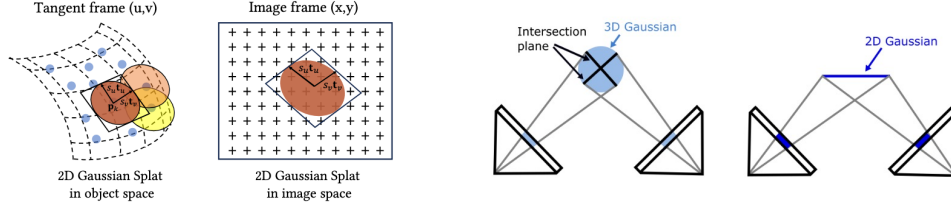


Figure 1.2: **Left:** visualization of projecting 2D Gaussians from 3D space to image (“splatting”). This can be implemented very efficiently for both 3D and 2D Gaussian primitives. **Right:** why 2D Gaussians is a more accurate depth representation than 3D Gaussians. Images taken from [51].

cost [22, 81, 105].

1.2 3D Gaussian Splatting

3D Gaussian Splatting also decomposes the scene into density and radiance fields, and follows the same volume rendering principle. The key difference is that, instead of neural networks, it uses explicit Gaussian primitives to encode the fields, which brings two advantages. First, density values can be evaluated much more efficiently than neural network queries. In addition, the formulation enables efficient approximation of volume rendering with a fast rasterization process. Combining both, 3DGS outperforms NeRF in speed by a large margin and quickly gained popularity.

Conceptually, each Gaussian primitive is centered at a 3D location (its mean) and has a 3D covariance matrix, roughly ensembling its shape. A 2D example is shown in Figure 1.2 (left). Its value represents the density field value and can be evaluated anywhere in space, although 3DGS truncates the range to 3 times its largest variance so that each Gaussian encodes the volume density of an ellipsoid region at its center. In addition, it is associated with three sets of spherical harmonics coefficients to encode the RGB color in every direction. During volume rendering, as the ray intersects a Gaussian, the density of the ray segment can be determined by integrating the

Gaussian, and the radiance can be found by evaluating the spherical harmonics with the direction of the ray.

In practice, the key contribution of 3DGS is *splatting*, which approximates the volumetric rendering process by first projecting the Gaussians onto the image plane and then combining their color with front-to-back alpha blending. This process is similar to *rasterization*, and is extremely computationally efficient compared to volumetric integration. As a result, 3DGS achieved significant speed-ups over NeRF-based methods while maintaining high view synthesis quality, and has been favored over NeRFs in research since its introduction.

However, the approximations bring several unique drawbacks. 3DGS suffers from “popping” artifacts, that is, there can be abrupt changes in color and brightness in the rendered images with a smoothly moving camera. This is caused by inaccuracies in both projection and z-ordering [91]. For the same reason, it is more challenging to extract consistent geometry like meshes from 3DGS, and various works have been proposed [42, 51, 149] to improve the use of Gaussians for geometry representation. In particular, 2DGS [51] use 2D surfels instead of 3D volumes as their primitives, which allows accurate per-Gaussian depth to be defined and calculated from ray-plane intersection. An illustration is in Figure 1.2 (right). It gives more accurate geometry compared to 3DGS-based works, and as we will show, it is important in other tasks like inverse rendering.

Similarly to NeRFs, many works have explored extensions of 3DGS to wider applications, such as incorporating depth hints [29, 63, 130], handling sparse input [71, 156], and reducing excessive amounts of Gaussian primitives [33, 83].

1.3 Inverse Rendering

As discussed above, the state-of-the-art view synthesis approaches decompose a scene into its geometry and radiance. While they achieve high-quality novel view rendering, one cannot easily modify the scene, for example, by adding other objects or changing the lighting because the illumination of the scene is encoded and fixed everywhere. A closely related problem is *inverse rendering*, whose goal is to decompose the scene into geometry, surface material properties, and lighting under a certain rendering model. Indeed, with a physically based rendering model, novel views can be rendered if the scene decomposition is accurate. Such a decomposition of radiance can enable more applications like scene relighting and editing, which allows the reconstructed objects to be easily inserted into new virtual environments. In contrast, the goal of novel view synthesis is to produce images, without much preference on the underlying scene representation, so it may not necessarily support these applications. One can therefore consider inverse rendering to be a more challenging extension of view synthesis under the constraint of physically-based rendering.

Similarly to recent view synthesis approaches, inverse rendering is also largely based on *differentiable rendering*. Given input images, the scene can be rendered from known camera poses, and the rendered image is compared with the ground truth to produce gradients to update the estimated scene parameters. However, for a long time, physically accurate path-tracing-based approaches had been computationally prohibitive. This is due to the need to model light transport according to the rendering equation, which is traditionally solved with recursive path tracing that requires many steps of ray tracing and surface evaluation in the scene. The cost was reduced by adjoint approaches that propagate gradients and reuse caches [85, 111]. Later, Neural

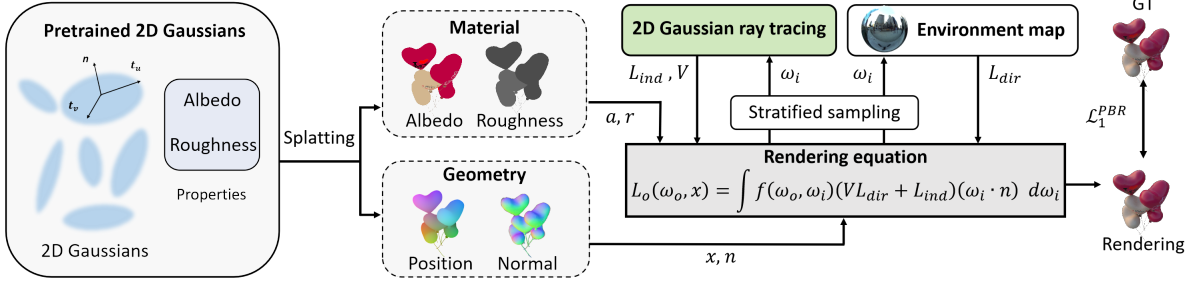


Figure 1.3: Pipeline of IRGS, a state-of-the-art two-stage method to use 2D Gaussians for inverse rendering. In the first stage, a 2DGS model is optimized for the scene to represent scene geometry and radiance. The 2D Gaussians are attached albedo and roughness values to be rendered with splatting. In the second stage, the rendering equation is used to render images from the geometry, radiance and material properties represented by the 2DGS model. The parameters are optimized to minimize the loss between renderings and ground truth images. Taken from IRGS [39].

Radiosity (NeRad) proposed to cache incident radiance with neural networks that are supervised with a residual constraint to reduce bias [44]. The approach reduces the required path length to as small as one, and the idea has been applied to inverse rendering in Inverse Neural Radiosity (InvNeRad) [45]. While these methods enable practical unbiased inverse rendering that handles global illumination effects caused by inter-reflections of light inside the scene, they are limited to known geometry and lights.

Interestingly, the developments of inverse rendering have been boosted by the advancements in novel view synthesis. Many recent works incorporate NeRF [5, 17, 18, 54, 73, 104, 110, 136, 139] and 3DGS [39, 43, 68, 102, 140] as their backbone. For one thing, both NeRF and 3DGS enable efficient reconstruction of scene geometry, giving a good starting point for in-the-wild inverse rendering. For another, as these methods decompose the scene into geometry and radiance fields, their scene radiance representation naturally serves as a radiance cache in physically based inverse rendering, as seen in the example below.

Here, we take one of the latest works, IRGS [39], as an example to depict a typical pipeline of inverse rendering. Their pipeline is illustrated in Figure 1.3. First, a 2DGS model is optimized

for the scene. Then, each Gaussian primitive is attached with material parameters that can be alpha-composited like their colors to produce material maps. To render an image, the Gaussians are splatted as usual; but instead of using the radiance field-composited color, only the produced material maps and depth maps are used. For each pixel, its corresponding surface point can be found by its depth, whose surface material is also obtained from the map. Then, the rendered color of this pixel is the sum of direct and indirect incident illumination at its surface point, modulated by its material parameters. To query incident illumination, many rays are casted towards the hemisphere over the point. The incident rays that do not hit other surfaces in the scene are considered to receive direct illumination, sampled from the environment light. Other rays will hit Gaussians in the scene, and they will carry indirect illumination that have been reflected in the scene to finally arrive at the surface point. Their values, instead of actually path-tracing the bounces, can be determined directly by volumetric rendering the Gaussians along the rays, since they encode radiance of the scene. In this way, the Gaussians serve as a cache of the scene’s indirect illumination. Therefore, the final pixel color is affected by its material parameters, direct illumination from the environment, and indirect illumination from the scene. The gradients of the loss can be subsequently backpropagated to update all of these components.

Although IRGS is efficient and handles global illumination effects, it has several limitations. Notably, it does not resolve the ambiguity between light and albedo and bakes shadows in the estimated albedo maps. Also, ray-tracing 2D Gaussians can be inaccurate due to the fact that the Gaussian primitives are not perfectly aligned with surfaces. Finally, incident indirect illumination depends on the encoded radiance in the 2DGS model, but there may be regions that are not directly observed and have inaccurate cached radiance.



Figure 1.4: In Chapter 2, we solve the problem of decoupling foreground objects with their associated effects like shadows from background.

1.4 Our Work

In this thesis, we present our approaches to relax the requirements of input for novel view synthesis and inverse rendering. Our goal is to enable high-quality solutions from less constrained image sets, making it easier to capture the data required to produce nice results. Extending the view synthesis methods to handle less constrained inputs like these is essential to make them accessible to more people and broaden their applications.

Specifically, we focus on three types of challenging input.

In Chapter 2, we handle the presence of foreground objects and their casted effects for scene reconstruction from videos. Imagine taking a video of a street and modeling it using NeRF, where people are walking and cars are moving. How can we exclude these foreground objects and obtain a clean background reconstruction? One viable approach is to obtain object masks and exclude them from background supervision. However, off-the-shelf object detection and segmentation tools generate masks that are limited to objects and do not include their *associated effects* like shadows and reflections. Naively applying these masks can result in unwanted outcomes like shadows and reflections baked into the background.

A closely related approach is Omnimatte [75] that enhances the video matting problem, which usually includes only the object in the mattes, to include objects with their associated

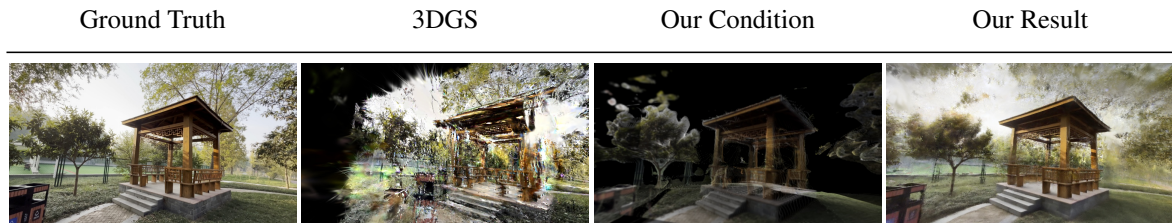


Figure 1.5: In Chapter 3, we handle view synthesis from very sparse inputs. These are examples of having only 3 input views.

effects. The resulting layers are called *omnimattes* to highlight this fact. They decompose a video into several foreground omnimatte layers and one background layer. The background is represented by an image and is warped to each video frame with homography. This representation limits the expressiveness of their background and their applications, since homography cannot handle common 3D effects like parallax in real-world videos.

To solve this problem, we propose a method that decomposes a video into 2D foreground omnimattes and a 3D background model. Each foreground layer includes an object with its casted effects, and the background is a clean reconstruction without baked artifacts. The key insight of our approach is that the foreground objects often include complicated shapes and motions that are difficult for 3D modeling but can be captured with image layers. As a result, our approach is robust for in-the-wild videos, while state-of-the-art methods to decompose videos into 3D foreground-background layers [127] are sensitive to hyperparameters and often fail for non-trivial objects and motions.

In Chapter 3, we consider the scenario in which only a few input images are available. Both NeRF and 3DGS work very well when dense observations are provided. However, the models can suffer from overfitting if the inputs are sparse, which results in inconsistencies in the reconstructed 3D geometry and artifacts in novel view renderings. For a medium level of sparsity, like reducing from a hundred views to a few dozen, numerous works have proposed to

utilize regularization, like surface normal heuristics, monocular depth estimators, and geometry foundation models [32, 58, 63, 130]. However, regularization becomes less effective when the number of images drops further, for example, to less than ten for a room. In these cases, there are too few observations to constrain the models, and there exist considerable portions of the scene that are simply unseen.

The emergence of powerful image and video generative models brings a solution to the problem with their capacity to produce plausible interpolations between views. Many works have explored the use of generated images as *pseudo-input views* for scene reconstruction [71, 103, 107]. Among them, 3DGS-Enhancer [71] had been shown to achieve reasonable reconstruction with as few as 3 views using a video diffusion model. Their pipeline trains two 3DGS models: first, a coarse 3DGS model is trained with the sparse input. It is used to render low-quality interpolated views for conditioning the video diffusion model, which generates high-quality predictions of the views. Then, the fine 3DGS model is trained with both the input and generated views.

Following this line of work, we propose a novel approach to leverage the same video diffusion model, but with three key contributions to further improve efficiency and quality. First, we leverage a powerful and fast feedforward geometry model [113] to obtain a dense point cloud of the scene. The geometry model is robust for very few views, and we use the point cloud both as the initialization of our 3DGS model and the condition to the video diffusion model. In this way, we not only reduce the time consumption, but also benefit from better scene initialization and conditions. Second, we observe that the generated images from diffusion models, while visually accurate, often include geometric inconsistencies. For example, the pillars and windows of a building may shift slightly. We address this issue with optimizable image warps that give the

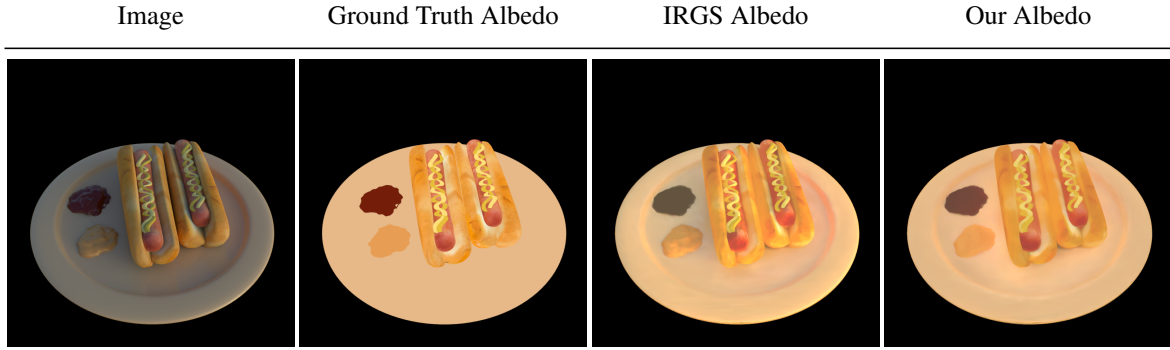


Figure 1.6: In Chapter 4, we reduce ambiguity in inverse rendering. In this example, IRGS appears more blue on the left and red on the right due to ambiguity of light.

model the ability to alleviate the inconsistencies by modifying the generated images, instead of hacking them in the 3D reconstruction. Lastly, we introduce a robust sampling strategy to prevent a situation in which the original input views are overwhelmed by a large number of interpolated views and have diminished effects in reconstruction.

Finally, in Chapter 4, we relax the input requirements for inverse rendering. Recent methods have achieved fast inverse rendering using 3D and 2D Gaussian-based scene representations [39, 68, 140]. The state-of-the-art method, IRGS [39], also takes into account global illumination effects by implementing a ray tracer for 2D Gaussians, allowing them to query incident radiance by integrating Gaussians along arbitrary rays. However, decomposing the radiance into the material properties and environment lighting remains a highly ambiguous problem. Although existing approaches usually incorporate heuristic-based regularization, like enforcing smoothness of albedo and roughness based on smoothness of input images, they still fall short when shadows and reflections are present and produce albedo maps with baked artifacts.

To this end, we propose a simple capture setup that effectively addresses the ambiguity. In typical capturing, a set of photos is taken from different angles. We additionally rotate the object by a predetermined amount and capture another set of images. This can be repeated as

many times as needed, but we find that rotating twice by 120° each a good balance between the outcome and the amount of work. We call this approach *RotLight*, since for each set of photos, we effectively capture the appearance of objects under a rotational environment lighting.

In addition, we identify and address two limitations that cause inaccuracies in incident light modeling. First, in 2D Gaussians, when viewpoints are changed, the same 3D point may be represented by different subsets of Gaussians depending on the ray being integrated. This results in inconsistent geometry queries, especially for incident rays, and harms the performance of the radiance cache MLPs. To resolve this, we propose to extract a mesh from the initial 2DGS reconstruction and use it as a proxy geometry to perform ray casting for incident queries. In addition, the mesh is useful for addressing the second limitation that the radiance cache is inaccurate at under-observed surface locations. Inspired by InvNeRad [45], we randomly sample points and directions in the scene using the extracted mesh, and minimize the residual between the cached radiance and the rendered radiance. We show that these proposed changes further boost the accuracy of the estimated albedo.

1.5 Contributions

In summary, we advance the research of 3D scene reconstruction in three aspects.

1. We reconstruct a clean background from videos with moving foreground objects by decomposing them into 2D foreground omnimatte layers and a 3D background model. Each foreground layer captures the dynamics of an object together with its casted effects, such as shadows and reflections, which enables the background layer to achieve a clean model of the scene free from baked effects. This work is presented in ICCV, 2023.

2. We improve view synthesis from extremely sparse inputs by combining large geometry priors and generative video priors. The geometry prior is leveraged to produce a dense point cloud estimation, which is then used to condition the video generation model. The generated images are used in addition to the sparse inputs to reconstruct a high-quality scene model. To further alleviate their hallucinations and inconsistencies, we introduce optimizable image warps to make the optimization more forgiving to geometric differences, and robust image sampling strategy to prevent the real input from being ignored. This work will be submitted to 3DV, 2026.
3. We propose a simple capture approach to improve albedo recovery in inverse rendering. We demonstrate that by observing the object from several rotated light sources, which can be easily achieved by rotating the object several times during capture, albedo maps with much fewer artifacts can be recovered. While keeping a 2DGS-based scene representation for efficiency, we identify its limitations in consistency for incident light querying. To address these issues, we propose to incorporate a proxy mesh and residual constraints to achieve better handling of global illumination effects. This work will be submitted to 3DV, 2026.

1.5.1 Additional Works

We have also collaborated on several works that are not presented in detail in this thesis.

1. Inverse Neural Radiosity (InvNeRad) [45] proposes the use of a neural radiance cache in differentiable path tracing for inverse rendering. The cache, modeled by a neural network, encodes the incident radiance everywhere in the scene. The correctness of the radiance cache is further improved by the residual constraint that the cached radiance of

any point should match the rendered radiance. Therefore, lengthy path trace operations can be replaced by an efficient query to the network, which enables practical and efficient physically-based inverse rendering.

2. In Speedy-Splat [47], we improve the rendering speed of 3DGS by addressing two key inefficiencies in their rasterizer implementation. First, we use an accurate rectangular bounding box for the Gaussians instead of a coarse square box. Second, we perform exact tile-Gaussian intersection tests in addition to tile-bounding box tests. In addition to these lossless improvements, we also propose a novel pruning technique to further reduce the number of Gaussians at a small cost of quality. The resulting rendering speed is more than 6 times faster than 3DGS.
3. In our inverse rendering work GaNI [126], we achieve better estimation of geometry and material parameters of real-world scenes using co-located capturing. In this capturing setup, we use the flash light of a mobile phone camera to obtain image observations in a dark room. Because of the fixed and small displacement between the light source and the camera, we essentially achieve a near known-light condition for real world scenes. The information is used in our light position-aware neural networks for less ambiguous reconstruction. Finally, we incorporate InvNeRad to better handle global illumination effects.
4. We propose Denoising Density Estimator (DDE) [13] for image data density estimation and generation. A DDE is a density estimator derived from a denoising estimator by modifying its optimization target from noise estimation to the *gradient* of the noise estimation. We then propose a learning objective that incorporates a DDE to optimize a model that directly samples the target distribution. DDE makes no assumptions about network architectures

and is a forward sampling model that does not require the iterative sampling process.

5. We propose a two-stage approach to recover plausible textured 3D point clouds from single RGB images [50]. The input image is processed by a pre-trained network to project visible 2D parts of the object to a partial 3D point cloud. The partial point cloud is projected to several partial depth maps and texture maps from different views, which are then combined with a multi-view texture and depth completion module to produce a complete textured point cloud.

Chapter 2: OmnimateRF: Robust Omnimate with 3D Background Modeling

2.1 Introduction

Video matting is the problem of separating a video into multiple layers with associated alpha mattes such that the layers are composited back to the original video. It has a wide variety of applications in video editing as it allows for substituting layers or processing them individually before compositing back, and thus has been studied well over decades. In typical applications like rotoscoping in video production and background blurring in online meetings, the goal is to obtain the masks containing only the object of interest. In many cases, however, it is often preferred to be able to create video mattes that include not only the object of interest but also its associated effects, like shadow and reflections. This could reduce the often-required, additional manual segmentation of secondary effects and help increase realism in the resulting edited video. Being able to factor out the related effects of foreground objects also helps reconstruct a clean background, which is preferred in applications like object removal. Despite these benefits, this problem is much more ill-posed and has been much less explored than the conventional matting problem.

The most promising attempt to tackle this problem is Omnimate [\[75\]](#). *Omnimattes* are RGBA layers that capture dynamic foreground objects and their associated effects. Given a video and one or more coarse mask videos, each corresponding to a foreground object of interest, the

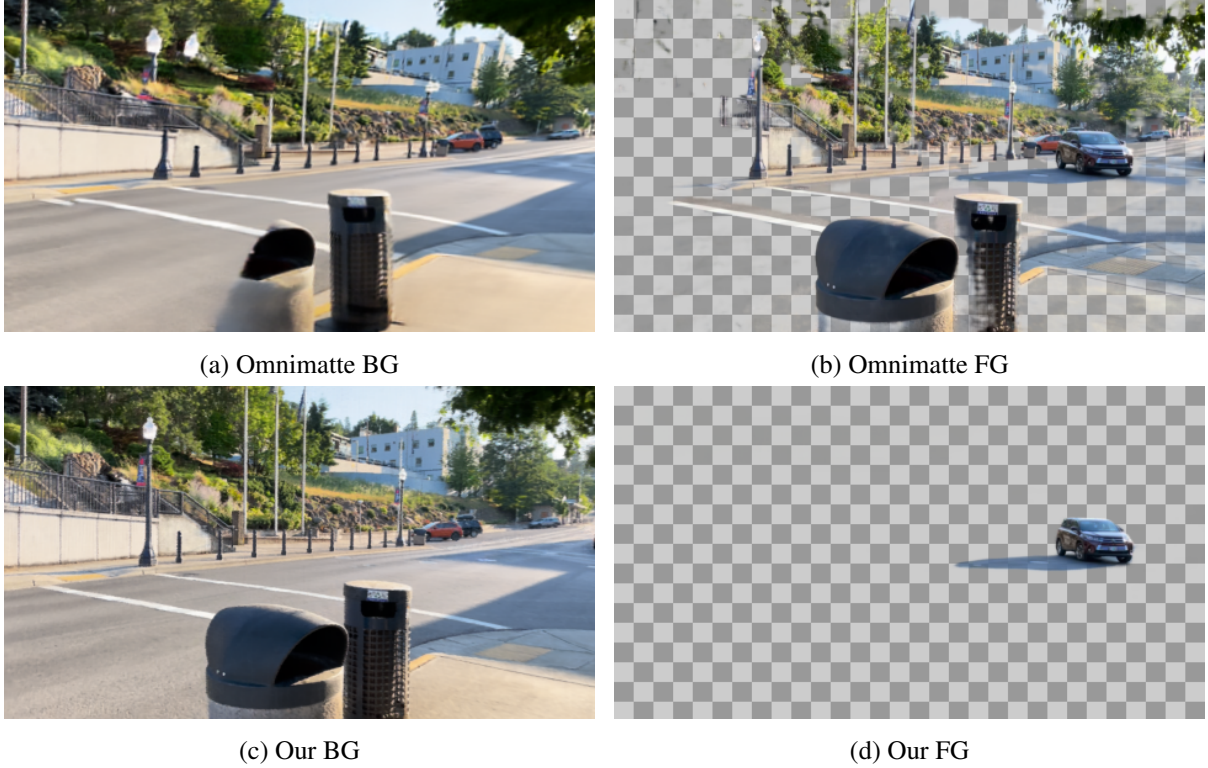


Figure 2.1: **Video with parallax effects.** Limited by their 2D image representation (a), previous works such as Omnimate fail to handle videos with parallax effects in the background. Their foreground layer (b) has to capture (dis)occlusion effects to minimize the reconstruction loss. In contrast, our method employs a 3D background (c), enabling us to obtain clean foreground layers (d).

method reconstructs an *omnimatte* for each object, in addition to a static background that is free from all of the objects of interest *and* their associated effects. While Omnimate [75] works well for many videos, it is limited by its use of homography to model backgrounds, which requires the background be planar or the video contains only rotational motion. This is not the case as long as there exists parallax caused by camera motions and objects occlude each other. This limitation hinders its application in many real-world videos, as shown in Fig. 2.1.

D²NeRF [127] attempts to address this issue using two radiance fields, which model the dynamic and static part of the scene. The method works entirely in 3D and can handle complicated scenes with significant camera motion. It is also self-supervised in the sense that no

mask input is necessary. However, it separates all *moving* objects from a static background and it is not clear how to incorporate 2D guidance defined on video such as rough masks. Further, it cannot independently model multiple foreground objects. A simple solution of modeling each foreground object with a separate radiance field could lead to excessive training time, yet it is not clear how motions could be separated meaningfully in each radiance field.

We propose a method that has the benefit of both by combining 2D foreground layers with a 3D background model. The lightweight 2D foreground layers can represent multiple object layers, including complicated objects, motions, and effects that may be challenging to be modeled in 3D. At the same time, modeling background in 3D enables handling background of complex geometry and non-rotational camera motions, allowing for processing a broader set of videos than 2D methods. We call this method *OmnimatteRF* and show in experiments that it works robustly on various videos without per-video parameter tuning. To quantitatively evaluate the background separation of a 3D scene, D²NeRF released a dataset of 5 videos rendered with Kubrics, which are simple indoor scenes with few pieces of furniture and some moving objects that cast solid shadows.

We also render five videos from open-source Blender movies [16] with sophisticated motions and lighting conditions for more realistic and challenging settings. Our method outperforms prior works in both datasets, and we release the videos to facilitate future research.

In summary, our contributions include the following:

1. We propose a novel method to make Omnimatte [75] more robust by better modeling the static background in 3D using radiance fields [79].
2. Utilizing the *omnimatte* masks, we propose a simple yet effective re-training step to obtain

a clean static 3D reconstruction from videos with moving subjects.

3. We release a new dataset of 5 challenging video sequences rendered from open-source blender movies [16] with ground truths to better facilitate the development and evaluation of the video matting with associated effects (aka *omnimatting* [75]) problem.

2.2 Related Work

Video Matting. There is a long line of work exploring video matting due to its importance in video editing. Green screening and rotoscoping are critical first steps in any visual effects pipeline. The matting problem aims to extract the foreground subjects into their own RGBA layers and separate them from the background RGB layer, which is a highly under-constrained problem. Many approaches have utilized motion and depth cues in addition to integrating user interactions [6, 19, 28, 66, 116]. Background Video Matting [69] specifically addresses real-time video matting of people and preserving strand-level hair details.

Matting with Associated Effects. Video matting is often insufficient, as foreground subjects might have associated effects like shadows or reflections that need to be extracted into the foreground RGBA layers. This problem has not been explored as extensively and, in practice, is often dealt with manually using advanced interactive rotoscoping tools [65]. Omnimatte [75] was the first to propose a generic framework capable of learning any associated effect. Previous works often specifically addressed associated effects like shadows [119, 120]. The ability to obtain matte layers with associated effects has many exciting applications, such as re-timing motions of different people [74], consistent background editing [56, 61], background subtraction, green screening, and many other video effects [75]. Recently, FactorMatte [41] has been proposed to improve the

quality with data augmentation and conditional priors. These works have in common that they take predefined masks that hint at the foreground objects and decompose each video into several layers, with one object in each layer with its associated effects. Then, there is a background layer, a 2D static image, or a deformable atlas shared by all the frames. The background is warped and cropped via a homography to render each frame. While the foreground layers have shown great potential in capturing dynamics, their single image background limits the application of these methods to videos with planar environments without parallax effects caused by camera motion.

Radiance Fields. Radiance fields (RF) emerged as 3D representations capable of capturing geometric details and photorealistic appearances [79]. Radiance fields model the 3D scene as a continuous function that maps the position and the viewing direction of any point in world space to its color and opacity. Novel views can be synthesized via volume rendering along rays cast. This continuous function is learned by optimizing with a reconstruction loss on the rendered images. This view-dependent volumetric representation can model various challenging scenes that previous surface-based methods struggled to handle: e.g., shiny surfaces like metals or fuzzy surfaces like hair or fur. Since then, it has been extended along multiple axes: better appearance modeling (e.g., reflection and refraction [3, 4, 12, 110], faster optimization [22, 81, 105] and modeling dynamic scenes [36, 67, 72, 129]. Since the MLP-based implicit RF representations are slow to train, we use voxel-based explicit radiance field representations [22] [105]. Specifically, we use the factorized voxel grid representation from [22].

Self-Supervised Video Dynamics Factoring. Another related work is video dynamics factoring without needing a predefined mask. One recent work is deformable sprites [143] that rely only

on motion cues. Similar to other video matting works, it has a 2D foreground and background layers and the same limitations as Omnimate. For modeling in 3D, D²NeRF [127] proposes to decouple the scene with two radiance fields, one for the dynamic content and the other for the statics. D²NeRF [127] handles a special case of matting with only one foreground object, and, compared to the other methods, it is not limited to planar backgrounds. However, the self-supervised method relies on the heuristics that require per-video hyper-parameter tuning and does not robustly generalize to new videos. The quality of the foreground reconstruction can also be limited for objects that have large nonrigid motions.

We, therefore, propose a method for video matting with associated effects that has the advantages of supervised 2D mattes that support multiple individual objects with great details, as well as 3D background decoupling that works with non-planar videos.

2.3 Method

The concept of *omnimattes* is proposed by Lu et al. [75], extending RGBA video mattes to capture associated effects of the objects of interest like shadows and reflections. To avoid any confusion, in the following text, we refer to their work as capital Omnimate, and the resulting RGBA layers as italic *omnimatte*. In the matting setup, the user prepares a video of T frames $\{I_t\}_{t=1}^T$, and N ordered mask layers $\{M_t^i\}_{i=1}^N$, each containing a coarse mask video of an object of interest. The video’s camera parameters are also precomputed as $\{P_t\}$.

The goal is to predict RGBA foreground layers C_t^i and α_t^i that contain the objects together with their associated effects, and a background layer B_t which is clean and free from the effects cast by the foreground objects. An input frame I_t should be reconstructed by alpha compositing

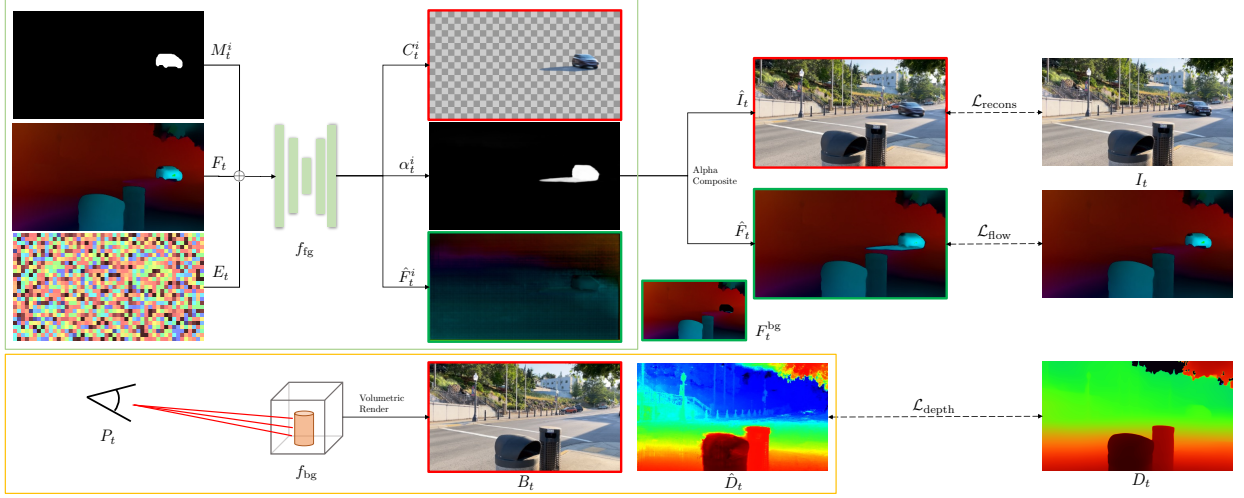


Figure 2.2: **Method overview.** We propose a video matting method, named OmnimatteRF, which combines 2D foreground layers with a 3D background layer. The foreground branch (f_{fg} , in green box) predicts an RGBA layer (C_t^i, α_t^i) for each object, and an auxiliary flow output ($\hat{F}_t^i$). The background branch (f_{bg} , in yellow box) produces a background layer with depths ($B_t, \hat{D}_t$). **Optimization.** During training, predicted colors ($\hat{I}_t$) and flow ($\hat{F}_t$) are alpha-composited, whose inputs have red and green borders respectively. The right most column illustrates the data terms in the loss function, and we omit the regularization terms in this illustration.

the foreground layers above the background.

In Omnimatte, the background is represented by a static 2D image and a homography transform P_t . To compose a frame, part of the static background is extracted according to the estimated homography P_t . The key idea of our work is to represent the static background in 3D using a radiance field, while keeping the foreground in 2D to better capture the dynamics of objects. We employ an explicit factorized voxel-based radiance field [22] to model the background. In this case, P_t represents a camera pose, and a background frame is rendered with volume rendering. Note that the foreground layers are still 2D videos. We refer to this combination as the OmnimatteRF model.

2.3.1 The OmnimateRF Model

An outline of our model is depicted in Figure 2.2. The model has two independent branches: foreground and background. For any given frame, the foreground branch predicts an RGBA image (*omnimatte*) for each object, and the background branch renders a single RGB image.

Preprocessing. Following similar works, we use an off-the-shelf model RAFT [108] to predict optical flow between neighboring frames. The flow is used as an auxiliary input and ground truth for supervision, denoted by $\{F_t\}$. We also use an off-the-shelf depth estimator MiDaS [92] to predict monocular depth maps $\{D_t\}$ for each frame and use them as ground truth for the monocular depth loss.

Background. The background branch consists of a static neural radiance field, f_{bg} , encoding the 3D representation of the scene. To render a pixel in a frame I_t , a ray is traced according to the estimated camera pose P_t , and the final RGB color is produced via volumetric rendering. The result of rendering the entire frame is $(B_t, \hat{D}_t) = f_{\text{bg}}(P_t)$, where B_t is an RGB image and $\hat{D}_t$ is a depth map.

Foreground. The foreground branch is a UNet-style convolutional neural network, f_{fg} , similar to that of Omnimate. The input of the network is a concatenation of three maps:

1. The coarse mask M_t^i . The mask is provided by the user, outlining the object of interest.

Mask values are ones if the pixels are inside the object.

2. The optical flow F_t . It provides the network with motion hints. Note that the network also predicts an optical flow as an auxiliary task (detailed in Sec. 2.3.2.2).

3. The feature map E_t . Each pixel (x, y) in the feature map is the positional encoding of the

3-tuple (x, y, t) .

Multiple foreground layers are processed individually. For the i -th layer, the network predicts the *omnimatte* layer (C_t^i, α_t^i) and the flow $\hat{F}_t^i$.

Detail Transfer. For a tradeoff between image quality and training time, the foreground network typically produces a color layer with missing details when the alpha layers have captured sufficient associated effects. To boost the output quality, Omnimatte transfers details from input frames. We include the same process in our pipeline. Note that this is a post-processing step to produce final results, and does not apply to model optimization.

2.3.2 Optimizing the Model

We optimize an OmnimatteRF model for every video since both branches of our model are video-specific. To supervise learning, we employ an image reconstruction loss and several regularization losses.

2.3.2.1 Reconstruction Loss

We compute the reconstruction loss with the composed image $\hat{I}_t$ by alpha composition of foreground and background layers:

$$\hat{I}_t = \sum_{i=1}^N \left(\prod_{j=1}^{i-1} (1 - \alpha_t^j) \alpha_t^i C_t^i \right) + \prod_{i=1}^N (1 - \alpha_t^i) B_t \quad (2.1)$$

And the reconstruction loss is the mean-squared-error between the predicted and input frame,

$$\mathcal{L}_{\text{recons}} = ||\hat{I}_t - I_t||^2 \quad (2.2)$$

The reconstruction loss supervises both branches of our pipeline simultaneously. Limited by the computational cost of volumetric rendering, the background layer is rendered only at sparse random locations at each step, where $\mathcal{L}_{\text{recons}}$ is computed for the composed pixel values.

2.3.2.2 Foreground Losses

We follow Omnimate and include the alpha regularization loss $\mathcal{L}_{\alpha\text{-reg}}$, alpha warp loss $\mathcal{L}_{\alpha\text{-warp}}$, and flow reconstruction loss $\mathcal{L}_{\text{flow}}$. We also bootstrap the initial alpha prediction to match the input mask with the mask loss $\mathcal{L}_{\text{mask}}$, which is gradually decayed and disabled once its value drops below the threshold.

While most regularization terms in Omnimate can be applied directly to our pipeline, the flow reconstruction loss is an exception. The formulation of the loss remains identical: given the per-layer flow prediction $\hat{F}_t^i$ and a background layer flows F_t^{bg} , the complete flow $\hat{F}_t$ is composed via alpha composition (Eq. 2.1). Then, the loss is defined as:

$$\mathcal{L}_{\text{flow}} = ||(\hat{F}_t - F_t) \otimes M_t^{\text{fg}}||^2 \quad (2.3)$$

Here, M_t^{fg} is the union of all foreground masks ($\{M_t^i\}$) for the frame I_t , and the loss is only evaluated at the location of *input* coarse masks. The authors of Omnimate have shown

the effectiveness of this loss in their case, and we also demonstrate its importance in an ablation study.

However, it remains unclear how F_t^{bg} can be obtained. In Omnimate, the background flow can be derived from image homography, which serves both as an input to the network and a background for composition. On the other hand, since our 3D background has only known camera poses but not depths, we cannot obtain background flows directly. Instead, we use the ground truth flow F_t as network input to provide motion cues and a masked version of F_t as background flow for composition. The masked flow is $F_t^{\text{m}} = F_t \otimes (1 - M_t^{\text{fg}})$, which is the ground truth optical flow with the regions marked in the coarse masks set to zeros. $\otimes$ denotes elementwise multiplication. We find it crucial to use F_t^{m} rather than F_t for composition, as the latter case encourages the network to produce empty layers with α_t^i equal to zero everywhere.

2.3.2.3 Background Losses

Apart from the reconstruction loss, the background network is supervised by the total variation regularization loss, $\mathcal{L}_{\text{bg-reg}}$, as in TensorRF [22]. In addition, monocular depth supervision is used to improve scene reconstruction when the camera motions consist of rotation only:

$$\mathcal{L}_{\text{depth}} = \text{metric}(D_t, \hat{D}_t), \quad (2.4)$$

where $\hat{D}_t$ is the estimated depth from volume rendering [79], and the metric function is the scale-invariant loss from MiDaS [92]. Also, we empirically find that $\mathcal{L}_{\text{depth}}$ can introduce floaters, and employ the distortion loss $\mathcal{L}_{\text{distort}}$ proposed in Mip-NeRF 360 [9] to reduce artifacts in the

background.

2.3.2.4 Summary

The combined loss for joint optimization is:

$$\mathcal{L} = \mathcal{L}_{\text{recons}} + \underbrace{\mathcal{L}_{\alpha\text{-reg}} + \mathcal{L}_{\alpha\text{-warp}} + \mathcal{L}_{\text{flow}} + \mathcal{L}_{\text{mask}}}_{\text{Foreground}} + \underbrace{\mathcal{L}_{\text{bg-reg}} + \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{distort}}}_{\text{Background}} \quad (2.5)$$

At every optimization step, $\mathcal{L}_{\text{recons}}$ and background losses are evaluated at sparse random locations. Foreground losses are computed for the full image.

2.3.3 Clean Background via Masked Retraining

When the pipeline is trained jointly as described above, it is sometimes observed that the background radiance field models some of the foreground contents like shadows (see Fig. 2.3(c)). Compared to 2D images, 3D radiance fields are so much more capable that they can exploit distorted geometry constructs, such as holes and floaters, to capture some temporal effects, although the models are given no time information. For example, as the camera moves over time, there may be a correlation between whether a surface is covered by shadow and the direction the surface is viewed from.

We illustrate this problem in Fig. 2.3 and explain the cause at an intuitive level. The foreground branch is bootstrapped to produce alpha values that match the coarse mask inputs,

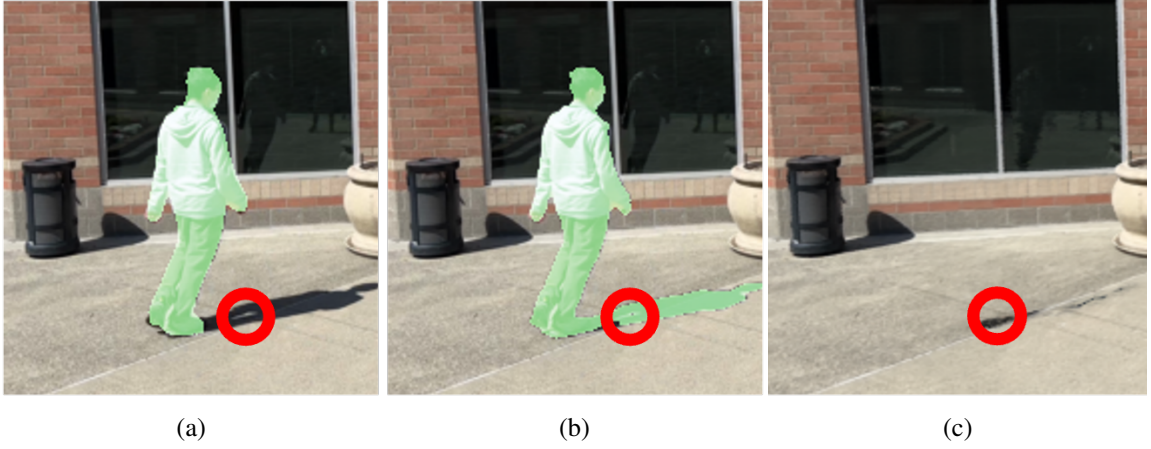


Figure 2.3: **Background Layer Training Signals.** We illustrate how the training signal to the background layer changes over time. It explains why the background captures some of the associated effects (in this example, shadows). We use the pixel circled in red as an example. (a) At the beginning of training, the foreground alpha value (in light green) does not include the shadow. Therefore, α is small and at this pixel, $\hat{I}_t(x, y) \approx B_t(x, y)$. The reconstruction loss $\mathcal{L}_{\text{recons}}$ encourages the background network f_{bg} to produce dark prediction at this location from this viewing angle. (b) As training progresses, α gets larger in the shadow region, and $\hat{I}_t(x, y) \approx C_t^i(x, y)$. This means that f_{bg} receives little to no supervision signals from this pixel. If it has modeled the shadow in some ways (in this case, a hole), it has little incentive to remove it, leaving the artifact in (c).

which include only the object without the associated effects. In other words, α_t values are close to one at the object, but zero in the shadows (for simplicity, we consider one foreground layer in which the object casts a shadow, like in Fig. 2.3). At a pixel (x, y) covered by shadow, Eq. 2.1 simply collapses to $\hat{I}_t(x, y) \approx B_t(x, y)$. The reconstruction loss will therefore encourage $B_t(x, y)$ to match the color of the shadow for a ray shot toward this location.

As training proceeds, f_{fg} will then gradually increase the predicted alpha values at the shadowed regions. If the shadow is hard and α gets close to one, Eq. 2.1 evaluates to $\hat{I}_t(x, y) \approx C_t^i(x, y)$, and the reconstruction loss gives little to no constraint to the background color at the pixel. As a result, f_{bg} is unable to learn to remove the shadow color that it produces for the ray towards frame I_t at (x, y) .

There are also cases where the shadow is soft and α is in between. In these cases, the problem remains ambiguous.

Therefore, we propose to obtain clean background reconstruction via an optional optimization step. In joint training, the foreground *omnimatte* layers can capture most associated effects, including the parts with leaked content in the background layer. The alpha layers α_t can then be used to train a radiance field model from scratch, with no samples from the foreground region where alpha values are high. We show in the ablation study (see Fig. 2.8) that this step produces cleaner background reconstruction for in-the-wild videos. As only the background is optimized, the process is fast and takes less than an hour to complete.

2.4 Evaluation

We compare our quantitative and qualitative methods with Omnimate and D²NeRF [75, 127], which are state-of-the-art methods in 2D video matting and 3D video segmentation, respectively. In addition, we compare with Layered Neural Atlas (LNA) [56], which uses a deformable 2D background in contrast to Omnimate’s static image.

2.4.1 The Movies Dataset

Quantitative evaluation of background segmentation requires a dataset with both input videos and ground-truth background imagery. Prior works primarily use datasets like CDW-2014 [121], which are limited to mostly static backgrounds and are not applicable to our settings. Recently, `Kubrics` is proposed in D²NeRF, which enables the evaluation of 3D background synthesis. However, these videos have relatively simple scenes and lighting. To facilitate the evaluation of video matting and background segmentation in challenging scenarios, we select six clips from three Blender movies in Blender Studio [16]. Compared to `Kubrics`, they feature more complicated scenes and lighting conditions, large nonrigid motion of the characters, and higher resolution. To ensure usability, we manually edit the camera trajectories so that there are sufficient camera motions and the actors have reasonable sizes. We render the clips with and without the actors to obtain input and ground truth for background reconstruction evaluation purposes. The camera poses are also exported.

2.4.2 Experiment Setup

We evaluate the performance of our proposed method on four datasets.

1. `Movies`: our novel challenging dataset.
2. `Kubrics`: the dataset generated and used in D²NeRF, which consists of five scenes of moving objects from 3D Warehouse [109] rendered with Kubric [38].
3. `DAVIS` [88, 89]: short clips with moving foreground subjects, like humans, cars, and animals. This dataset is widely used to evaluate 2D-background matting methods [56, 75, 143].
4. `Wild`: in-the-wild sequences collected from the internet that are closer to casually captured videos, with natural and noisier camera motions, including translations and rotations, as well as objects at different distances from the camera. Naturally, these videos have backgrounds that are challenging for pure 2D methods.

`Kubrics` and `Movies` are synthetic datasets with clean background layer renderings available. Note that novel view synthesis is not the focus of our method, so we evaluate the background with input views. Both datasets have known camera poses and object masks which are used for training and evaluation.

`DAVIS` and `Wild` are real-world videos without clean background. Therefore, we only perform a qualitative evaluation to demonstrate the robustness of our method. For videos in `Wild` we recover camera poses with COLMAP. For videos that COLMAP cannot process reliably, including `DAVIS` videos, we use poses from RoDynRF [72].

To obtain coarse object masks, we attempt to extract them with pre-trained object segmentation models from Detectron 2 [128]. In case it does not work, we use the Roto Brush tool in Adobe After Effects. Detailed procedures are described in the supplementary material. It takes about 10 minutes of manual effort to produce a 200-frame mask.

For all videos, we also estimate homographies with LoFTR [106] and OpenCV to enable

Kubrics	Car			Cars			Bag			Chair			Pillow		
	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑
D ² NeRF	<u>0.135</u>	<u>0.854</u>	<u>34.10</u>	<u>0.105</u>	<u>0.859</u>	<u>34.77</u>	<u>0.131</u>	<u>0.880</u>	<u>33.98</u>	<u>0.090</u>	<u>0.916</u>	<u>33.29</u>	0.105	<u>0.926</u>	<u>38.80</u>
Omnimatte	0.162	0.819	31.14	0.157	0.834	31.20	0.271	0.796	23.64	0.175	0.865	26.91	0.270	0.841	21.17
LNA	-	-	-	-	-	-	0.138	0.835	27.08	0.105	0.881	21.21	<u>0.080</u>	0.923	31.66
Ours	0.033	0.958	39.09	0.032	0.961	39.78	0.029	0.972	39.58	0.023	0.977	42.46	0.022	0.982	43.62

Movies	Donkey			Dog			Chicken			Rooster			Dodge		
	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑
D ² NeRF	-	-	-	0.370	0.694	22.73	-	-	-	0.340	0.708	25.13	0.408	0.729	20.95
Omnimatte	0.315	0.653	<u>19.11</u>	0.279	0.706	21.74	0.312	0.704	<u>20.95</u>	0.220	0.741	23.14	<u>0.067</u>	0.879	23.88
LNA	<u>0.104</u>	<u>0.849</u>	18.79	<u>0.154</u>	<u>0.828</u>	<u>26.08</u>	<u>0.190</u>	<u>0.818</u>	19.22	<u>0.131</u>	<u>0.804</u>	<u>26.46</u>	0.068	<u>0.937</u>	<u>24.94</u>
Ours	0.005	0.990	38.24	0.030	0.976	31.44	0.021	0.978	32.86	0.024	0.969	27.65	0.006	0.991	39.11

Table 2.1: **Quantitative evaluations.** We present the background reconstruction comparison of our method and baselines on the `Kubrics` and `Movies` datasets. Best results are in **bold** and second place are underlined. Results marked - are the ones the method failed to give good separations (visuals in supplementary).

Omnimatte processing.

As mentioned in D²NeRF [127], the method is sensitive to hyperparameters. The authors released five sets of configurations for different videos. We experiment with every video using all provided configurations and report the best-performing ones.

2.4.3 Implementation Details

Our network is built upon the publicly available official implementation of Omnimatte [75], and TensorRF [22]. The videos in `Kubrics` have resolution 512×512 , and all methods run at the resolution 256×256 . For videos in other datasets with a higher resolution of 1920×1080 , we downsample them by a factor of 4.

We optimize the networks for up to 15,000 steps. The learning rate of f_{fg} is set to 0.001 and is exponentially decayed after 10,000 steps. For f_{bg} we use the learning rate scheduling scheme of TensorRF. Training takes up to 6 hours on a single RTX3090 GPU. Detailed network architecture, hyper-parameters and timing data are presented in the supplementary. Our code and datasets will also be made publicly available.

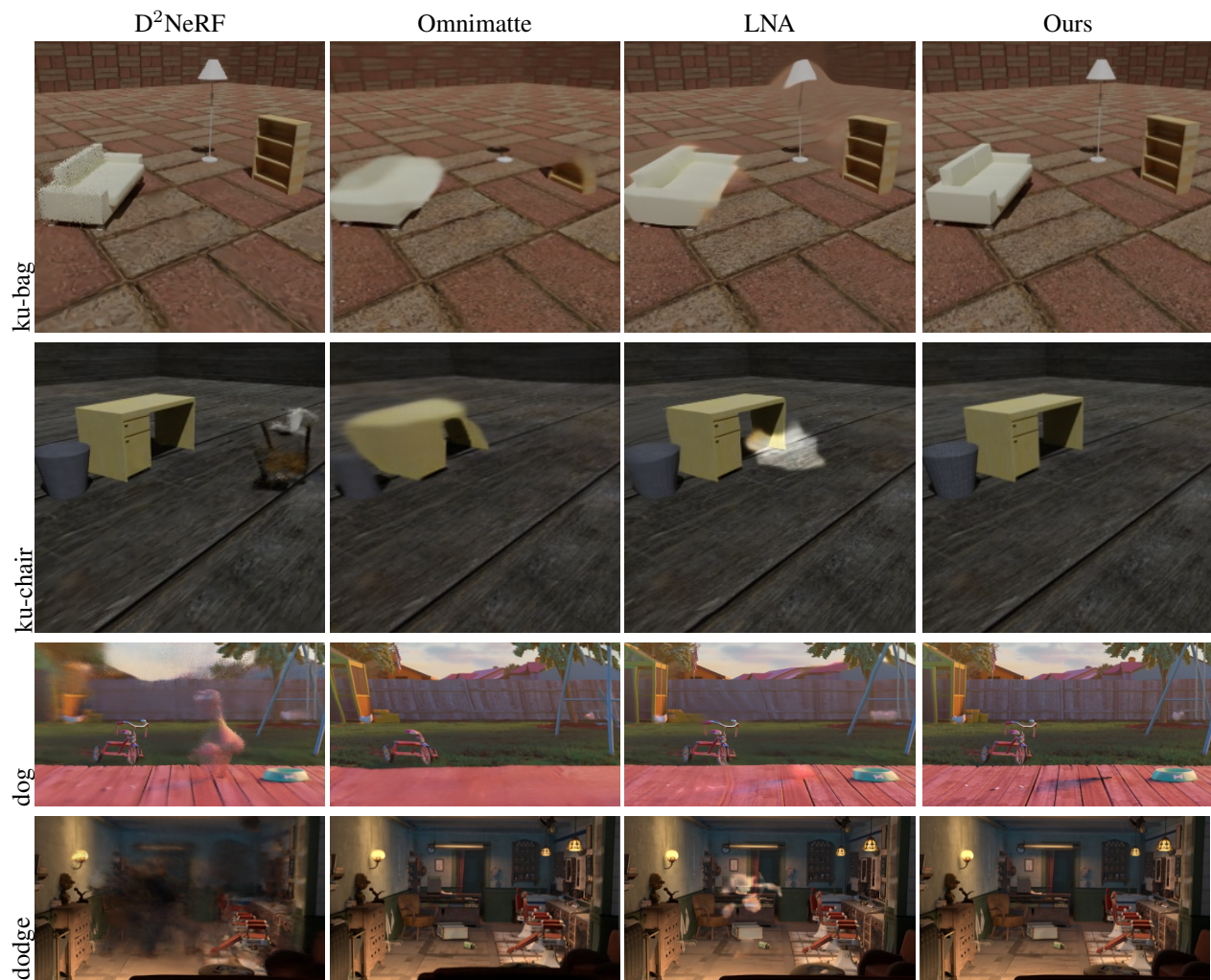


Figure 2.4: **Background Reconstruction.** We show examples of results presented in quantitative evaluations. For videos with parallax effects, 3D methods like D²NeRF and ours reconstruct less distorted background than Omnimatte and LNA.

2.4.4 Quantitative Evaluation

We quantitatively evaluate the background reconstruction quality of our method on two synthetic datasets. We report PSNR, SSIM and LPIPS for all videos in Table 2.1, and some visualizations in Fig. 2.4. For D²NeRF, we tried every provided pre-set configuration for every video in `Movies`, and it only gave good results for the Dog, Rooster, and Dodge videos. Omnimate and LNA with the 2D background layers struggles in both datasets. Our method can handle these videos well.

2.4.5 Qualitative Evaluation

We present a qualitative comparison of the methods in Fig. 2.5. Due to space limitations, we present at least one video from every dataset but show a frame from every selected video in the figure. The original videos are available in supplementary and we highly recommend watching them. D²NeRF works well for the fine-tuned videos but not for new inputs without further hyper-parameter tuning. Omnimate background has significant distortion around objects, and its foreground layer has to compensate for the limitation by capturing all residuals. Our method is versatile enough to perform well for a variety of videos with our 3D background model.

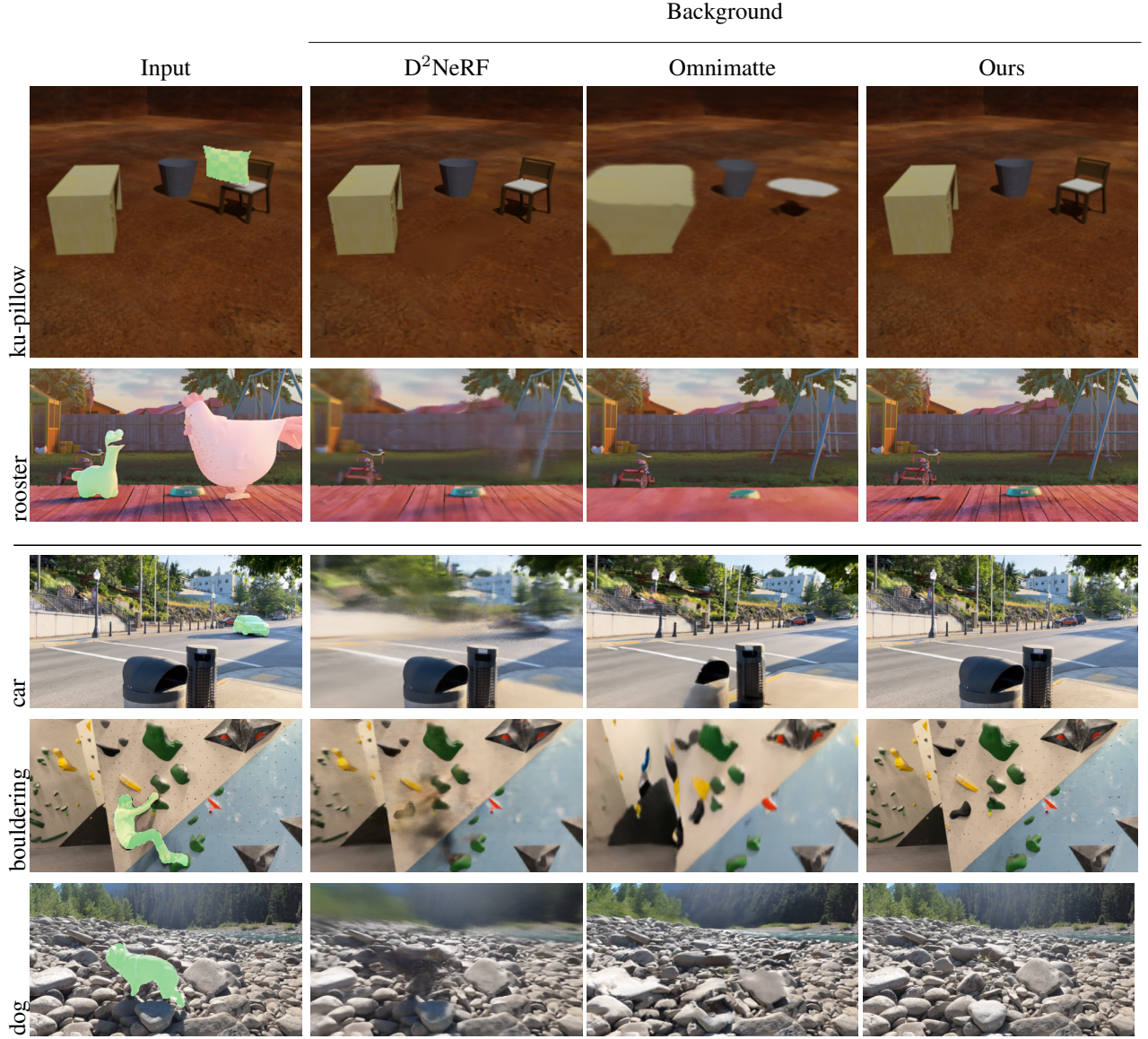


Figure 2.5: **Qualitative comparison (background).** We compare results of our and baseline methods on videos from each dataset. The first two videos are synthetic from *Kubrics* and *Movies*, followed by three *Wild* videos. Omnimatte fails to handle objects in 3D and produces distorted background. D²NeRF works for videos with appropriate hyper-parameters, but does not generalize to new videos easily. Our method handles videos in many different settings. Due to space constraint we defer LNA results to the supplementary. Due to space constraint, the foreground layers are visualized in Figure 2.6.

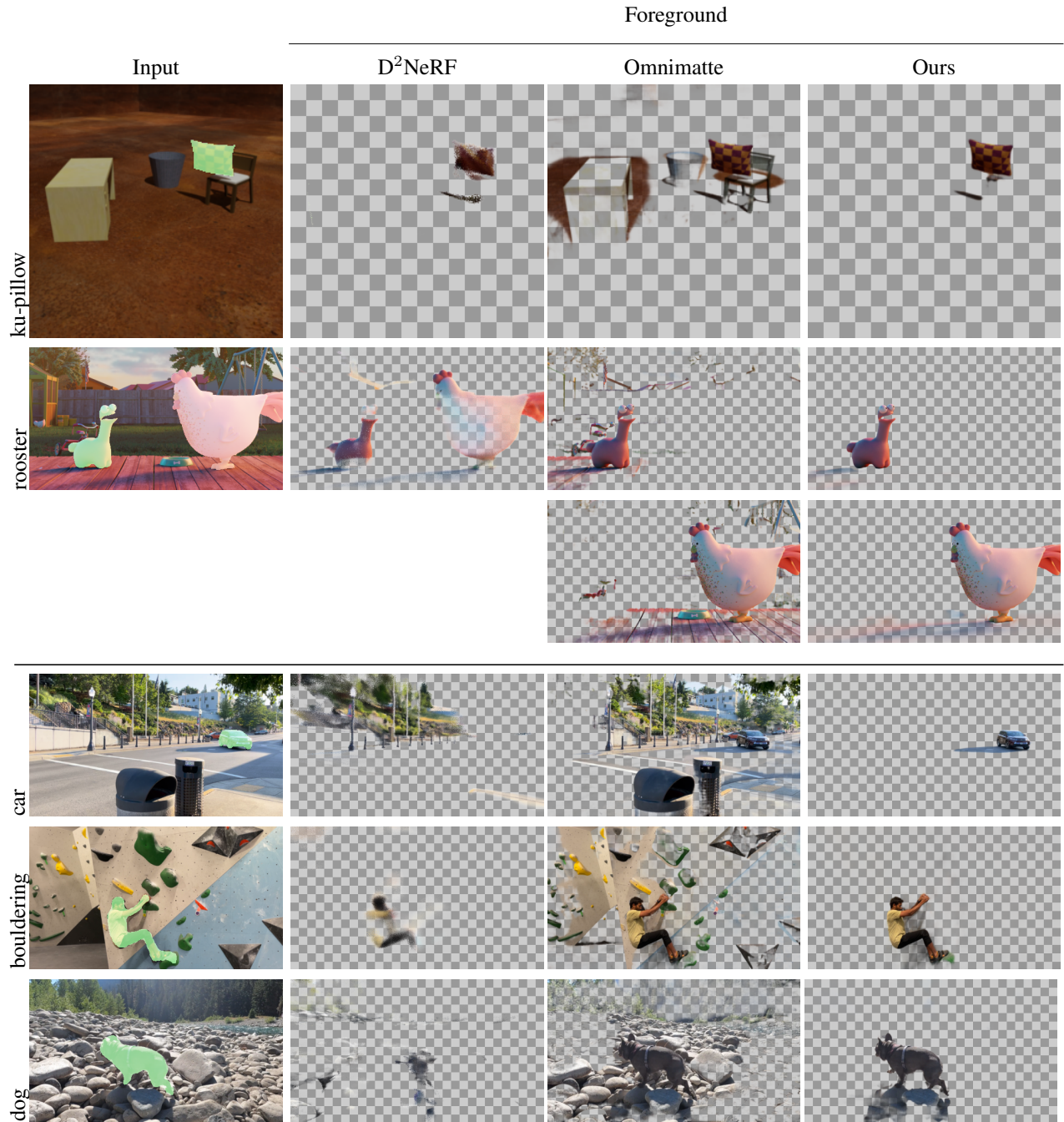


Figure 2.6: **Qualitative comparison (foreground).**

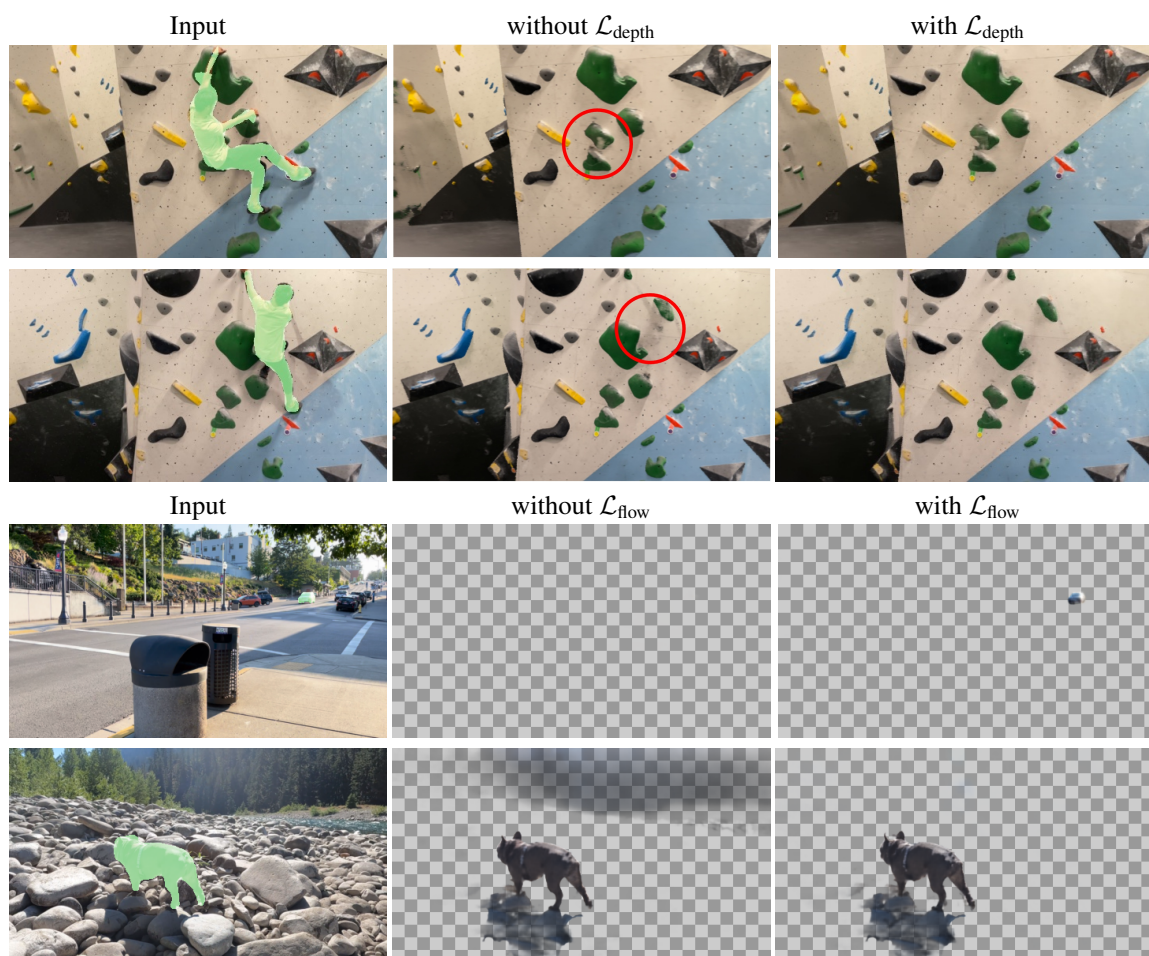


Figure 2.7: **Loss Term Ablations.** Background of real-world videos without $\mathcal{L}_{\text{depth}}$ and foreground without $\mathcal{L}_{\text{flow}}$ can be degraded for real-world videos.

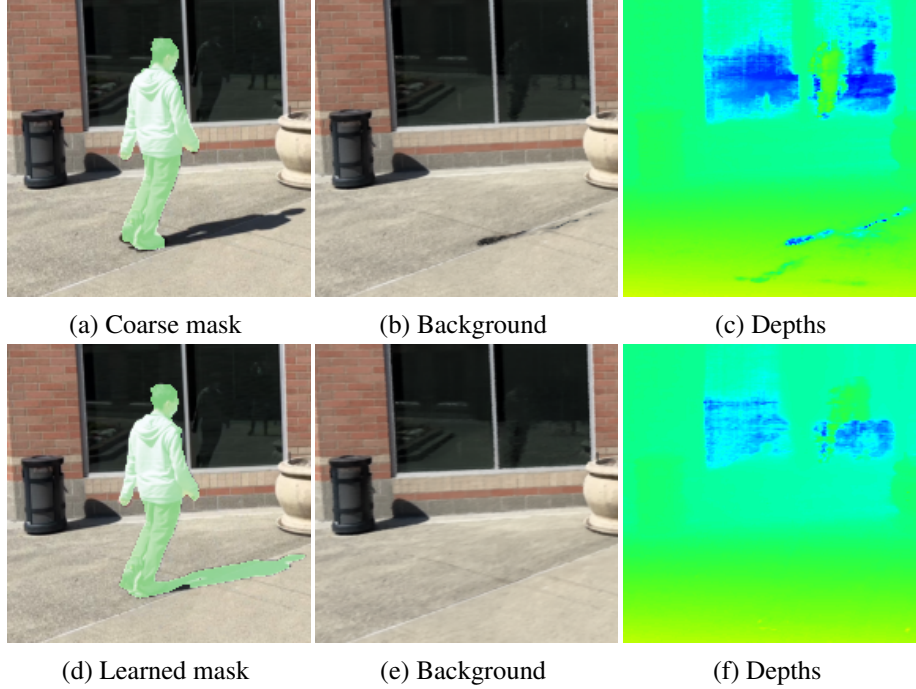


Figure 2.8: **Clean Background Retraining.** Background layers jointly trained can capture the shadows as a hole on the ground (a-c). After the joint training, the foreground *omnimatte* provides a better mask that can be used to train a clean background (d-f).

2.4.6 Ablation Studies

2.4.6.1 Loss Terms

We present background reconstruction results without $\mathcal{L}_{\text{depth}}$ in Fig. 2.7. For video sequences with rotational camera poses, the model struggles to extract 3D information from the input videos because of a lack of 3D clues. This loss is critical to extending our method to a broader range of videos. The effects of $\mathcal{L}_{\text{flow}}$ are also demonstrated in Fig. 2.7. The auxiliary task improves foreground quality and reduces unrelated content.



Figure 2.9: **Limitations.** (a), (b): Background can have baked-in shadows when the region is covered by shadows for most frames of the video. (c): The foreground layer captures irrelevant object motions (middle left) in the background. Best viewed in videos.

2.4.6.2 Clean Background Retraining

We employ an additional step for real-world sequences to optimize a clean background from scratch. In Fig. 2.8, we compare the background layer from the initial joint optimization and the final result. This is a simple yet robust way to obtain a better background.

2.4.7 Limitations

We list some limitations that future works can explore.

1. If a background region is covered by shadows nearly all of the time, the background model cannot recover its color correctly. An example from a `Movies` video is shown in Fig. 2.9. In theory, an *omnimatte* layer has an alpha channel and can capture only the additive shadow that allows the background to have the original color. However, this problem is largely under-constrained in the current setting, making it ambiguous and leading the background to unsatisfying solutions.
2. The foreground layer captures irrelevant content. In real-world videos, unrelated motions often exist in the background, like swaying trees and moving cars. These effects cannot be modeled by the static radiance field and will be captured by the foreground layer regardless

of their association with the object. Possible directions include i) using a dummy 2D layer to catch such content or ii) a deformable 3D background model with additional regularization to address the ambiguity as both background and foreground can model motion.

3. Foreground objects may have missing parts in the *omnimatte* layers if they’re occluded. Since our foreground network predicts pixel values for alpha composition, it does not always hallucinate the occluded parts.
4. The video resolution is limited. This is primarily due to the U-Net architecture of the foreground model inherited from Omnimatte. Higher resolutions can potentially be supported with the use of other lightweight image encoders.
5. The foreground layer may capture different content when the weights are randomly initialized differently. We include visual results in the supplementary materials.

2.5 Conclusion

We propose a method to obtain *omnimattes*, RGBA layers that include objects and their associated effects by combining 2D foreground layers and a 3D background model. Extensive experiments demonstrate that our approach is applicable to a wide variety of videos, expanding beyond the capabilities of previous methods.

2.6 Additional Qualitative Results

Video files of results presented in the main paper (all videos from the *Movies*, *Kubrics*, *Wild*, and *DAVIS* datasets) are available on our project website as part of the supplementary material. We highly recommend watching them on: <https://omnimatte-rf.github.io>.

1. For our method (OmnimatteRF), we include results (inputs with masks, foreground layers, background layer, background depth map) for every video.
2. For D²NeRF [127], we use the best result among all configurations provided by the authors for every video. If none of the configurations successfully reconstruct non-empty static and dynamic layers, we drop the video files and only show a frame in Fig. 2.11.
3. For Omnimatte [75] and Layered Neural Atlas (LNA) [56], we include videos from `Wild`, `Movies`, and `Kubrics`. Results of DAVIS can be found in prior works.

2.7 Random Initialization

As is also discussed in Omnimatte [75], different random initializations can lead to varying results of the foreground layers. We show two examples in Fig. 2.10.

In all our experiments, the random seed is set to 3.

2.8 Additional Implementation Details

2.8.1 Mask Generation

Our method and Omnimatte relies on coarse mask videos that outlines every object of interest. The synthetic `Kubrics` and `Movies` videos have ground truth object masks and we use them directly. To obtain mask for an in-the-wild video, we use one of the two workflows:

1. We first process the video a the pretrained Mask R-CNN model (X101-FPN) from Detectron 2 [128]. Then, we manually select a mask in every frame that best capture the

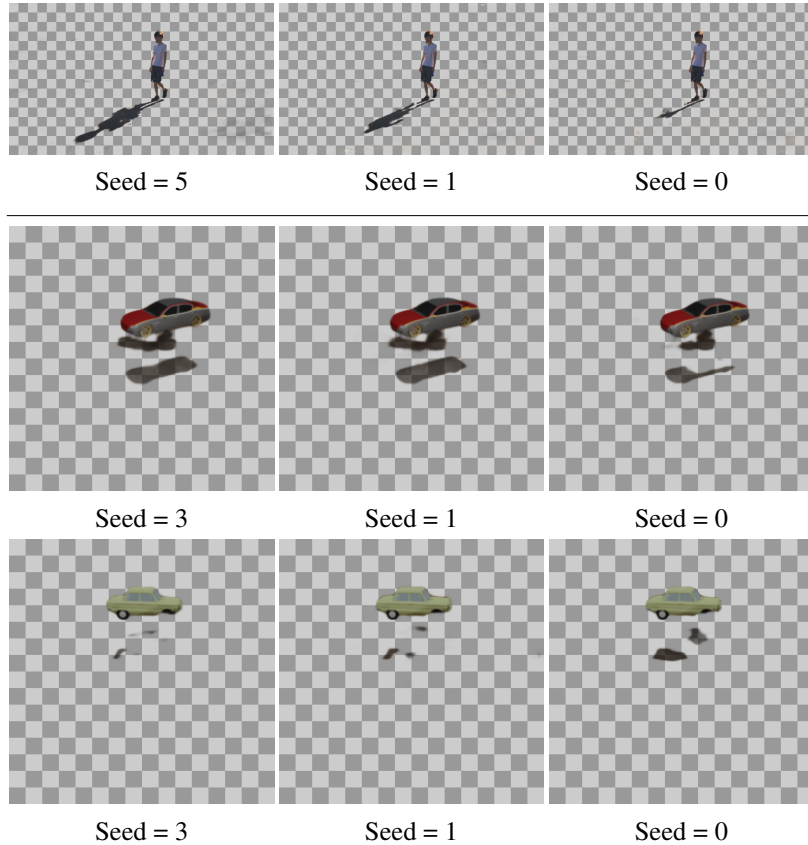


Figure 2.10: **Effect of random initialization.** Top: for the *Wild/dogwalk* video, different seeds lead to different amount of hallucinated shadow of the person. Bottom: for the *Kubrics/cars* video, seeds influence how shadows are associated to the objects.

object.

2. We use the Roto Brush tool in Adobe After Effects to track the object. This method is useful when Mask R-CNN fails produce good masks for a video. In particular, we processed *Wild/dance* and *Wild/solo* manually.

It takes about 10 minutes of manual work to generate a mask sequence for a 200-frame video.

2.8.2 Network Architecture

Our foreground network is based on the U-Net architecture of Omnimate, which is detailed in their supplementary [75]. To adopt their network to OmnimateRF, we replace the background noise input by the 2D feature map E_t . Each pixel in E_t is the positional encoding of the 3D vector (x, y, t) where (x, y) is the pixel location and t is the frame number. The positional encoding scheme is the same as proposed in NeRF [79], with $L = 10$ frequencies.

For background, we use the Vector-Matrix decomposition model in TensorRF [22] with the MLP feature decoder. Our initial grid has the same resolution $N_0 = 128$, and the final grid is limited to $N = 640$. The vectors and matrices are upsampled at steps 2000, 3000, 4000, 5500.

2.8.3 Hyper-parameters

For all videos, we use a learning rate of 0.001 for the foreground network, which is exponentially decayed from the 10,000 step at a rate of $0.1 \times$ per 10,000 steps. We find the decay crucial in preventing the foreground training from diverging. The mask bootstrapping loss $\mathcal{L}_{\text{mask}}$ has an initial weight of 50, which is first reduced to 5 when the loss value (before weighting) drops to below 0.02, and then turned off after the same number of steps. We document weights of other loss terms in Table 2.2.

Background network learning rate scheduling and $\mathcal{L}_{\text{bg-reg}}$ weight are identical as the original TensorRF [22].

In general, we use the same set of hyper-parameters for most videos, and only add additional terms when artifacts are observed.

Video	Steps	$\mathcal{L}_{\text{recons}}$	$\mathcal{L}_{\alpha\text{-reg}}$	$\mathcal{L}_{\alpha\text{-warp}}$	$\mathcal{L}_{\text{flow}}$	$\mathcal{L}_{\text{depth}}$	$\mathcal{L}_{\text{distort}}$
All	15,000	1	0.01 (L1) / 0.005 (L0)	0.01	1	0	0
Wild/bouldering	-	-	-	-	-	0.1	0.01
DAVIS	10,000	-	-	-	-	1	0

Table 2.2: **Hyper-parameters.** We document the hyper-parameters (number of steps and weights of loss terms) in our experiments. The first row is the configuration shared by most videos. Remaining rows are videos with different configurations, and - means unchanged from the shared number.

Method	Steps	Training (hours)	Rendering (s/image)
Omnimatte	12,000	2.7	2.5
D ² NeRF	100,000	4.5	4.8
LNA	400,000	8.5	0.40
Ours	15,000	3.8	3.5
Omnimatte	12,000	1.2	0.95
D ² NeRF	100,000	3.2	2.2
LNA	400,000	8.5	0.21
Ours	15,000	2.5	1.2

Table 2.3: **Running Time Measurement.** We measure and compare the time it takes to train OmnimatteRF and baseline methods. **Top:** Movies, Wild (480×270 , DAVIS has a similar resolution of 428×240). **Bottom:** Kubrics (256×256).

2.8.4 Running Time Measurement

We measure and report the time it takes to train OmnimatteRF and baseline methods in Table 2.3. All measurements are conducted on a workstation with an eight-core AMD R7-2700X CPU and a single NVIDIA RTX3090 GPU.

Our method takes longer to train than Omnimatte due to the addition of the 3D background radiance field.

Optimizing the background model only, as in the clean background retraining process, takes about 30 minutes per video.

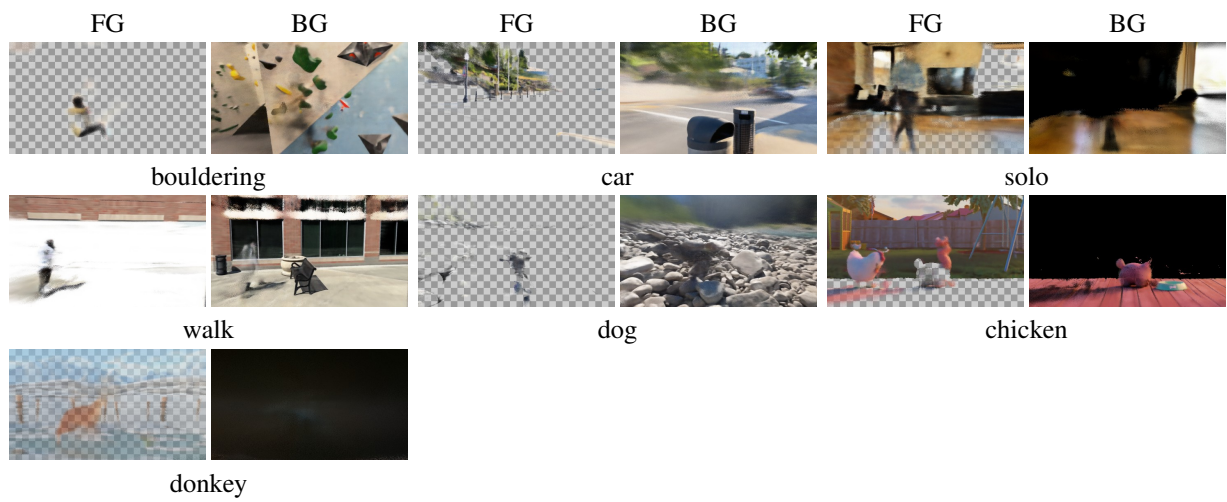


Figure 2.11: **D²NeRF** results for failed scenes.

Chapter 3: Few View 3D Gaussians with Feed-Forward Geometry Model and Generative Priors

3.1 Introduction

Image-based 3D reconstruction and novel view synthesis is a core problem in computer graphics and 3D computer vision that has seen rapid progress recently. This has been driven by powerful scene representations including point-based [1, 57] and neural techniques [79], in combination with differentiable rendering techniques. Together, these frameworks enable straightforward, gradient-based optimization of the scene parameters to match the given input images, and recent work has demonstrated impressive accuracy in recovered 3D scene geometry and rendered novel views.

Optimizing the scene parameters solely to match the given input images, however, suffers from overfitting issues. In scenarios where only a sparse set of input images is available, this leads to inaccurate 3D geometry reconstruction and artifacts in novel views rendered from viewpoints that deviate significantly from the given inputs. Numerous techniques have been proposed to address these challenges by introducing additional regularization and loss terms, for example to constrain the density in volumetric scene representations, surface normals, or rendered depth [32, 58, 63, 130].



Figure 3.1: We propose a scene reconstruction pipeline from sparse inputs. It leverages feed-forward geometry models and generative priors to densify the input views with pseudo ground truth images from interpolated views. With approaches to handle inconsistencies from generated views, we our method achieves high quality reconstruction.

One of the most powerful approaches to date is to leverage conditional generative image and video models as priors for novel views, where the generative models are typically implemented using diffusion models. Image and video diffusion models can be “lifted” from 2D to 3D in at least two different ways: diffusion updates can be back-propagated to 3D iteratively during reverse diffusion using techniques such as score distillation sampling (SDS) and related methods [90, 123]. Alternatively, novel views predicted by diffusion models can simply be added as “pseudo-ground truth” to the given input images and included in the usual scene optimization process. In practice, the latter approach is more efficient and has proven effective in improving reconstruction quality [71, 103, 107].

In this paper, we propose a novel approach to leverage diffusion models to improve image-based 3D reconstruction and view synthesis. Similarly to related techniques [71], we use a video diffusion model as the generative model to generate pseudo-ground truth images, which are then added to the set of training images to regularize scene reconstruction. However, we make several key contributions to increase the effectiveness of this basic approach. First, prior work uses

images rendered from an initial scene reconstruction to condition the diffusion model. This requires a time consuming two-stage optimization procedure. Instead, we leverage a powerful and highly efficient feedforward model [113] both to obtain an initial scene reconstruction and also to produce the conditioning images for the diffusion model. This leads to a simpler and faster training pipeline. In addition, we observe that, while images generated by the diffusion prior generally look visually accurate, they typically suffer from geometric inconsistencies. We address this issue by introducing an optimizable image warp that provides additional degrees of freedom and alleviates the inconsistencies in the pseudo-ground truth views. Finally, our pipeline includes a color adjustment approach and a robust strategy to sample training views from the set of input and pseudo-ground truth views.

We demonstrate that with our proposed pipeline, high-quality scene reconstruction can be achieved with as few as three input images. Our code and models will be made publicly available upon acceptance to facilitate future research.

3.2 Related Works

Sparse Novel View Synthesis Novel View Synthesis is a classic problem in computer vision and computer graphics, where the goal is to synthesize images for novel viewpoints given a set of multi-view input images. Traditionally, this task was addressed by Multi-View Stereo (MVS) methods, which first reconstruct dense 3D geometry from the input views and then project the 3D representation into the target views. Classical MVS methods [35, 98, 101] often struggle to recover geometry in textureless regions, and learning-based approaches [21, 40, 137, 138, 148, 154] have been proposed to alleviate their limitations.

In recent years, NeRF [79] has demonstrated remarkable performance in novel view synthesis and gained popularity. Follow-up works have extended NeRF in various aspects, like anti-aliasing [8, 10, 11] and fast training [34, 82, 105, 145].

Despite these improvements in quality and efficiency, NeRF methods still struggle to achieve real-time rendering, which limits their practical applicability. To address this, 3D Gaussian Splatting (3DGS) [57] proposes to represent scenes using anisotropic 3D Gaussian primitives and achieves real-time rendering by rasterization. It has also gained much attention and extensions in various directions. For example, [42, 51, 149] improve surface reconstruction quality, [147] enhances its anti-aliasing capabilities, and [33, 83] compress the 3DGS representation to reduce memory usage.

Among them, the one direction that our work concerns is: how to reconstruct high quality 3D scenes from sparse inputs? 3DGS typically relies on densely captured images, and many recent works have attempted to address this challenge. Specifically, MVSplat [24] directly predicts accurate 3D Gaussian parameters from sparse multi-view images using cost volume features. DNGaussian [63], SparseGS [130] and [29] introduce depth-based constraints to mitigate geometry degradation. FSGS [156] improves sparse-view representations through proximity-guided Gaussian unpooling and virtual view synthesis. CoR-GS [151] proposes a co-regularization strategy to detect and suppress unreliable reconstructions. InstantSplat [32] jointly optimizes scene geometry and camera poses using geometric priors. DIFIX3D+ [125] proposes a single-step image diffusion model to eliminate artifacts in underconstrained regions. The state-of-the-art and most closely related to our work is 3DGS-Enhancer [71], which leverages video diffusion priors to enforce view consistency and refines rendered novel views through spatial-temporal feature integration. A key difference that distinguishes our work from 3DGS-Enhancer is that we condi-

tion the diffusion model using an initial feedforward scene prediction, rather than by requiring a time consuming two-stage optimization procedure.

Handling Distractors Some recent works [7, 31, 97, 122, 131, 150] focus on improving robustness of 3DGS reconstruction from inconsistent multiview datasets, such as transient objects, occlusions, appearance and lighting changes. [122] introduce a residual spherical harmonic module to adapt to lighting conditions and jointly predict transient occlusions and appearance using spatial attention. Similarly, [131, 150] model appearance with separate intrinsic and dynamic components or material and illumination attributes, while leveraging adaptive sampling, visibility maps, and depth regularization to address local details and transient occlusions. [31, 97] target unstructured, in-the-wild photo collections by incorporating pre-trained features and introducing transient Gaussians or robust optimization techniques to mitigate distractions such as motion and lighting variations. [7] focus on distractor robustness through mask-based training and a two-stage inference pipeline. Due to the structural inconsistency of generated videos from SVD, we develop an image warping module to handle the inconsistencies to generate high-quality 3DGS.

Video Diffusion Models Video Diffusion models have emerged as a powerful paradigm for generating temporally coherent 3D structures. Initial approaches, such as [49], adopt a standard diffusion setup to generate text-conditioned and unconditional videos extending the 2D-Unet to work for 3D data more suited towards video modeling. Imagen Video [48] used a cascade of diffusion models, including image and temporal super-resolution models to improve the resolution and temporal coherence. Recent works have shifted towards a latent diffusion framework [95]. In this paradigm, a VAE is first used to compress data to a lower dimension latent space, where the

diffusion process directly operates. Video LDM, built upon this idea for videos [15], adds temporal layers to an existing latent image diffusion model while further improving the decoder for temporal consistency. Stable Video Diffusion (SVD) [14], used the same architecture as Video LDM, focusing on dataset curation and a systematic training framework including large scale, image and text pre-training and high quality video fine-tuning to produce both image and text conditioned video generation.

The rapid advancement in video diffusion models has led to several approaches using these models as priors to improve novel view synthesis, especially for sparse inputs. SVD [14], V3D [112], and IM-3D [77] explore directly using camera controlled video diffusion models for simple object-level novel view generation. 3DGS-enhancer [71] and MVSplat-360 [25] further integrate video diffusion models, specifically SVD [14] for more complex scene-level novel view synthesis from sparse views. 3DGS-enhancer [71] however requires training a coarse 3DGS model to render novel views from 3DGS, which are directly used to condition the video diffusion model. MVSplat-360 [25], uses MVSplat [24] to generate coarse 3DGS features for conditioning instead. Our approach instead uses images rendered through a fast feedforward model, simplifying the conditioning for SVD and reducing computational overhead.

Feedforward Geometry Models Although earlier 3D representation methods like NeRF [79] or 3D Gaussian Splatting [57] have achieved impressive results, they typically require known camera information and per-scene optimization. On the other hand, feedforward approaches leverage priors learned from large-scale datasets to directly and quickly infer 3D attributes from uncalibrated images. DUST3R [118] formulates the pairwise reconstruction task as pointmap regression and demonstrates strong performance. It has become the basis of many follow-up

works. MAST3R [62], Fast3R [133] and MUST3R [20] improve its speed and scalability to larger image sets.

Another line of works develop geometry understanding based on pixel tracking cues [114, 115]. Visual Geometry Grounded Transformer (VGGT) [113] tokenizes the images and use multiple decoding heads to simultaneously produce various 3D attributes like camera poses, depth maps, and point clouds. Its architecture enables fast inference, processing dozens of images in a matter of seconds. Given its efficiency and accuracy, we adopt the VGGT model to estimate dense point clouds, for conditioning the video diffusion model and initializing the 3DGS reconstruction.

3.3 Method

We outline our method in Figure 3.2. On a high level, our pipeline starts by densifying the input images from less than ten to the magnitude of hundreds. To achieve this, we use a feed-forward geometry model [113] to estimate a dense point cloud from the input images. Then, a video diffusion model is conditioned on the view-consistent point cloud renderings to generate novel images from interpolated camera locations (Section 3.3.1). Using the mixed set of sparse real and dense generated images, we optimize 3D Gaussians to reconstruct the scene. As the diffusion process inevitably introduces inconsistencies, we include adaptive warping (Section 3.3.2), color adjustment maps (Section 3.3.3), and biased sampling towards real images (Section 3.3.5) in our pipeline to improve the resulting reconstruction. In the following, we describe each of our proposed components in detail.

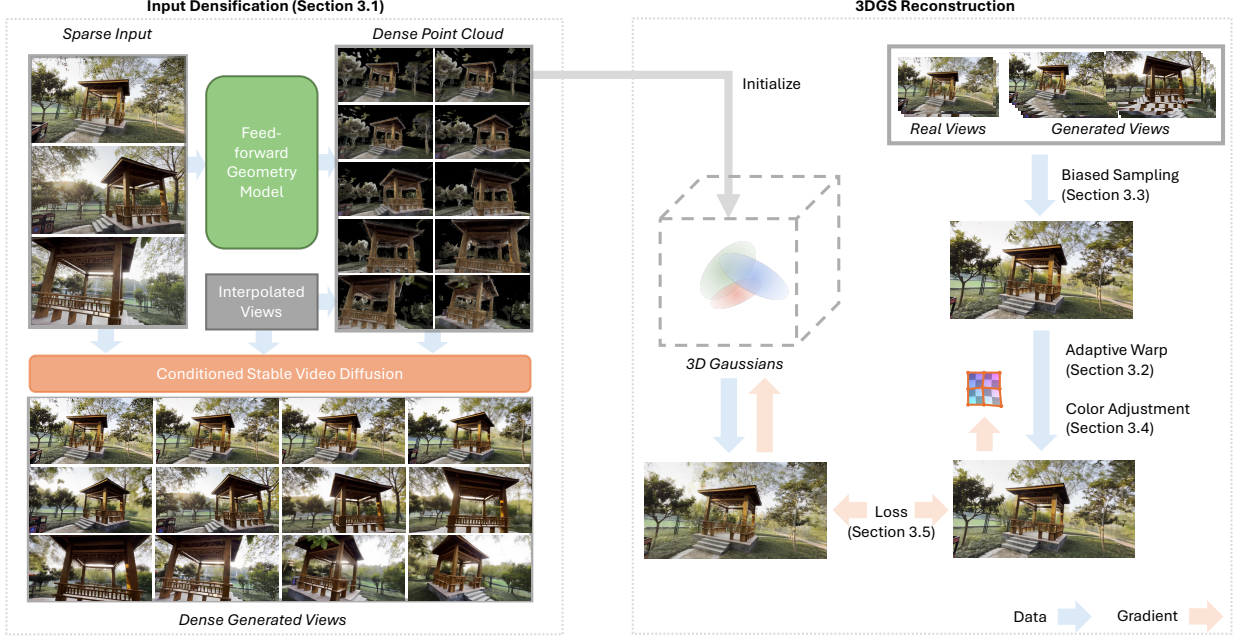


Figure 3.2: Overview of our pipeline. **Left:** we first densify the sparse input images. A feed-forward geometry model is used to estimate a dense point cloud, which is then rendered from interpolated views. The point cloud renderings are used to condition a video diffusion model that generates dense pseudo ground truths. **Right:** we initialize 3DGS from the dense point cloud, and optimize it with both input and generated views. We employ a sampling strategy biased towards real views, as well as learnable image warps and color adjustments to handle the inconsistencies in generated views.

3.3.1 Enhancing Sparse Input with Geometry Model and Generative Prior

Given the sparse input images $\mathcal{I}_{\text{sparse}}$, the first step is to obtain a dense image set $\mathcal{I}_{\text{gen}}$ from interpolated view points with generative priors, which are subsequently mixed with the input images to form the enhanced training set $\mathcal{I}_{\text{dense}} = \mathcal{I}_{\text{sparse}} + \mathcal{I}_{\text{gen}}$. Inspired by 3DGS-Enhancer [71], we use a video diffusion model to produce dense interpolations. To encourage view consistency across interpolated views, 3DGS-Enhancer trains a coarse 3DGS model from the sparse inputs and uses its renderings from novel cameras as conditions for the diffusion model. However, this process is time consuming, and the coarse 3DGS model often produces hardly legible images for very sparse inputs.

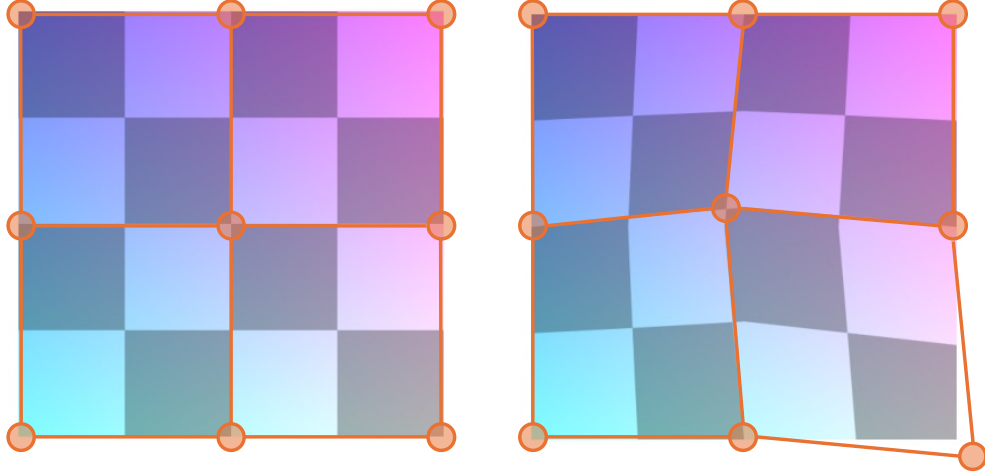


Figure 3.3: Adaptive warping illustration. The figure is an example of a warp grid with resolution $r_{\text{warp}} = 3$, in which $3 \times 3 = 9$ control points are optimized. Left: initialization with all offsets set to zeros. Right: a possible outcome with some control points moved and the image distorted.

To alleviate these issues, we instead utilize a feed-forward geometry model that estimates a dense point cloud P from the input images $\mathcal{I}_{\text{sparse}}$. The point cloud is then rendered from the interpolated views to produce geometrically consistent condition images for the diffusion model. To render the point cloud, we utilize the existing pipeline and create a Gaussian centered at each point, with the same color and an empirically set uniform scale $\mathbf{s} = (0.01, 0.01, 0.01)$. As shown in Figure 3.6, our approach produces high-quality conditions while reducing the required time from dozens of minutes to just a few seconds.

For this paper, we use the visual geometry grounded transformer (VGGT) [113] to estimate relative camera poses and depth maps for each image. We can then back-project the pixels with the depths to obtain the point cloud.

3.3.2 Adaptive Warps

Despite the effort in conditioning, the diffusion process is stochastic and inevitably introduces geometric inconsistencies in the generated images. Moreover, since the diffusion U-Net

operates on image latents that are spatially low resolution (usually $1/8$ of the image resolution), it can be challenging to produce perfectly consistent images. Figure 3.4 shows how the buildings and structures in the generated images can be slightly inaccurate in their positions.

To this end, we propose to optimize a piecewise linear warp for every generated image to compensate for the inconsistency. Each warp is represented as a low-resolution grid $O_i \in \mathbb{R}^{r_{\text{warp}} \times r_{\text{warp}} \times 2}$. Intuitively, we define a grid on the image with $r_{\text{warp}} \times r_{\text{warp}}$ control points, and the image can be distorted by moving these points, shown in Figure 3.3.

Image warping can be implemented with grid sampling. The grid is computed by upsampling O_i with linear interpolation to the same resolution as the image I_i . For numerical stability, O_i actually stores the offset, that is, the amount to move the grid control points, instead of their pixel locations. Therefore, to obtain the final grid, it is added to the identity grid $\mathfrak{I}$ where $\mathfrak{I}_{x,y} = (x, y)$. The warped images is computed as:

$$I_i^{\text{warp}} = \text{grid_sample}(I_i, \mathfrak{I} + \text{upsample}(O_i)) \quad (3.1)$$

We use bilinear interpolation for both upsampling the warp offset map O_i and grid sampling. The warp is jointly optimized with the 3D Gaussian parameters with image reconstruction loss.

3.3.3 Color Adjustment Map

We additionally employ per-view low-resolution color adjustment maps to mitigate minor color differences in the generated views introduced in the random process. Each adjustment

map is a low-resolution image $A_i \in \mathbb{R}^{r_{\text{adj}} \times r_{\text{adj}} \times 3}$ and initialized to zeros. A generated image is multiplied by an upsampled adjustment map for loss computation:

$$I_i^{\text{adj}} = (\mathbf{1} + \text{upsample}(A_i)) \otimes I_i^{\text{warp}} \quad (3.2)$$

The color adjustment map is also jointly optimized with the other model parameters. In addition, we employ a regularization term to limit the amount of adjustment, where $\mathbf{0}$ is a zero matrix:

$$\mathcal{L}_{\text{adj}} = \lambda_{\text{adj}} \text{L1}(A_i, \mathbf{0}) \quad (3.3)$$

3.3.4 Losses

The image reconstruction loss that we use is the same as the original 3DGS, but the generated pseudo-ground truths are processed as described above.

$$\mathcal{L}_{\text{recons}} = \lambda_1 \text{L1}(I_i^{\text{render}}, I_i^{\text{tgt}}) + \lambda_2 \text{SSIM}(I_i^{\text{render}}, I_i^{\text{tgt}}), \text{ where} \quad (3.4)$$

$$I_i^{\text{tgt}} = \begin{cases} I_i, & I_i \text{ is real,} \\ I_i^{\text{adj}}, & I_i \text{ is generated} \end{cases} \quad (3.5)$$

And the total loss is $\mathcal{L} = \mathcal{L}_{\text{recons}} + \mathcal{L}_{\text{adj}}$.

3.3.5 High Real View Ratio

The 3DGS training process shuffles all the training views and supervises the model with one view at a time until the entire queue is exhausted, when all views are shuffled again and the process is repeated. This ensures that all views are observed equally frequently in random order. However, this scheme may not be ideal when our training views consist of a combination of sparse real inputs $\mathcal{I}_{\text{sparse}}$ and dense generated images $\mathcal{I}_{\text{gen}}$, where $|\mathcal{I}_{\text{sparse}}| \ll |\mathcal{I}_{\text{gen}}|$. As the generated images are imperfect, using them for more than 90% of the time can lead to reconstructions that are biased toward errors.

One approach taken in prior works is to assign a weight per image and make the real inputs have larger effects on the supervision. However, this comes with the drawback that the learning process may be less efficient as the model is constantly supervised with weaker signals. We take a different path by sampling the real inputs more frequently. More precisely, every time the entire training set is shuffled, the sparse inputs are duplicated so that the *ratio* of real images exceeds a predefined parameter σ_{real} . Intuitively, with σ_{real} varying from 0 to 1, the reconstruction quality will first increase for observing more real information, then decrease as the generated views are effectively dropped. The details are further analyzed with the experimental results in Section [3.5.4](#).

Method	3 Views			6 Views			9 Views		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
Mip-NeRF [8]	10.92	0.191	0.618	11.56	0.199	0.608	12.42	0.218	0.600
RegNeRF [84]	11.46	0.214	0.600	12.69	0.236	0.579	12.33	0.219	0.598
FreeNeRF [134]	10.91	0.211	0.595	12.13	0.230	0.576	12.85	0.241	0.573
3DGS [57]	10.97	0.248	0.567	13.34	0.332	0.498	14.99	0.403	0.446
DNGaussian [63]	11.10	0.273	0.579	12.67	0.329	0.547	13.44	0.365	0.539
3DGS-Enhancer [71]	14.33	0.424	0.464	16.94	0.565	0.356	18.50	0.630	0.305
Ours	14.78	0.499	0.542	17.25	0.564	0.441	19.11	0.625	0.360
Ours w/ 3DGS-Enhancer SVD	14.28	0.495	0.556	16.97	0.564	0.449	18.85	0.620	0.374

Table 3.1: Sparse view synthesis results on the DL3DV dataset. Our method achieves on-par results with state-of-the-art. We additionally trains an SVD model following the setup in 3DGS-Enhancer and report the result of using this model in our pipeline in the last row.

Method	6 Views			9 Views		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
Mip-NeRF [8]	13.08	0.159	0.637	13.73	0.189	0.628
RegNeRF [84]	12.69	0.175	0.660	13.73	0.193	0.629
FreeNeRF [134]	12.56	0.182	0.646	13.20	0.198	0.635
3DGS [57]	11.53	0.144	0.651	12.65	0.187	0.607
DNGaussian [63]	11.81	0.208	0.689	12.51	0.228	0.683
3DGS-Enhancer [71]	13.96	0.260	0.570	16.22	0.399	0.454
Ours	15.22	0.383	0.569	16.58	0.422	0.501

Table 3.2: Sparse view synthesis results on the Mip-NeRF 360 dataset.

3.4 Experiments

3.4.1 Implementation

We implement our pipeline with PyTorch, building upon 3DGS-Enhancer [71] for the conditioned SVD model and 3DGS [57] for reconstruction. The 3DGS model is optimized for 30,000 steps with the same default hyperparameters, including $\lambda_1 = 0.8$ and $\lambda_2 = 0.2$. For our proposed components, we use $r_{\text{wrap}} = 8$, $r_{\text{adj}} = 8$, $\sigma_{\text{real}} = 0.8$, and $\lambda_{\text{adj}} = 0.1$ in all experiments. The learned warp and adjustment maps have the same learning rate of 10^{-4} .

3.5 SVD Model Training

We use the code released by 3DGS-Enhancer [71] to train SVD models with point cloud rendering conditions and 3DGS rendering conditions. We empirically find that the learning rate (10^{-5}) and number of iterations (100K) in their code result in better models than the hyperparameters (10^{-4} , 80K) in their manuscript. For both SVD model variants, we use the entire DL3DV-1K dataset with 1000 scenes, except the 20 scenes reserved as test set.

3.5.1 Evaluation

Following 3DGS-Enhancer [71], we evaluate our method with their 20-scene subset from DL3DV as well as the Mip-NeRF 360 dataset. Quantitative comparisons are presented in Table 3.1 and Table 3.2 respectively. Our method achieves on-par performance with state-of-the-art while requiring less computation.

Qualitative results of 3, 6, and 9 input views are in Figures 3.6, 3.7, and 3.8. For sparse input, our proposed approach to generate condition images from dense point cloud estimations produces more reliable, consistent, and informative images than the 3D Gaussian models trained with sparse images.

3.5.2 SVD Model Comparison

To evaluate the effectiveness of our proposed condition images for the SVD model, we additionally train an SVD model using the code release of 3DGS-Enhancer [71], with our best effort to follow their original configuration. We present the results in Table 3.3.

In addition, we conduct experiments on the DL3DV dataset using images from the repro-

Cond.	3 Views			6 Views			9 Views		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
3DGS-E.	12.29	0.381	0.545	14.91	0.456	0.451	16.57	0.505	0.394
Ours	12.56	0.385	0.532	14.57	0.432	0.454	15.81	0.467	0.403
$\{I^{\text{adj}}\}$	13.27	0.416	0.524	15.70	0.483	0.441	17.05	0.524	0.390

Table 3.3: Comparison of SVD models with different conditions. We optimize an SVD model using the code and settings of 3DGS-Enhancer [71] and compare it with ours. The metrics are computed for all interpolated training views between SVD outputs and ground truths. In addition, the last row computes the metrics with our warped and color-adjusted images. The processed images are more faithful to ground truth, meaning that they are better training inputs than SVD outputs.

Method	Initial Render SVD Train Total				
3DGS-Enhancer	643	61	1018	-	-
Ours	2	50	1018	1119	2189

Table 3.4: Time analysis. All numbers are seconds, averaged over the 9-view DL3DV scenes. Initial model is a coarse 3DGS model for 3DGS-Enhancer, and VGGT inference for ours. SVD inference is measured on an H100 GPU, while all other steps are measured on a RTX A5000 GPU.

duced model in the last row of Table 3.1. While our SVD model predictions are slightly worse in metrics, the performance is better when used for downstream 3DGS training.

3.5.3 Time Analysis

We compare the time used in each stage of 3DGS-Enhancer [71] and our method in Table 3.4. Thanks to the use of an efficient feed-forward model, we removed the time spent on training the initial coarse model entirely. Note that the full 3DGS training time for their method is not available as their code has not been released at the time of writing.

Method	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
w/o adaptive warping (3.3.2)	18.66	0.604	0.385
w/o real view ratio (3.3.5)	17.33	0.544	0.451
w/o color adjustment (3.3.3)	18.99	0.619	0.370
Homography warping	19.07	0.623	0.372
Ours	19.11	0.625	0.360

Table 3.5: Ablation study. We disable the components in our pipeline one at a time to demonstrate their effectiveness. In addition, we experiment with a homography-based warping instead of the proposed grid sampling offset method.

3.5.4 Ablation Study

We demonstrate the effectiveness of each of our proposed components in Table 3.5 by removing each of them from the pipeline. The proposed image warping and color adjustments are also further evaluated in Figure 3.4 and Table 3.3, where the processed images are shown to be closer to ground truth.

Notably, we find that assigning a high ratio to real views (σ_{real}) is critical in reconstruction quality. We further analyze the effect in Figure 3.5, and find that the best value is as high as using real views for 80% of training. This is likely because of the inconsistency and hallucination from the diffusion sampling process. The effect of σ_{real} and its ideal value may both decrease in the future when better consistency is achieved with novel image generation priors.

3.5.5 Image Warp with Homography

We also experiment with an alternative to the image warping method proposed in Section 3.3.2. This approach splits each generated image into $r_{\text{warp}} \times r_{\text{warp}}$ blocks. For each block, seven homography parameters are optimized: 2D scale (s_x, s_y) , 2D shear (v_x, v_y) , 2D translation (t_x, t_y) , and 1D rotation θ . The transformation without translation is defined as the following

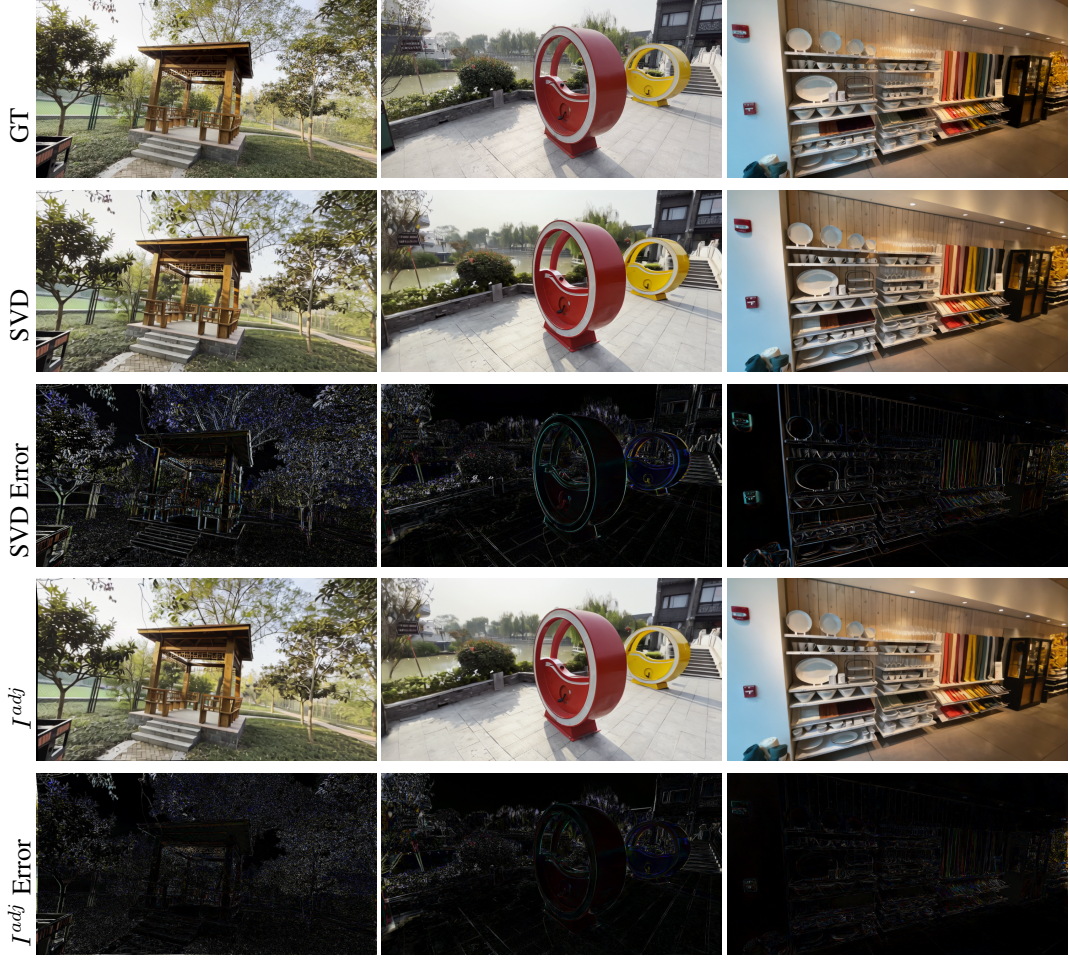


Figure 3.4: Effects of proposed warp and color adjustment. We compare SVD output I_i (2nd row) with the warped and color-adjusted image I_i^{adj} (4th row) after training. For better visualization we also show their absolute error on the 3rd and 5th rows. The high error regions at the boundary of the buildings and objects due to geometric inconsistency in the SVD output are effectively mitigated in the processed images.

matrix:

$$H = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} 1 + s_x & v_x \\ v_y & 1 + s_y \end{pmatrix}. \quad (3.6)$$

To warp the image, each block is processed separately. Every pixel location $p = (x, y)$ in the block is mapped to a warped coordinate in the image:

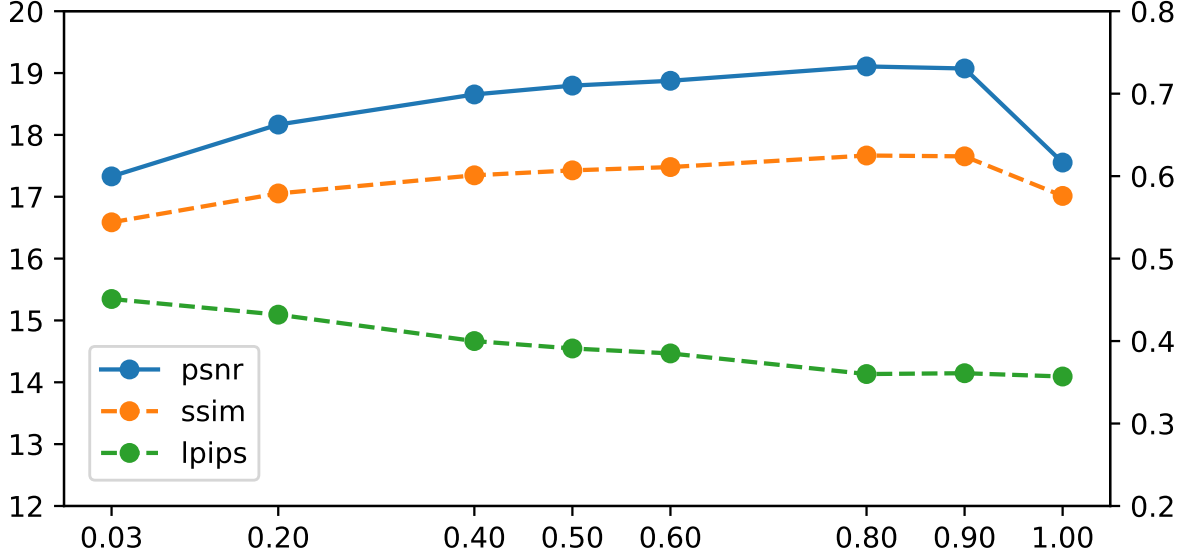


Figure 3.5: Real view ratio analysis. Increasing the ratio of real inputs improves reconstruction results until $\sigma_{\text{real}} = 0.8$. The smallest ratio $\sigma_{\text{real}} = 0.03$ corresponds to not using the proposed sampling strategy (i.e. the ratio between input and generated images is approximately 1 to 30), and $\sigma_{\text{real}} = 1$ means not using generated images at all. The results are from the DL3DV 9-view test set.

$$p' = Hp + (t_x, t_y). \quad (3.7)$$

The warped coordinates are used in grid sampling to obtain a warped version of the block.

This approach is a more constrained version of the proposed offset maps to grid sampling, as it limits the warps to affine transformations. As we show in Table 3.5, it produces slightly worse reconstruction. This is possibly due to the inconsistencies in diffusion model samples being arbitrary and require a more flexible image warp method.

3.5.6 Diffusion Model for Post Processing

Since the video diffusion model trained in 3DGS-Enhancer [71] is capable of enhancing the quality of 3D Gaussian renderings, a natural idea is to use it as a post-processing tool for

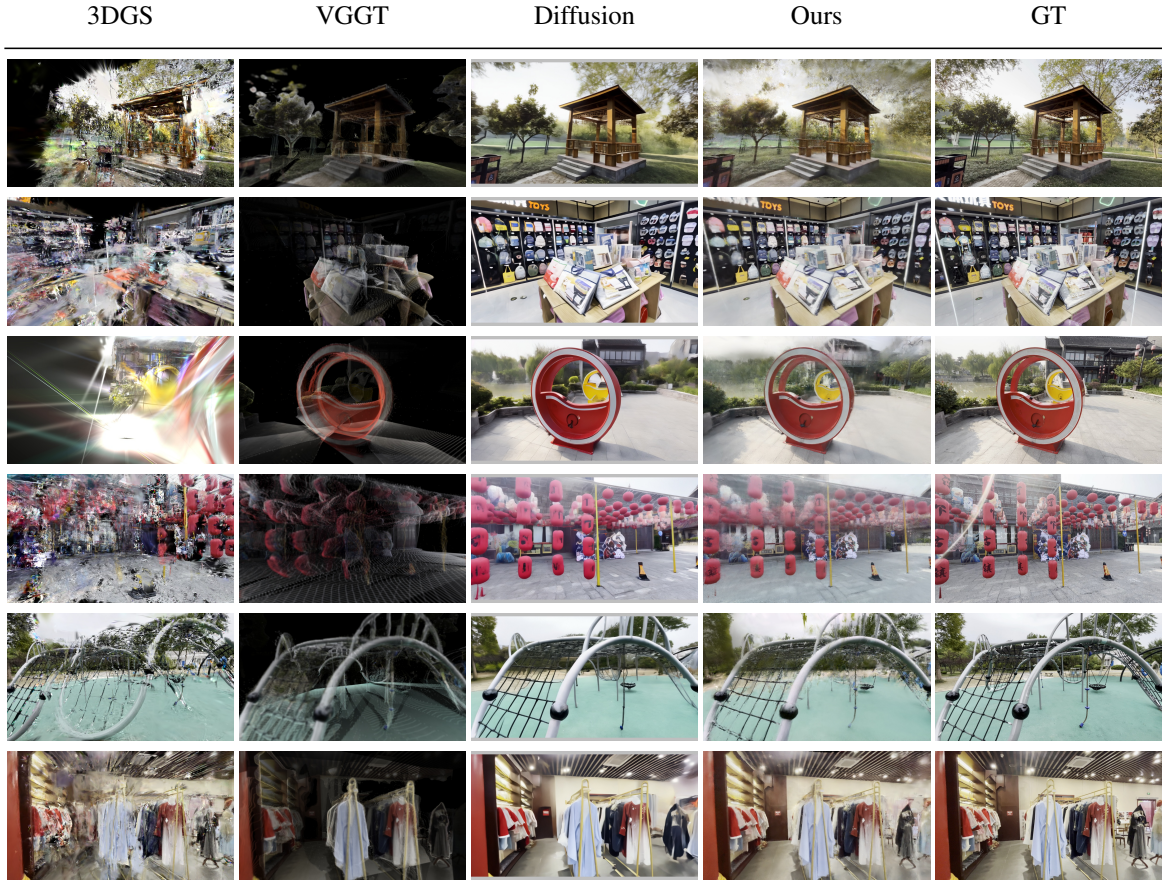


Figure 3.6: Qualitative results of 3 input views. The first two columns compare the two initial models: 3DGS from sparse views and dense point cloud from VGGT. With very sparse inputs, VGGT produces more coherent results than 3DGS. The 3rd column shows our fine-tuned SVD model output conditioned with the VGGT point cloud renderings. Our 3DGS model trained with the densified images achieves high quality reconstruction (4th column).

novel views rendered from the 3DGS model. However, it does not immediately work, and we observed two key issues:

1. Geometry inconsistencies still exist in the samples, even when the input images are relatively high quality and consistent.
2. When a large number of novel views are produced, the video diffusion process must be run in batches, and the consistency between batches is poor.

As a result, the naive approach does not produce better images than the 3DGS model.

Additionally, we experiment with training a second 3DGS model from the above samples. However, due to the lack of significant improvement in video diffusion samples, this iterative approach does not consistently improve reconstruction.

We leave the exploration of these directions to future research.

3.6 Limitations

The most important limitation of our approach is the computational cost and instability of stable video diffusion, the image generative prior that we leverage. Processing a batch of 26 images at half-HD resolution (960×512) requires 56GB of VRAM and a minute on an H100. Despite the condition images provided, its output is far from consistent. Future research on consistent image generation priors that are more computationally affordable will be beneficial for this problem.

3.7 Conclusion

We propose a novel pipeline for sparse input scene reconstruction with 3D Gaussians. Our key insight is to utilize feedforward geometry models for efficient generation of dense point clouds as conditions for image generation priors and dense initialization. To deal with inconsistencies of generated pseudo-ground-truth images, we propose to learn image warps, sample real views more frequently, and use color adjustment maps. Experimental results show that our pipeline achieves state-of-the-art reconstruction quality from sparse input images.

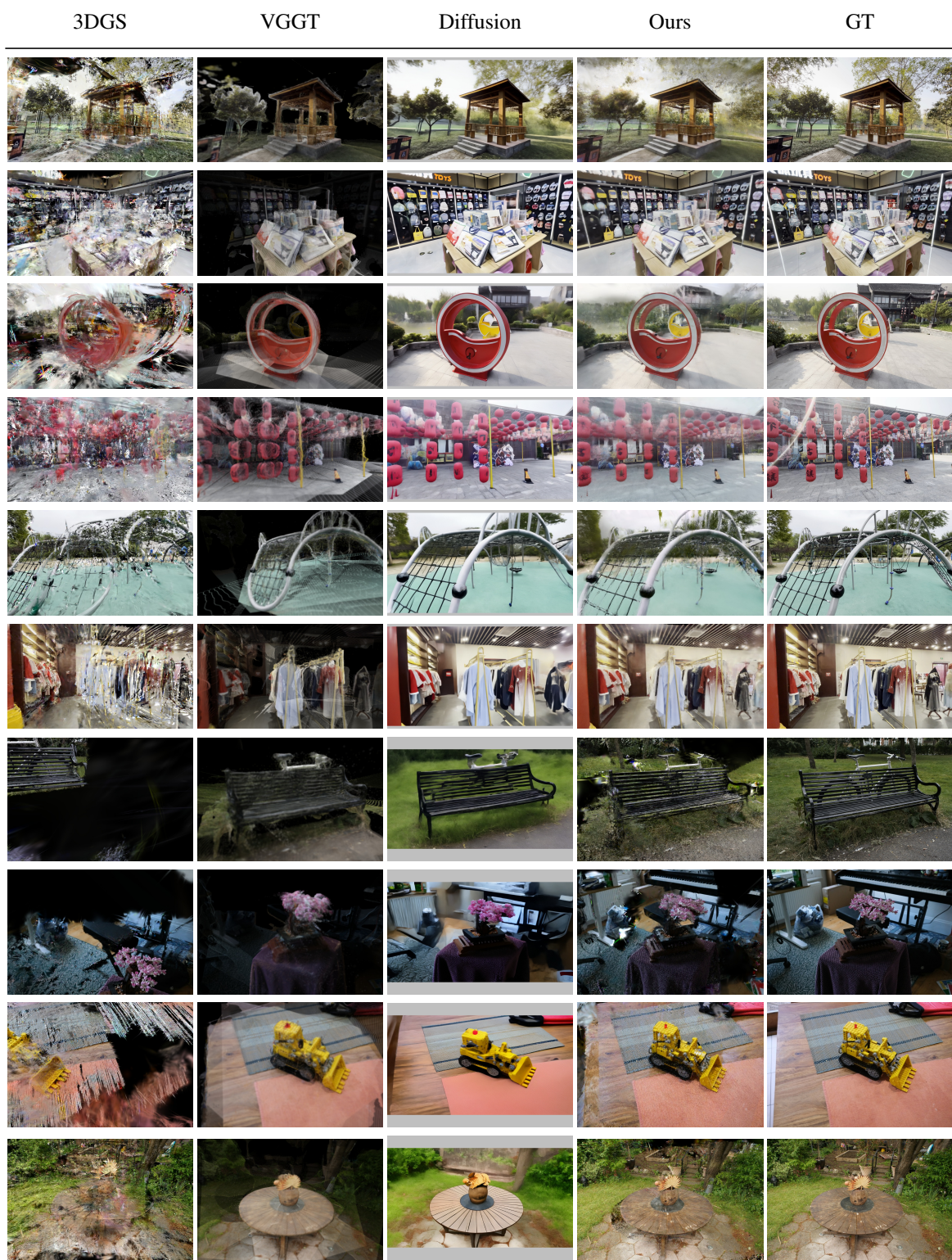


Figure 3.7: Qualitative results of 6 input views.

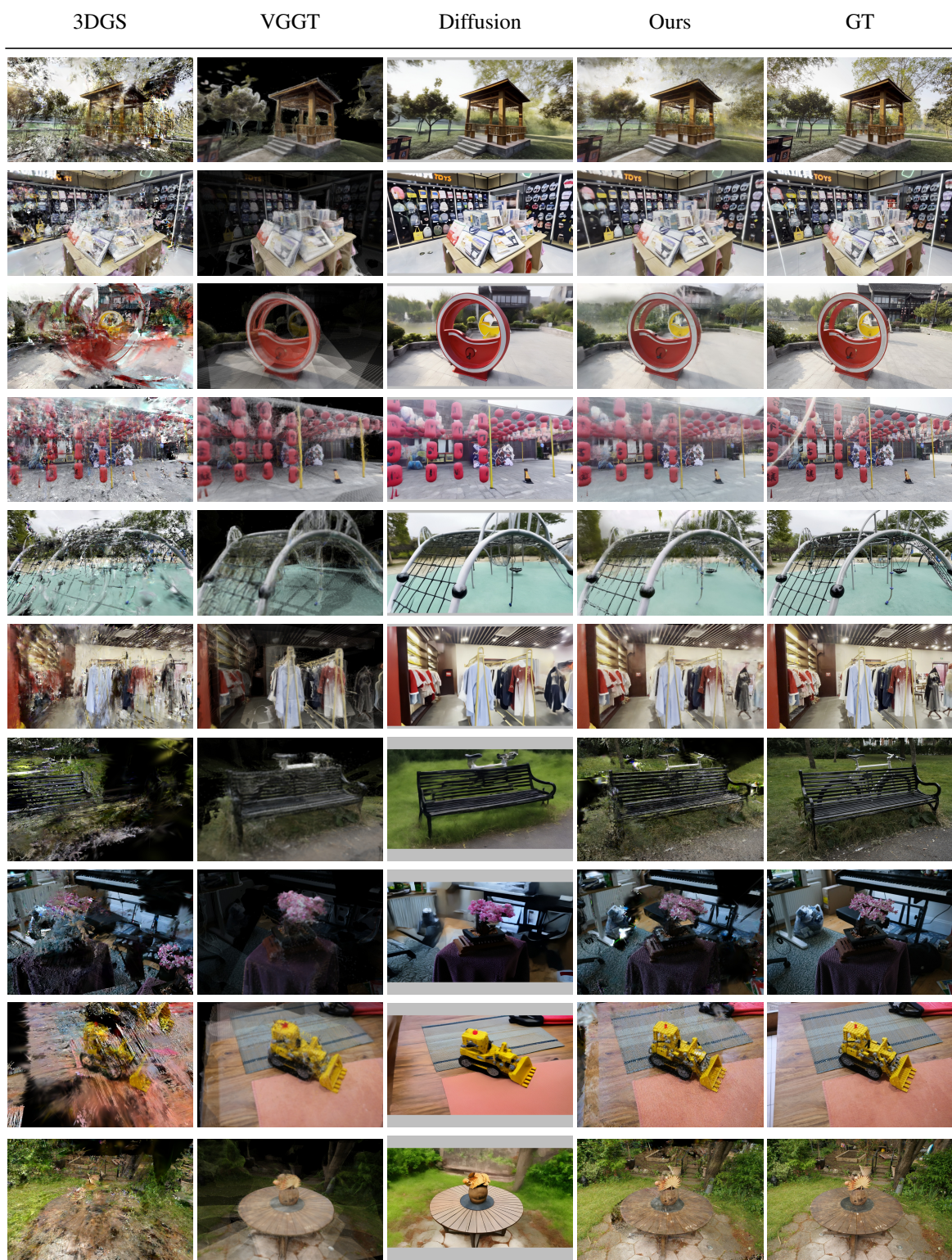


Figure 3.8: Qualitative results of 9 input views.

Chapter 4: Rotated Lights for Consistent and Efficient 2D Gaussians Inverse Rendering

4.1 Introduction

Inverse rendering solves the decomposition of a scene into its geometry, surface material properties, and lighting, usually from a sequence of images of the scene. It has wide applications like scene relighting and digital asset creation.

Usually, inverse rendering is achieved by implementing a differentiable forward rendering model, such as rasterization and physically-based path tracing. Traditionally, differentiable path tracing is considered computationally infeasible until adjoint and path reusing approaches [85, 111] and neural radiance cache [45] were proposed to significantly lower the cost. However, they still assume known geometry. Recent developments in view synthesis methods have shed much light on inverse rendering research. This is largely because recent view synthesis approaches like NeRF [79], SDF [117] and 3D Gaussian Splatting [57] all decompose the scene into high-quality geometry and radiance representations efficiently. Geometry is essential for inverse rendering, and the radiance field can be useful to model indirect illumination.

In particular, Gaussian splatting approaches [51, 57] have encouraged works such as GS-IR [68], RefGaussians [140], and the state-of-the-art IRGS [39]. These approaches are much

faster than previous methods, due to the fast rasterization of Gaussian splatting. In particular, IRGS proposes a two-stage pipeline by first reconstructing the scene geometry and radiance with 2DGS, and then recovering the surface material and lighting using differentiable physically-based rendering. To better handle global illumination, they propose to ray trace the 2D Gaussian primitives and use the integrated radiance as indirect illumination.

While IRGS achieve fast and state-of-the-art quality inverse rendering, the recovered albedo often still contains baked shadows and inconsistent shifts in colors. This is largely due to the ambiguity between light and surface color. To this end, we propose to capture the object under several rotated environment lighting. In practice, this can be easily achieved by rotating the object itself and taking a set of photos at each rotation. We demonstrate that with this simple addition, a significant portion of shadows and artifacts in the estimated albedo can be removed.

In order to handle multiple radiance fields from rotated lights, we use MLPs as scene radiance caches for indirect illumination queries. While directly ray-tracing incident rays on the 2D Gaussians, as proposed in IRGS, is an intuitive approach, we find that it is geometrically inconsistent and may harm incident light integration. To alleviate this, we propose to make use of an extracted mesh as a proxy geometry for accurate and efficient incident queries.

To further improve global illumination handling, we incorporate a residual constraint on the radiance cache, inspired by Inverse Neural Radiosity [45]. The constraint enforces the consistency between cached and rendered radiance and effectively improves albedo in regions covered in shadow where indirect lights dominate. To enable residual evaluation at arbitrary scene locations, we utilize the mesh to randomly sample on the scene surface. Our experiments show that baked shadows in the estimated albedo are further reduced with the residual constraint.

In summary, our contributions are:

1. We propose *RotLight*, a simple but effective capture setup to improve albedo estimation.
2. We analyze the drawbacks of using 2DGS as a scene representation and propose mitigation with a proxy mesh.
3. We incorporate the residual constraint into 2DGS-based inverse rendering to better handle global illumination effects.

4.2 Related Works

4.2.1 Novel View Synthesis

In recent years, neural radiance fields (NeRFs) [79] have gained popularity in novel view synthesis. It predicts view-dependent colors and volume densities for each 3D location along the ray with multi-layer perceptrons (MLPs) and applies volume rendering [55] to render images from novel views. However, NeRF-based methods suffer from slow training and rendering speeds due to the huge amount of MLP queries for volumetric integration, despite a good amount of effort [26, 34, 37, 70, 82, 93, 94, 105, 142, 145] to improve NeRF rendering efficiency. As a result, it remains challenging for the NeRF-based method to achieve real-time high-quality rendering.

Another method to represent scene geometry and radiance is using signed distance fields (SDFs) [53, 117, 141]. These methods are capable of representing accurate and consistent geometry thanks to the regularization derived from their mathematical definitions of SDFs. However, they require iteratively stepping along each ray and evaluating the SDF value to determine if a surface is hit as determined by a radius threshold. Often, many small steps per pixel are required ensure accuracy, making it computationally expensive. Similarly, UniSURF [86] uses expensive

root finding to extract surfaces.

Finally, an increasingly popular branch of works utilize the point-based rendering model [2, 59, 60, 87, 96, 124, 132, 144], which developed combinations of differentiable point-based rendering potentially and learned neural networks. More recently, 3D Gaussian Splatting [57] has attracted much attention. It represents a scene using anisotropic 3D Gaussian primitives, and achieves a new state-of-the-art in both rendering efficiency and quality with its efficient tile-based rasterization method. However, the rasterizer makes approximations that result in multi-view inconsistencies in its renderings, often referred to as *popping artifacts*. StopThePop [91] proposes a more accurate sorting algorithm to mitigate the issue, and 3D Gaussian Ray Tracing [80] implements a ray tracing-based renderer of 3D Gaussian primitives that achieves multi-view consistent renderings at the cost of slower speed. Nevertheless, their abilities to represent accurate geometry are still limited by the lack of a good depth definition for 3D Gaussian primitives. To this end, SuGaR [42] regularizes Gaussians to align with the surface and a method to extract meshes from the Gaussians using Poisson reconstruction. 2D Gaussian Splatting [51] further replaces 3D Gaussian volumes with 2D Gaussian disks, resulting in scene reconstruction with consistent depths, from which high-quality meshes can be extracted with truncated signed distance fields [30].

In our work, we leverage 2D Gaussians for its accuracy in geometry representation, which is crucial for inverse rendering. In addition, we make use of the mesh extraction technique to obtain a less accurate but more consistent model for better incident light tracing.

4.2.2 Inverse Rendering

Inverse rendering can be achieved by implementing a differentiable rendering pipeline and running gradient descent on the material parameters. However, as physically-based rendering pipelines often rely on path tracing, they can have prohibitive computational requirements for complicated problems. Adjoint approaches [85, 111] reduce the cost by propagating gradients and reusing paths. Inverse Neural Radiosity (InvNeRad) [45] proposes to use a neural network as a radiance cache to record incident lights everywhere, with a residual constraint to reduce bias. These methods are accurate, but require known geometry and light.

Recent developments of inverse rendering are closely related and inspired by the advancements in novel view synthesis, as it can be seen as a downstream task that relies on scene geometry being available. In addition, recent view synthesis methods often include certain forms of radiance fields that are convenient to use as radiance caches. For example, NeRF inspired NeRV [104], NeRD [17], Ref-NeRF [110], and many more [5, 18, 54, 73, 136, 139].

Similarly, signed distance fields have been used for modeling geometry in NeILF++ [152], InvRender [153], and WildLight [27].

Gaussian splatting is no exception in being integrated into inverse rendering pipelines [39, 43, 68, 102, 140]. It is particularly preferred because of its fast convergence and rendering speed compared to NeRF and SDF-based methods. The point-based nature of Gaussian splatting also makes it straightforward to attach material properties as additional data to every Gaussian primitive, resulting in natural and fast rendering of material maps. However, a major drawback of using Gaussians as a radiance field is that the splatting rasterization is image-centric and does not provide an easy method to trace incident radiance of arbitrary rays in space. It prevents prior

works from properly handling global illumination effects and limits their performance.

To alleviate this weakness, IRGS [39] proposes to incorporate global illumination by tracing 2D Gaussian primitives as incident light samples, effectively using them as a radiance cache. However, their work still suffers from unwanted effects like baked shadows and color shifts, due to the ambiguous nature of the problem.

Our work takes a step in resolving the ambiguity by capturing objects with multiple rotations, creating observations of the object under different but coherent lighting. This scheme introduces little additional effort, but has been shown to be effective in improving the consistency of the estimated albedo. We also propose a proxy mesh and include a residual constraint to further improve the accuracy of material estimation.

4.3 Background

Our inverse rendering pipeline consists of two stages: *scene reconstruction* that uses novel view synthesis approaches to model the scene with a geometry field and a radiance field, and *material decomposition* that estimates material and environment light given the model from the first stage.

In this section, we cover the related background of both stages.

4.3.1 3D Gaussian Splatting

Given a set of images $\{I_i\}$ and their associated camera parameters $\{P_i\}$, Kerbl et al. [57] propose to model the scene with 3D Gaussian primitives $\mathcal{G} = \{G_i\}$. Each Gaussian is parameterized by its 3D location $\mu_i \in \mathbb{R}^3$, rotation $q_i \in \mathbb{R}^4$, and scale $s_i \in \mathbb{R}^3$. For view-dependent

color rendering, it is also associated with an opacity value σ_i and a set of spherical harmonics (SH) parameters. To render the scene from a camera, all 3D Gaussians are first projected onto the image plane using the camera view matrix W and an affine transformation J that approximates the camera projection. A 3D Gaussian's covariance matrix Σ and the projected Σ' are (dropping index for clarity):

$$\Sigma = R S S^T R^T \quad (4.1)$$

$$\Sigma' = J W \Sigma W^T J^T \quad (4.2)$$

where S and R are the scale and rotation matrices derived from s and q , respectively. The projected Gaussian on the image plane is obtained by skipping the third row and column of Σ' [157].

3DGS then renders each pixel $\mathbf{x}$ by alpha blending all projected Gaussians evaluated at that pixel:

$$\mathbf{c} = \sum_{i=1}^N \left(c_i \sigma_i G_i(\mathbf{x}) \prod_{j=1}^{i-1} (1 - \sigma_j G_j(\mathbf{x})) \right), \quad (4.3)$$

$$G_i(\mathbf{x}) = \exp \left(-\frac{1}{2} (\mathbf{x} - \mathbf{x}_i)^T \Sigma'^{-1} (\mathbf{x} - \mathbf{x}_i) \right) \quad (4.4)$$

where the Gaussian are sorted by the distances between their centers and the camera. $\mathbf{x}_i$ is the projected pixel location of the Gaussian center, and c_i is the view-dependent color of the Gaussian derived from its SH parameters evaluated at the direction from its center to the camera.

For efficiency, 3DGS only considers Gaussians whose computed alpha $\sigma_i G_i(\mathbf{x})$ at the pixel is above a threshold $\alpha_{\min}$. It divides an image into tiles and uses a preprocessing step to find a per-tile list of Gaussians that would have above-threshold alphas in the tile. All pixels in a tile traverse a shared list of Gaussians in order.

4.3.2 2D Gaussian Splatting

Although 3D Gaussians provide good image reconstruction quality, the method imposes challenges to surface geometry reconstruction due to the ambiguity of depth. To this end, Huang et al. [51] propose to replace the primitives with 2D Gaussians, which are similar to 3D Gaussians, except that their scale in the local z direction is zero. Formally, a 2D Gaussian centered at μ lies on a plane defined by two orthogonal unit vectors $\mathbf{t}_u$ and $\mathbf{t}_v$. Its normal is then $\mathbf{n} = \mathbf{t}_u \times \mathbf{t}_v$. These three vectors form a rotation matrix $R = [\mathbf{t}_u, \mathbf{t}_v, \mathbf{n}]$ that characterizes the orientation of the primitive. Similarly to 3DGS, the variance of the Gaussian along these directions is defined by a scale vector $s = (s_u, s_v, 0)$. Then, a point lying on the plane can be parameterized in the Gaussian's local space:

$$P(u, v) = \mu + s_u \mathbf{t}_u u + s_v \mathbf{t}_v v. \quad (4.5)$$

The value of the Gaussian at the point is the standard Gaussian:

$$G(u, v) = \exp\left(-\frac{u^2 + v^2}{2}\right). \quad (4.6)$$

To rasterize a 2D Gaussian, they follow the formulation of [158] to approximate the projection between Gaussian space and screen space points, and further derive an efficient method to solve for (u, v) given image plane $\mathbf{x}$. To compose the color of a pixel from multiple Gaussians, 2DGS follows the same alpha blending formulation as Equation 4.3. Note that the depth and normal of a 2D Gaussian intersecting a ray is uniquely defined, and the depth and normal of the pixel can be composited in the same way.

4.3.3 Gaussian Splatting in Inverse Rendering

With the scene geometry and radiance recovered from the reconstruction methods, inverse rendering poses a further challenge to decompose the radiance into surface materials and environment lighting. Similarly to IRGS [39], we use a physically based rendering model. To render a pixel, its camera ray $\mathbf{r}(t)$ towards direction $\mathbf{d}$ is defined by the camera parameters, and a surface point $\mathbf{x}$ can be determined by using the depth map value d from splatting, i.e. $\mathbf{x} = \mathbf{r}(d)$. The color of the pixel is the outgoing radiance at $\mathbf{x}$ towards $\omega_o = -\mathbf{d}$, defined as:

$$L_o(\mathbf{x}, \omega_o) = \int_{\Omega} f(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i) (\omega_i \cdot \mathbf{n}) d\omega_i, \quad (4.7)$$

where ω_i is the incident direction over the hemisphere Ω defined by the surface normal $\mathbf{n}$ at point $\mathbf{x}$, L_i is the incident radiance function, and f is the bidirectional reflectance distribution function (BRDF).

We follow prior works and use a simplified Disney BRDF model that has two parameters: albedo $\mathbf{a} \in \mathbb{R}^3$ and roughness $r \in \mathbb{R}$. The parameters are attached to every Gaussian primitive and are rendered with the same method as color and depth.

To render an image, we first splat the 2D Gaussians with the 2DGS rasterizer to obtain maps of per-pixel depth, normal, albedo, and roughness. For every pixel, its corresponding 3D point is derived from its depth d by $\mathbf{x} = \mathbf{r}(d)$. Then, a set of incident directions $\{\omega_i\}$ is randomly sampled to compute $L_i(\mathbf{x}, \omega_i)$ and evaluate Equation 4.7 with Monte Carlo integration.

The computation of $L_i(\mathbf{x}, \omega_i)$ can be challenging, as it requires tracing incident rays from $\mathbf{x}$ to determine if it hits scene geometry or the environment lighting. However, 2DGS rasterization is not built for integrating individual rays in arbitrary directions. To this end, IRGS [39] proposes 2D Gaussian Ray Tracing, which follows 3DG-RT [80] to utilize Nvidia Optix for efficient implementation. The ray tracing operator Ray-Trace($\mathbf{x}, \mathbf{d}$) integrates the 2D Gaussian primitives along the ray and composite the same quantities as rasterization, including color $\mathbf{c}_{\text{rt}}$ and alpha α_{rt} . Note that the integrated color corresponds to the *indirect illumination* towards $\mathbf{x}$. Intuitively, ray tracing allows the use of the 2D Gaussians as a cache of the scene radiance. Environment visibility can be derived as $(1 - \alpha)$, and the *direct illumination* can be queried from the optimizable environment map $\mathbf{E}(\omega_i)$. The value of L_i is then the sum of both:

$$L_i(\mathbf{x}, \omega_i) = \mathbf{c}_{\text{rt}} + (1 - \alpha_{\text{rt}})\mathbf{E}(\omega_i). \quad (4.8)$$

4.4 Method

Our proposed inverse rendering approach consists of a simple yet effective capturing setup, *RotLight*, and a two-stage reconstruction pipeline. *RotLight* captures an object under rotated lighting, reducing the ambiguity of the problem (Section 4.4.1). During scene reconstruction, we first train a 2D Gaussian model to represent the scene geometry. Similarly to previous work [39,

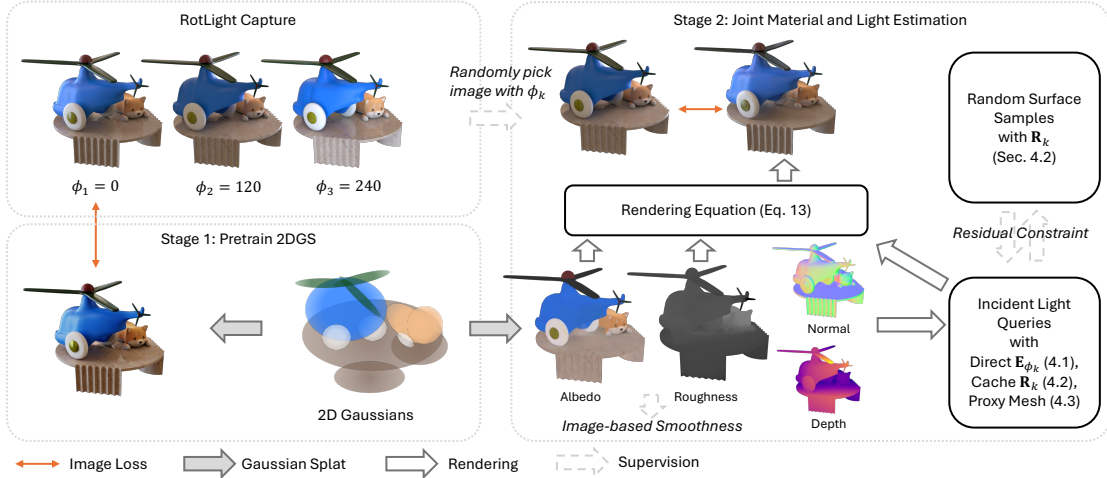


Figure 4.1: Overview of our pipeline. **RotLight**: we capture the same object with $K = 3$ different rotations. **Stage 1**: we pretrain a 2DGS model with a set of images under the same lights (e.g. ϕ_1) to obtain an initial geometry reconstruction of the scene. **Stage 2**: we obtain pixel material, depth and surface normal by splatting Gaussians. The depth and normal are used to query incident lights, with direct illumination from the environment map $\mathbf{E}_{\phi_k}$ (Section 4.4.1), and indirect illumination from a radiance cache (Section 4.4.2). The radiance cache is further supervised with a residual constraint evaluated at random surface points. For accurate incident sampling, we also use a proxy mesh instead of ray tracing the Gaussians (Section 4.4.3).

140], the Gaussian primitives are then augmented with material properties $\mathbf{a}, r$ to represent the surface material. Then, jointly estimate surface material properties and environment lighting. To handle global illumination effects, we use a neural radiance cache represented by several MLPs (Section 4.4.2). Finally, we propose a proxy mesh geometry for consistent and efficient incident radiance queries (Section 4.4.3).

4.4.1 RotLight Capturing

Although prior work [39] achieves plausible material reconstruction by taking global illumination effects into account, they still suffer from the ambiguity of the problem. As every observed color can be explained by an arbitrary combination of albedo color, roughness, direct illumination, and indirect illumination, the optimization process often converges to sub-optimal

results despite regularization efforts. To this end, we propose a simple modification to the capturing setup that drastically reduces the amount of ambiguity and improves albedo recovery. First, the object is captured as usual. Then, we rotate the object by a predefined angle and capture an additional set of images. The process can be repeated as needed, and we use two rotations of 120 degrees in our experiments. We call this approach *RotLight*, referring to the fact that the light on the object is effectively rotated.

In this capture setup, each image in the image set is also tagged with an environment rotation angle ϕ_k . When querying direct illumination of the direction ω_i for a pixel, the angle of its image is taken into account, and $\mathbf{E}_{\phi_k}(\omega_i)$ is obtained by rotating the queried direction along the up axis:

$$\mathbf{E}_{\phi_k}(\omega_i) = \mathbf{E}(R_{\phi_k} w_i), \text{ where} \quad (4.9)$$

$$R_{\phi_k} = \begin{pmatrix} \cos(\phi_k) & 0 & \sin(\phi_k) \\ 0 & 1 & 0 \\ -\sin(\phi_k) & 0 & \cos(\phi_k) \end{pmatrix}. \quad (4.10)$$

4.4.2 Neural Radiance Cache with Residual Constraint

The idea of modeling indirect illuminations with neural networks has been seen in several inverse rendering approaches [39, 45, 152]. In particular, IRGS effectively uses the 2D Gaussians as the cache whose radiance information is trained in the first stage. They query the cache with the Ray-Trace operator that integrates the radiance along the ray and use the integrated color $\mathbf{c}_{\text{tr}}$

for indirect illuminations.

The idea of using a neural radiance cache in forward rendering was first proposed in Neural Radiosity (NeRad) [44]. Their cache is modeled with a spatial feature grid and a multi-layer perceptron (MLP) decoder. It records radiance everywhere in the scene, so that lengthy path-tracing operations can be replaced by a single query to the cache. Importantly, NeRad proposes a *residual constraint* to minimize the following difference, when tracing a ray $\text{Ray-Trace}(\mathbf{x}, \mathbf{d})$:

$$\mathcal{L}_{\text{residual}} = |\mathbf{c}_{\text{tr}}, L_o(\mathbf{y}, -\mathbf{d})|, \quad (4.11)$$

where $\mathbf{c}_{\text{tr}}$ is the cached value from ray tracing, $\mathbf{y}$ is the hit point of the ray, and L_o is defined in Equation 4.7. Intuitively, the constraint requires that the *cached radiance* to match the *rendered radiance*. Without this constraint, other works [39, 152] constrain their radiance cache (SDF and 2D Gaussians) solely with input images and camera rays. As shown in InvNeRad [45], the residual loss improves cache queries from unseen locations and directions and makes the handling of global illumination more accurate.

In this work, we use MLPs to model the radiance cache, instead of re-using the 2D Gaussians like IRGS. Although the residual constraint can be applied to the Gaussians-based cache, we made this choice for the following considerations:

1. With our *RotLight* capture, a cache is required for each unique light angle. It is faster in computation and simpler in implementation to train several MLPs than to optimize several sets of Gaussians.
2. For a given point and direction in space, querying an MLP is more efficient than tracing the Gaussians.

For every light angle ϕ_k in the captured images, we define a radiance cache $\mathbf{R}_k(\mathbf{x}, \mathbf{d})$. The indirect illumination at a point $\mathbf{x}$ in direction ω_i is found by:

$$\mathbf{c}_{\text{tr}} = \mathbf{R}_k(\mathbf{y}, -\omega_i), \quad (4.12)$$

where $\mathbf{y}$ is the hit point of the ray from $\mathbf{x}$ towards ω_i . We discuss how to find $\mathbf{y}$ in Section 4.4.3.

4.4.2.1 Sampling Scene Geometry

To evaluate residual loss $\mathcal{L}_{\text{residual}}$, random surface points and directions need to be sampled from the scene. However, since depths of 2DGS are defined with volumetric integration, individual Gaussian primitives do not provide explicit information about surface locations. Instead, surface information is encoded with depth maps, which has been shown in 2DGS [51] to be capable of supporting high-quality mesh extraction with TSDF [30]. Therefore, we choose to extract a mesh in the same manner. Then, at each optimization step, we uniformly randomly choose N_{residual} triangles from the mesh. For each triangle, we pick a random point and a random view direction over the hemisphere defined by its normal.

While the mesh should theoretically be updated periodically during the second stage to reflect geometry changes, we find empirically that the changes in geometry are often very small and do not require the mesh to be regenerated.

4.4.3 Proxy Mesh for Incident Queries

For an arbitrary ray from a scene surface $\mathbf{x}$ in direction $\mathbf{d}$, we need to find its intersection point $\mathbf{y}$ to query the MLP radiance cache in Equation 4.12. We propose to use the extracted mesh discussed previously for incident ray queries. For efficient queries, we implement an Optix-based ray tracing extension for Python, which reports the ray distance, hit triangle index, and barycentric coordinate. Compared to the naive alternative of re-using IRGS’s 2D Gaussian ray tracer, the mesh is not only more view consistent but also faster for incident queries, since they are opaque and do not require integration. Our approach, as demonstrated in Figure 4.9, results in accurate and smooth ambient occlusion renderings.

One catch of this approach is the primary hit points of camera rays, where the pixel colors are computed. For a camera ray, its hit point $\mathbf{x}$ from splatting may not be exactly on the mesh, causing issues in subsequent incident ray tracing. We use a simple workaround to also trace the ray with the mesh and obtain an alternative hit point $\mathbf{x}_m$. Then, Equation 4.7 is modified to find incident lights for $\mathbf{x}_m$:

$$L_o(\mathbf{x}, \omega_o) = \int_{\Omega} f(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}_m, \omega_i) (\omega_i \cdot \mathbf{n}) d\omega_i, \quad (4.13)$$

4.4.3.1 Comparing 2D Gaussian Ray Tracing

Since we use 2D Gaussians to model the scene, an obvious alternative is to follow IRGS: ray-trace the Gaussian primitives and obtain per-pixel depth. However, this method suffers from inconsistencies when the Gaussians are rendered from different views. This issue of Gaussian splatting has also been discussed in prior work [91]. In addition, as Gaussians form thick clouds

on model surfaces, the origin of an incident ray must be shifted along its direction by a significant amount. This amount can be tricky to handle and will affect the accuracy of incident queries. We demonstrate this problem by rendering ambient occlusion images that show the amount of direct light that is deemed visible at every point during rendering. In Figure 4.9, it can be seen that ray-tracing 2D Gaussians results in visible artifacts in ambient occlusion.

4.4.4 Optimization Objective

In summary, our optimization objective is to minimize the following losses.

- $\mathcal{L}_{data} = |L_o(\mathbf{x}, \omega_o) - \mathbf{c}_{gt}|$, the difference between the rendered color and the ground truth color.
- $\mathcal{L}_{cache} = |\mathbf{R}_k(\mathbf{x}, \omega_o) - \mathbf{c}_{gt}|$, which optimizes the radiance cache MLPs for camera rays.
- $\mathcal{L}_{residual}$, the residual loss discussed in Section 4.4.2.
- $\mathcal{L}_{reg}$, regularization loss terms that include mask entropy, albedo smoothness, light smoothness, roughness smoothness, and light whiteness. All of these regularization terms follow IRGS [39].

So, the final loss is the sum:

$$\mathcal{L} = \mathcal{L}_{data} + \mathcal{L}_{cache} + \lambda_{residual}\mathcal{L}_{residual} + \mathcal{L}_{reg} \quad (4.14)$$

In all experiments, we use $\lambda_{residual} = 10$.

4.5 Experiments

4.5.1 SynRotLight Dataset

To perform fair quantitative evaluations of our proposed method, we create and release a dataset *SynRotLight*, examples shown in Figure 4.3. It consists of four object-centric scenes with varying complexity, each containing 100 training views and 50 test views. The scenes were created and rendered with Blender. The model of the *hotdog* scene is modified from the NeRF dataset, while the rest are adopted from online royalty free and CC0 resources. Following NeRF [79], the images have resolution 800×800 .

4.5.2 Implementation

We implement our pipeline with PyTorch based on 3DGS, RefGaussians, and IRGS. In addition, to achieve fast ray tracing on meshes, we implement a PyTorch extension with Nvidia Optix. For a fair comparison, our method shares the same first stage 2DGS model with IRGS, and our second stage is trained for the same number of iterations. Our radiance cache MLPs are pre-trained for 20,000 steps regardless of the number of lights.

4.5.3 Material Recovery

We evaluated and compared the performance of the material reconstruction of our proposed method in Table 4.1. We also show qualitative comparisons of albedo in Figure 4.4, and roughness in Figure 4.5. In addition, the results of the environment light estimation are shown in Figure 4.6.

Method	Hotdog			Table			Toy			Counter		
	Albedo PSNR	Roughness SSIM	MSE	Albedo PSNR	Roughness SSIM	MSE	Albedo PSNR	Roughness SSIM	MSE	Albedo PSNR	Roughness SSIM	MSE
IRGS	25.157	0.931	0.017	25.667	0.952	0.009	30.524	0.930	0.008	21.971	0.943	0.053
Ours	29.227	0.952	0.015	28.621	0.962	0.008	34.073	0.937	0.007	27.098	0.965	0.019
w/o residual	29.037	0.952	0.015	28.529	0.962	0.008	33.368	0.936	0.008	27.071	0.966	0.018
w/o proxy mesh	27.226	0.946	0.014	28.329	0.961	0.007	33.083	0.936	0.007	26.560	0.965	0.015
w/o <i>RotLight</i>	25.357	0.930	0.018	26.289	0.954	0.009	31.774	0.932	0.008	22.907	0.944	0.048

Table 4.1: Quantitative comparison of material estimation. All experiments are conducted in linear color space. **Top:** our method outperforms prior works by a large margin thanks to the capturing setup. **Bottom:** ablation studies show that our method gains the most improvement from *RotLight*.

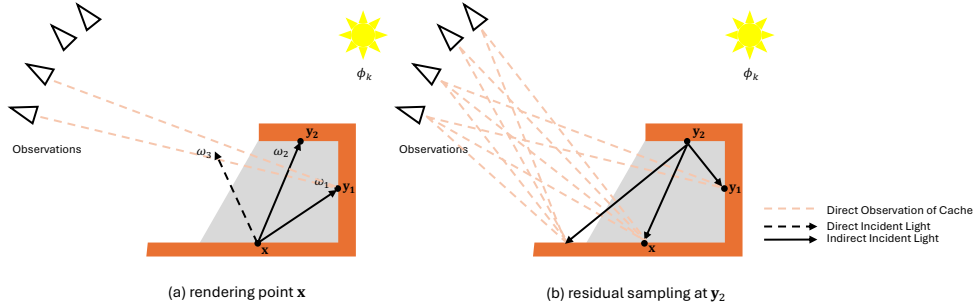


Figure 4.2: Analysis of residual constraint. In the above example, consider the albedo estimation of point x . **(a)** when querying incident radiance $L_i(x, \omega_i)$, we show three possibilities of the sampled directions. ω_3 does not hit scene geometry and queries the environment. ω_1 and ω_2 hits the scene at y_1, y_2 respectively, and will query the radiance cache $\mathbf{R}_k$. Note that y_1 can be observed from several cameras, so the cached radiance $\mathbf{R}_k(y_1, -\omega_1)$ may be relatively accurate, in contrast to $\mathbf{R}_k(y_2, -\omega_2)$. **(b)** in this case, the residual constraint can be useful to correct $\mathbf{R}_k(y_2, -\omega_2)$ by rendering $L_o(y_2, -\omega_2)$ with Equation 4.13, which in turns queries incident radiance from more directly observable regions (like x and y_1) towards y_2 . As a result, the cached value $\mathbf{R}_k(y_2, -\omega_2)$ will be more accurate, which benefits the material estimation of x .

4.5.4 Ablation Study

We perform an ablation study of our proposed components in Table 4.1. Qualitative examples are shown in Figure 4.7 and Figure 4.8. The use of the *RotLight* capture setup drastically improves the recovery of albedo. With the proposed proxy mesh geometry for incident queries, incident lights are more accurately integrated, which also benefits material estimation. The residual constraint, while not significant in metrics, is shown to be effective in improving regions

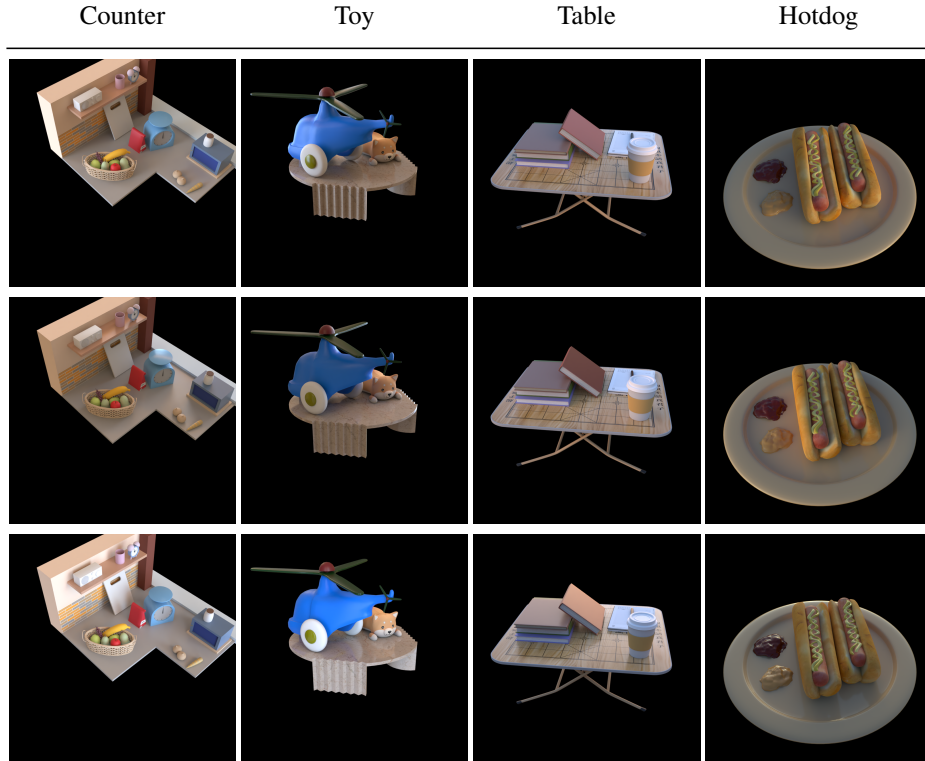


Figure 4.3: Examples from our *RotLight* dataset. The three rows are light rotations at 0, 120 and 240 degrees respectively.

covered in shadows. We provide a more detailed analysis below.

4.5.5 Residual Constraint

As depicted in Figure 4.2, the residual constraint is most effective in regions where indirect illumination comes from points and directions that are not directly visible in any input views. In practice, this often means regions that are covered or surfaces that are close to each other, such as the examples in Figure 4.7.

4.5.6 Mesh Proxy Geometry

We use a mesh as a proxy geometry for incident queries for its better accuracy compared to integrating 2D Gaussians. To demonstrate its effects, we show ambient occlusion maps in Figure 4.9. When ray tracing 2D Gaussians, the origins must be moved along the ray by an appropriate amount, which can be challenging to determine and adapt. A small value may lead to ray intersecting surfaces at the origin, while a large value can move the origins so far that some intersections are ignored. In practice, IRGS [39] uses a large factor of 0.05 which, as we show, leads to noisy and erroneous incident visibility results.

4.5.7 Relighting

We follow the algorithm in IRGS [39] to render the recovered models with different environment maps and show the results in Figure 4.10. Overall, our method produces colors that are closer to ground truth.

4.5.8 Real World Examples

We capture two real-world videos, *box* and *desk*, with examples shown in Figure 4.11. Each scene has three videos with the object rotated by 0, 90 and 180 degrees. For every scene, object masks are created by rotoscoping in Adobe After Effects, and the masked images from all videos of the scene are registered with COLMAP [99, 100]. We show the recovered albedo maps from IRGS and our method in Figure 4.12. Our method makes use of multiple light sources and estimates albedo with less baked shadows.

4.6 Limitations

Our method is not free from limitations. First, its performance is highly dependent on the quality of the mesh extracted from the initial 2D Gaussian model. This is particularly challenging near the object intersections like the bottom of the hotdogs, where inaccurate mesh prevents decomposition of shadows from albedo. We further demonstrate this in Figure 4.13, where we experiment with known mesh geometry and it helps reduce baked artifacts. Second, the details of the reconstructed albedo in the shadowed regions can be blurred, like under the books of *table*. The reason is that the material properties are encoded in the Gaussians, and these dark regions tend to be under-represented with low Gaussians density. It may be possible to mitigate this problem with alternative material encoding methods such as MLPs.

4.7 Conclusion

We propose a novel inverse rendering method that utilizes rotated captures of an object. Using 2D Gaussians as scene representation, we incorporate a proxy mesh and residual constraints on the scene radiance cache to improve accuracy of global illumination effect modeling, resulting in state-of-the-art performance in albedo estimation.

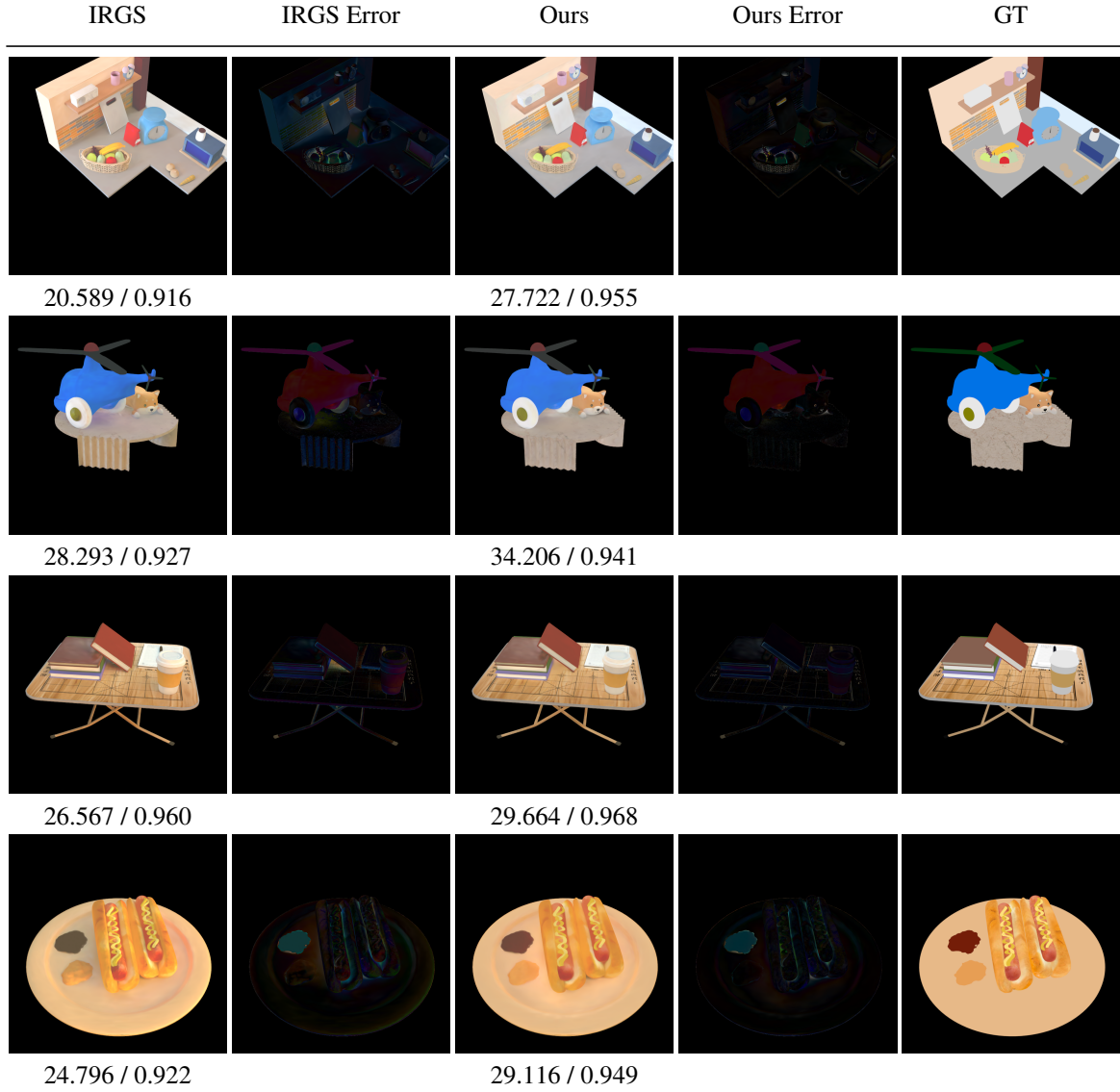


Figure 4.4: Qualitative results of albedo estimation. Thanks to our *RotLight* setup, our method is able to better factor out environment illumination from albedo, resulting in more consistent and accurate estimations. Numbers below images are PSNR and SSIM metrics.

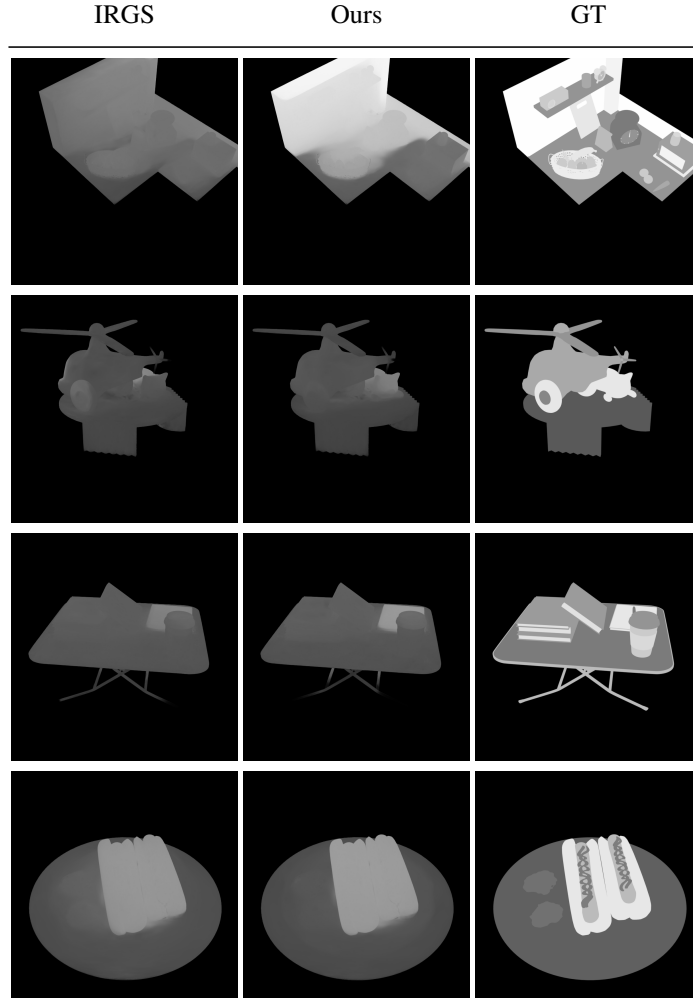


Figure 4.5: Qualitative results of roughness estimation. Our method is better in *counter* and similar to IRGS in other scenes. However, roughness estimation still remains challenging overall.

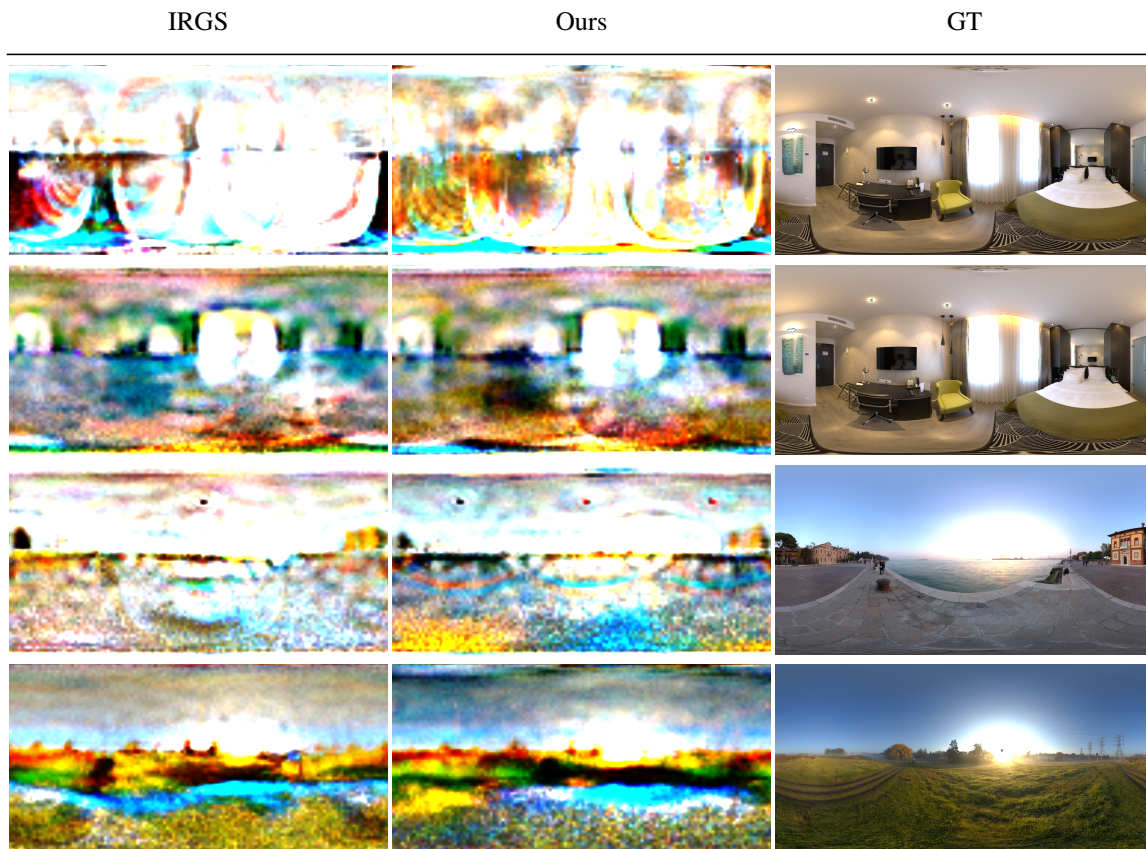


Figure 4.6: Qualitative results of environment light estimation. Top to bottom: *counter*, *toy*, *table*, *hotdog*. Note that the problem is still extremely ill-posed without direct observations from very specular surfaces. Our method give overall closer results, such as the color of the sky.

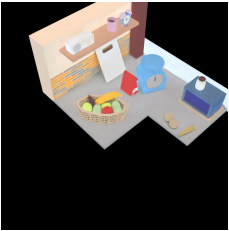
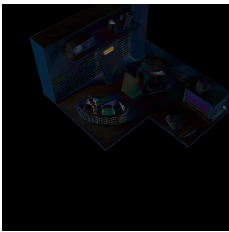
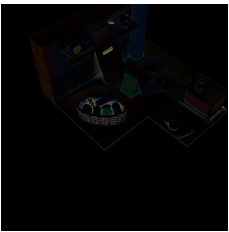
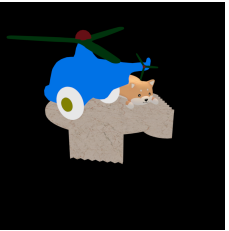
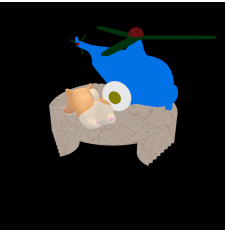
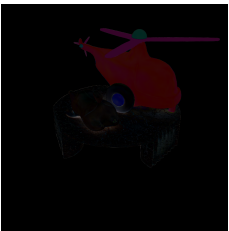
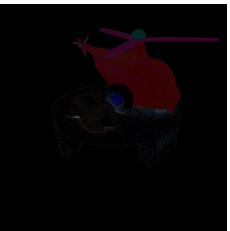
w/o RotLight	w/o Proxy Mesh	w/o Residual	Ours	GT
				
				
22.204 / 0.918	27.279 / 0.955	27.481 / 0.955	27.722 / 0.955	
				
				
29.880 / 0.929	32.982 / 0.940	33.347 / 0.941	34.206 / 0.941	
				
				
31.984 / 0.939	32.943 / 0.942	32.620 / 0.942	33.540 / 0.942	

Figure 4.7: Ablation study. Even rows show error maps. Additional examples shown in Figure 4.8. Without residual constraint, the estimated albedo has more baked shadow where incident illumination presents from locations not directly visible from the camera. *counter*: under the chopping board, between the red bag and blue scale.

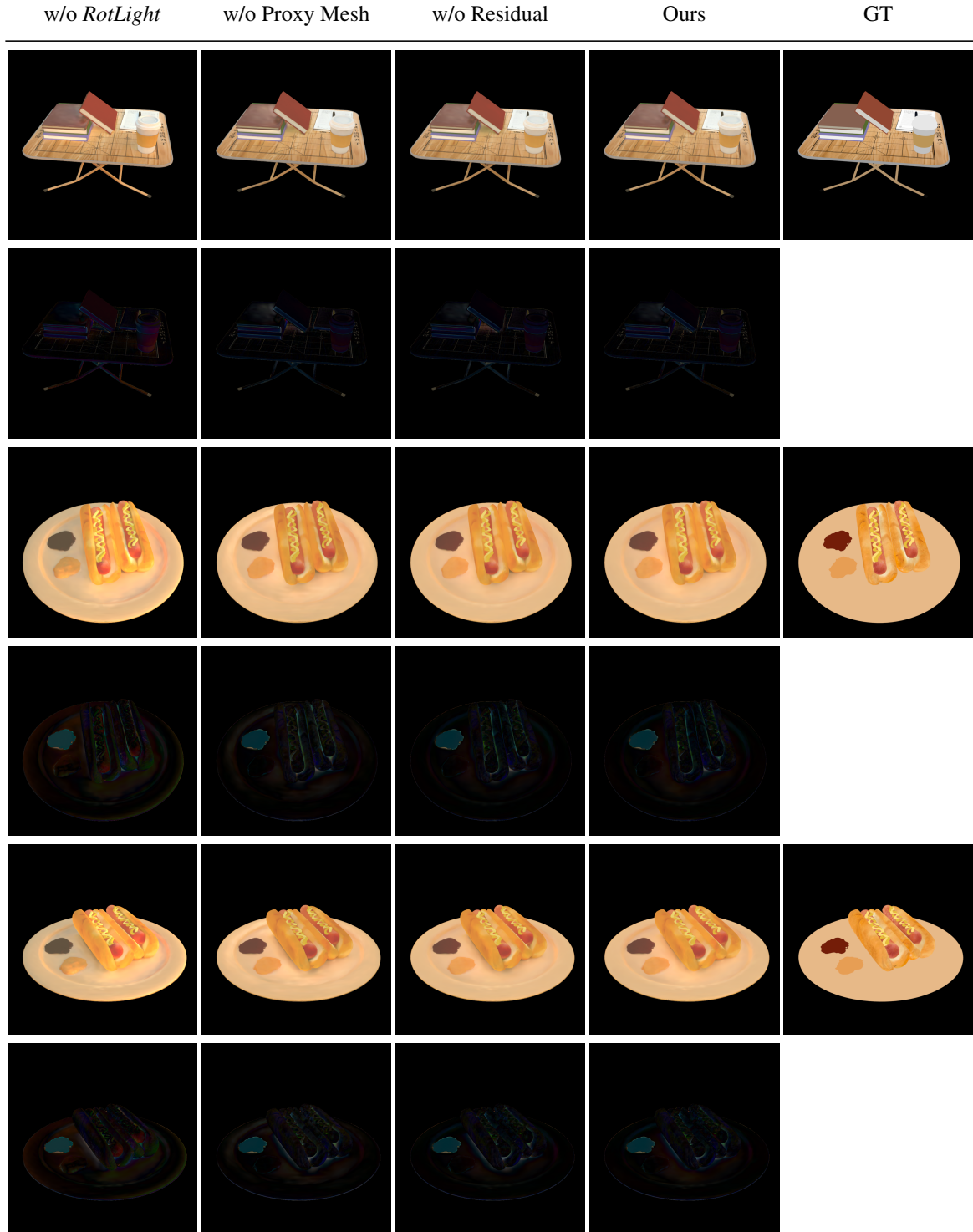


Figure 4.8: Ablation study of more scenes. Baked shadows in albedo can be found without residual constraint. *toy*: under the helicopter and dog. *table*: between the books. *hotdog*: under the front end of the hotdog.

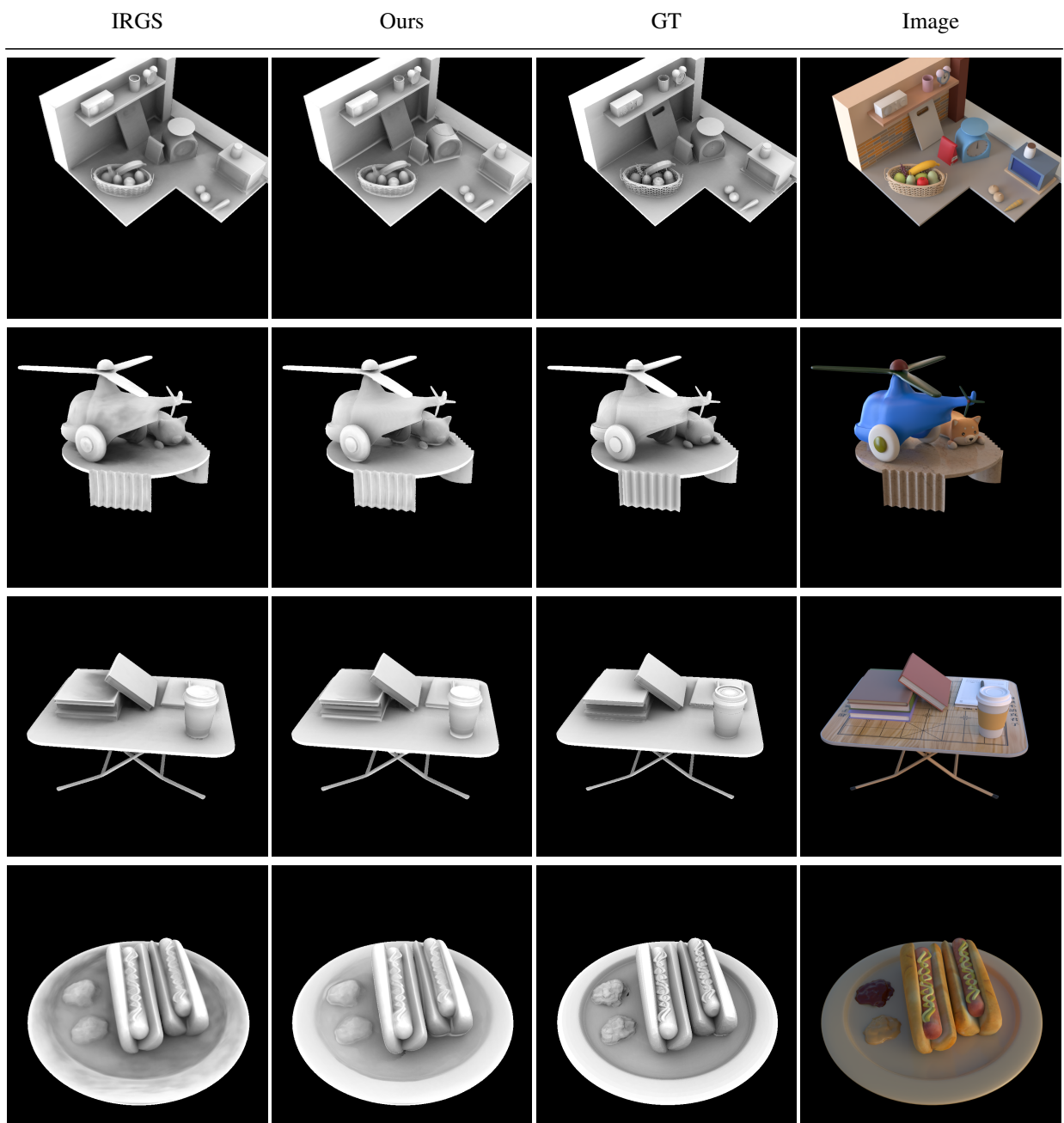


Figure 4.9: Ambient occlusion rendering. With the proxy mesh, our ambient occlusion rendering is smoother and more accurate.

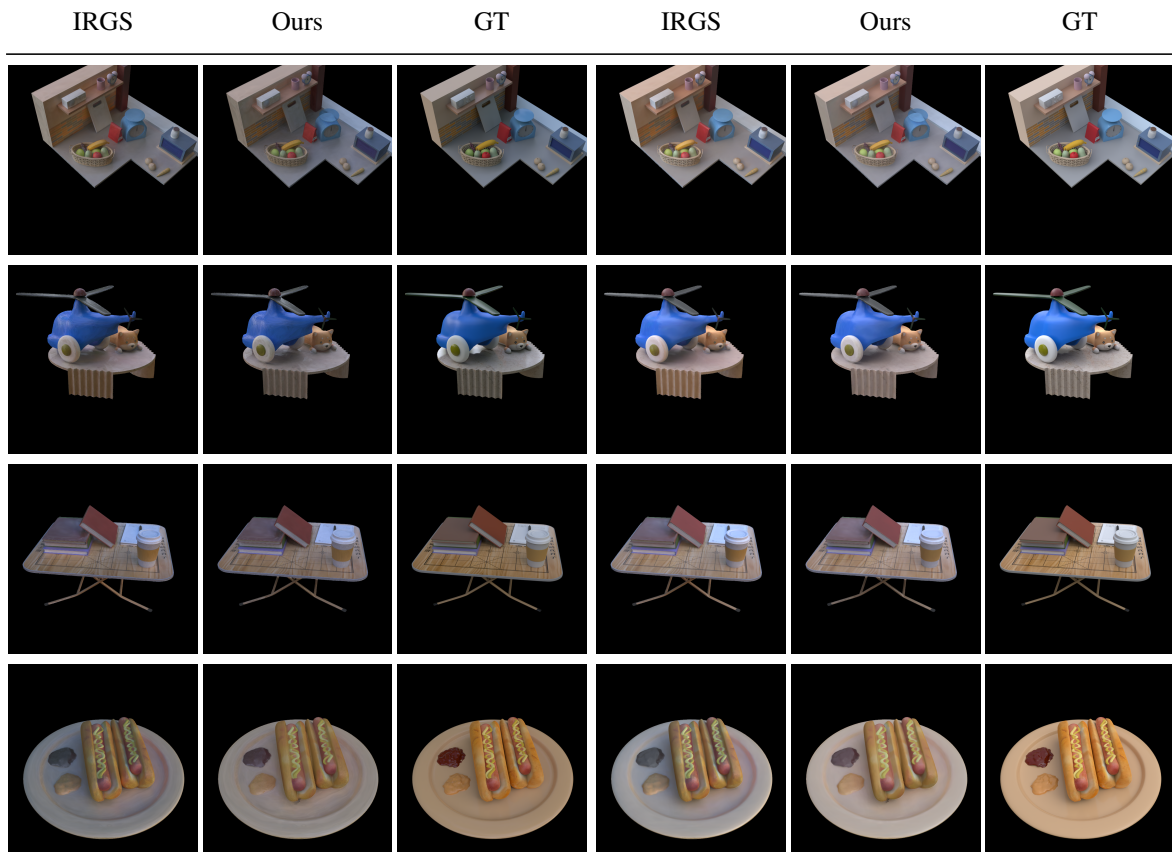


Figure 4.10: Relighting comparison. We demonstrate relighting results using our recovered scene parameters and compare them with IRGS. Our method produces colors that are closer to ground truth.

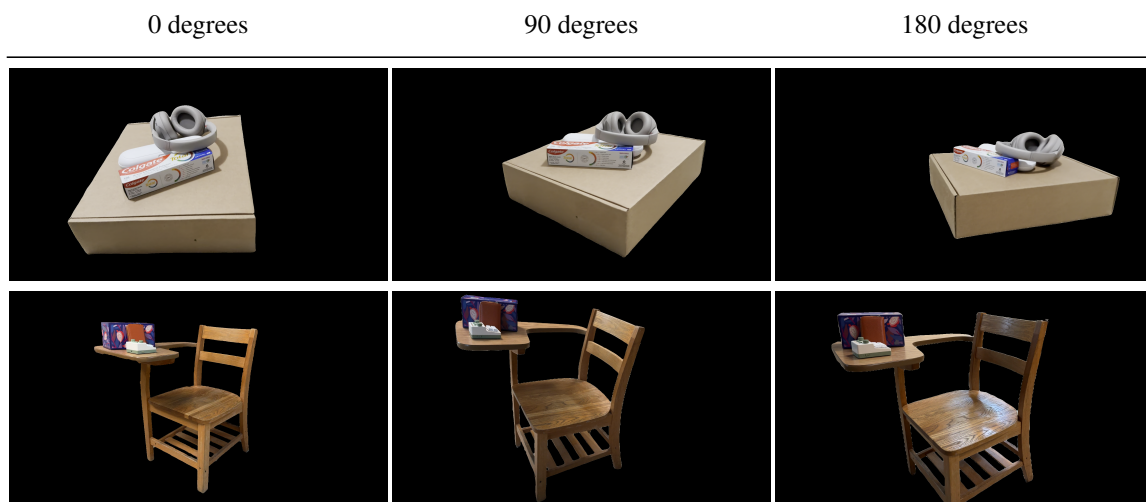


Figure 4.11: Examples of the captured real-world videos, *box* and *desk*.

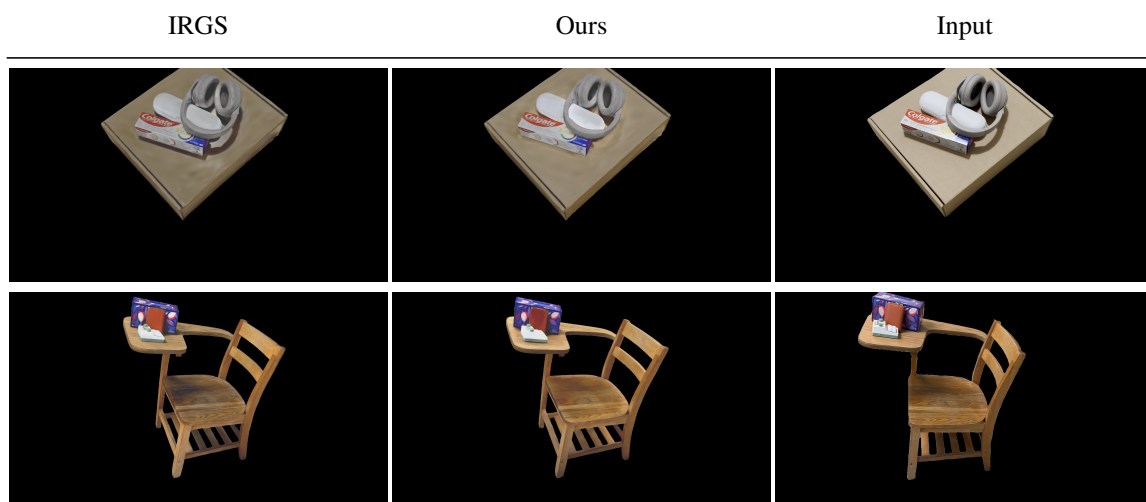


Figure 4.12: Albedo recovery of real-world data. Thanks to RotLight capture, our recovered albedo maps have less baked shadows.

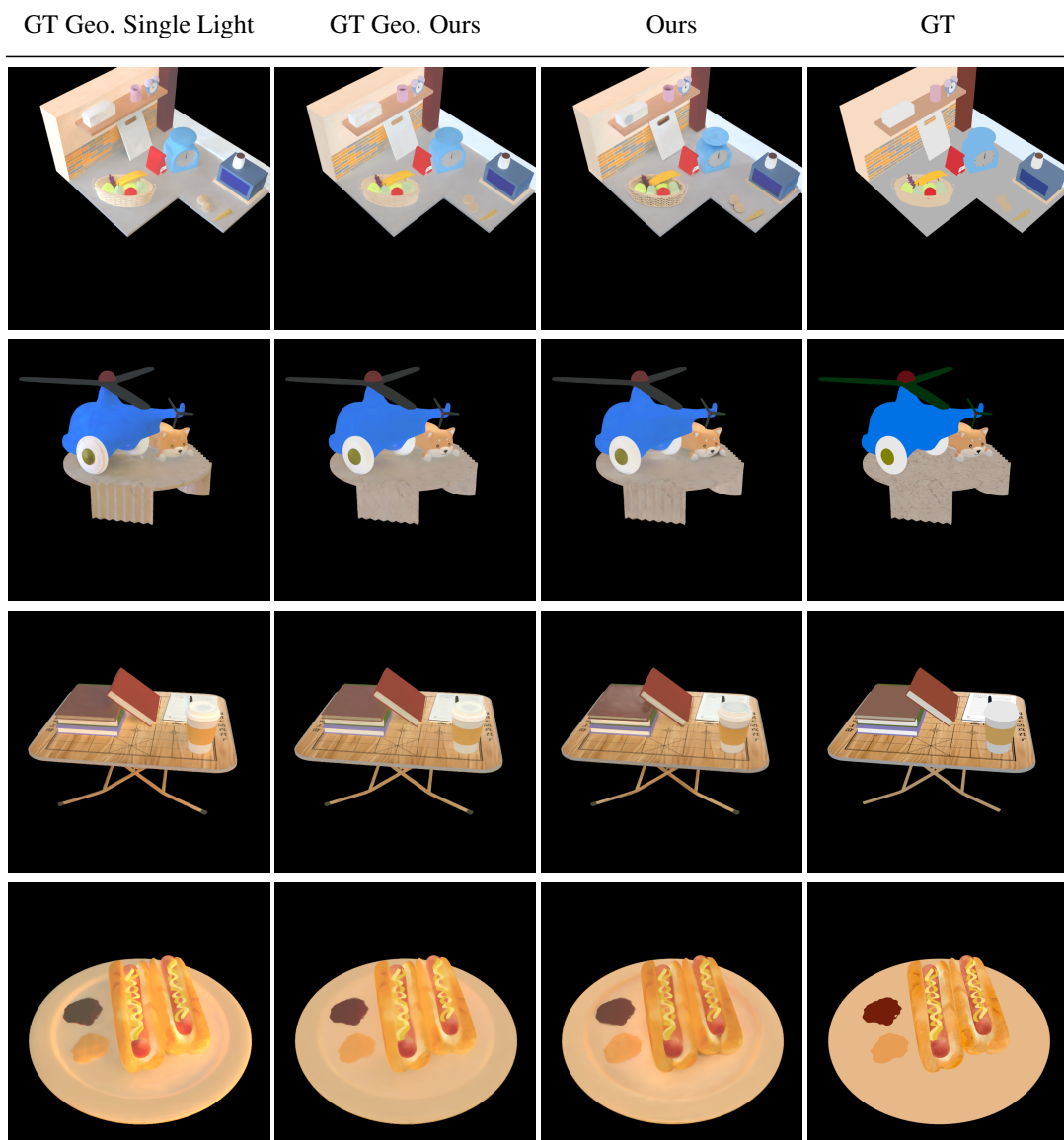


Figure 4.13: Known geometry experiments. The first two columns are experiments run with known (GT) geometry, with single light and RotLight setups respectively. **Column 1, 2:** we show that the RotLight capture setup and other proposed components is effective with known geometry. **Column 2, 3:** the performance of our method is limited by inaccuracies in geometry, such as under the onions in *counter*, under the dog in *toy*, the top of the book pile in *table*, and under the hotdogs in *hotdog*.

Chapter 6: Conclusions

The focus of this thesis is to improve the quality and robustness of novel view synthesis and inverse rendering for challenging inputs, thus allowing more flexibility in data acquisition and preprocessing. In particular, we study three aspects of reducing input constraints.

1. Foreground distractions in video scene reconstruction (Chapter 2). We propose decomposing the foreground objects with their associated effects into 2D image layers and a static 3D background layer. The foreground layers are capable of capturing complicated motions and shapes, which helps to create a clean background layer without artifacts from the shadows and reflections.
2. Novel view synthesis from very sparse inputs (Chapter 3). We leverage a large geometry foundation model and conditional video generative priors to create a pipeline that creates dense image sets from sparse inputs and supervises 3D scene reconstruction. The geometry foundation model robustly estimates dense point clouds for both conditioning the video prior and effective initialization of the reconstruction. We also address hallucination with image warps and robust sampling.
3. Ambiguity between light and material in inverse rendering (Chapter 4). We propose a simple capturing setup to take multiple sets of images under rotational environment lighting, which effectively reduce the baked shadows in the estimated albedo and improve overall

quality. We also incorporate a proxy mesh for accurate incident light queries and residual constraints. As a result, global illumination is handled more accurately in our pipeline, and less shadows are baked in the albedo.

6.1 Limitations

Our work has made it easier to capture for efficient and high-quality view synthesis. However, there remains room for improvement.

1. Our OmnimatteRF pipeline from Chapter 2 does not handle dynamics in the background. In real world videos, subtle movements like trees shaking from winds can present in the background and may cause problems. A deformable background model may be useful. In addition, our method relies on heuristics like motion cues from optical flow estimation but does not have actual semantic understanding of the objects. This limits it to foreground objects with sufficient motion, and may impose challenges when handling multiple objects.
2. In Chapter 3, we propose methods to improve the robustness of the reconstruction with the presence of inconsistencies from video generations, but do not resolve the cause of inconsistencies itself. Future works may make better use of the initial point cloud and guide the video generation prior by injecting geometric cues. Also, the video diffusion model we use is computationally heavy in both amount of work and VRAM requirements, which may limit the practical use of our approach.
3. For the inverse rendering pipeline in Chapter 4, the quality of our method is limited by the accuracy of geometry reconstruction and the extracted mesh. An ideal approach would give additional supervision signals to the geometry model during joint optimization. Also,

while the proposed RotLight capture is practical for objects, it is challenging to extend it to scene-level reconstruction.

A limitation common to all the works is that while the resulting 3D models achieve real-time rendering, the creation of these models still requires per-scene optimization. They may also require excessive amounts of memory and computation for large scenes. Therefore, further exploration in fast and efficient extensions will be helpful.

6.2 Future Work

To further improve the flexibility of input, it can be favorable to multiple modalities such as text descriptions and sound information. When we look further into the applications of these methods, one goal is to enable interactions in virtual reality. For this purpose, it may be useful to associate semantic understandings and physical properties with objects to better simulate their interactions with the environment and the user.

Recall the imaginary scenario at the beginning, making virtual visits to museums and gathering with friends in virtual spaces. With these advances, virtual spaces can be created by casually capturing a corner on the street. As fewer photos are required to achieve high-quality reconstruction, ordinary people can also showcase their creations without the resources of a museum, and the converted models will have a realistic look under any virtual lighting.

In the near future, it may be possible for us to put on a headset that not only brings us into a virtual world but also quickly digitalizes the physical space around us, so that we can immediately share with others and invite them to have an immersive tour along. It enables enormous possibilities by bringing people together: explorers can share their excitement with people who

cannot make those trips, and families living apart can have a glance of each other's lives. The works in this thesis are a step towards closing the gap between this imagination and reality.

Bibliography

- [1] Marc Alexa, Markus Gross, Mark Pauly, Hanspeter Pfister, Marc Stamminger, and Matthias Zwicker. Point-based computer graphics. In *ACM SIGGRAPH 2004 Course Notes*, pages 7–es. Association for Computing Machinery, 2004.
- [2] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 696–712. Springer, 2020.
- [3] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O’Toole, and Changil Kim. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- [4] Benjamin Attal, Jia-Bin Huang, Michael Zollhöfer, Johannes Kopf, and Changil Kim. Learning neural light fields with ray-space embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.
- [5] Benjamin Attal, Dor Verbin, Ben Mildenhall, Peter Hedman, Jonathan T Barron, Matthew O’Toole, and Pratul P Srinivasan. Flash cache: Reducing bias in radiance cache based inverse rendering. In *European Conference on Computer Vision*, pages 20–36. Springer, 2024.
- [6] Xue Bai, Jue Wang, David Simons, and Guillermo Sapiro. Video snapcut: robust video object cutout using localized classifiers. *ACM Trans. Graph.*, 28(3):70, 2009. doi: 10.1145/1531326.1531376.
- [7] Yanqi Bao, Jing Liao, Jing Huo, and Yang Gao. Distractor-free generalizable 3d gaussian splatting. *arXiv preprint arXiv:2411.17605*, 2024.
- [8] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5855–5864, 2021.
- [9] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022.

- [10] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022.
- [11] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19697–19705, 2023. arXiv:2303.13485.
- [12] Mojtaba Bemana, Karol Myszkowski, Jeppe Revall Frisvad, Hans-Peter Seidel, and Tobias Ritschel. Eikonal fields for refractive novel-view synthesis. In Munkhtsetseg Nandigjav, Niloy J. Mitra, and Aaron Hertzmann, editors, *SIGGRAPH '22: Special Interest Group on Computer Graphics and Interactive Techniques Conference, Vancouver, BC, Canada, August 7 - 11, 2022*, pages 39:1–39:9. ACM, 2022. doi:10.1145/3528233.3530706.
- [13] Siavash A. Bigdeli, Geng Lin, L. Andrea Dunbar, Tiziano Portenier, and Matthias Zwicker. Learning Generative Models Using Denoising Density Estimators. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12, 2023. doi:10.1109/TNNLS.2023.3308191.
- [14] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.
- [15] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22563–22575, 2023.
- [16] Institute Blender, Oct 2022. URL: <https://studio.blender.org/>.
- [17] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12684–12694, 2021.
- [18] Mark Boss, Andreas Engelhardt, Abhishek Kar, Yuanzhen Li, Deqing Sun, Jonathan Barron, Hendrik Lensch, and Varun Jampani. Samurai: Shape and material from unconstrained real-world arbitrary image collections. *NeurIPS*, 2022.
- [19] Gabriel J. Brostow and Irfan A. Essa. Motion based decompositing of video. In *Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece, September 20-25, 1999*, pages 8–13. IEEE Computer Society, 1999. doi:10.1109/ICCV.1999.791190.

- [20] Yohann Cabon, Lucas Stofl, Leonid Antsfeld, Gabriela Csurka, Boris Chidlovskii, Jerome Revaud, and Vincent Leroy. Must3r: Multi-view network for stereo 3d reconstruction. *arXiv preprint arXiv:2503.01661*, 2025.
- [21] Chenjie Cao, Xinlin Ren, and Yanwei Fu. Mvsformer: Multi-view stereo by learning robust image features and temperature-based depth. *arXiv preprint arXiv:2208.02541*, 2022.
- [22] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022.
- [23] Xingyu Chen, Qi Zhang, Xiaoyu Li, Yue Chen, Ying Feng, Xuan Wang, and Jue Wang. Hallucinated neural radiance fields in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12943–12952, 2022.
- [24] Yuedong Chen, Haoifei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European Conference on Computer Vision*, pages 370–386. Springer, 2024.
- [25] Yuedong Chen, Chuanxia Zheng, Haoifei Xu, Bohan Zhuang, Andrea Vedaldi, Tat-Jen Cham, and Jianfei Cai. Mvsplat360: Feed-forward 360 scene synthesis from sparse views. *arXiv preprint arXiv:2411.04924*, 2024.
- [26] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16569–16578, 2023.
- [27] Ziang Cheng, Junxuan Li, and Hongdong Li. Wildlight: In-the-wild inverse rendering with a flashlight. *arXiv preprint arXiv:2303.14190*, 2023.
- [28] Yung-Yu Chuang, Aseem Agarwala, Brian Curless, David Salesin, and Richard Szeliski. Video matting of complex scenes. *ACM Trans. Graph.*, 21(3):243–248, 2002. doi: 10.1145/566654.566572.
- [29] Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. Depth-regularized optimization for 3d gaussian splatting in few-shot images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 811–820, 2024.
- [30] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96*, page 303–312, New York, NY, USA, 1996. Association for Computing Machinery. doi: 10.1145/237170.237269.
- [31] Hiba Dahmani, Moussab Bennehar, Nathan Piasco, Luis Roldao, and Dzmitry Tsishkou. Swag: Splatting in the wild images with appearance-conditioned gaussians. In *European Conference on Computer Vision*, pages 325–340. Springer, 2024.

- [32] Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, et al. Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds. *arXiv preprint arXiv:2403.20309*, 2(3):4, 2024.
- [33] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, Zhangyang Wang, et al. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *Advances in neural information processing systems*, 37:140138–140158, 2024.
- [34] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5501–5510, 2022.
- [35] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2009.
- [36] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.
- [37] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14346–14355, 2021.
- [38] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [39] Chun Gu, Xiaofei Wei, Zixuan Zeng, Yuxuan Yao, and Li Zhang. Irgs: Inter-reflective gaussian splatting with 2d gaussian ray tracing. In *CVPR*, 2025.
- [40] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2495–2504, 2020.
- [41] Zeqi Gu, Wenqi Xian, Noah Snavely, and Abe Davis. Factormatte: Redefining video matting for re-composition tasks, 2022. URL: <https://arxiv.org/abs/2211.02145>, doi:10.48550/ARXIV.2211.02145.
- [42] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the*

- IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2024.
- [43] Yijia Guo, Yuanxi Bai, Liwen Hu, Ziyi Guo, Mianzhi Liu, Yu Cai, Tiejun Huang, and Lei Ma. Prtgs: Precomputed radiance transfer of gaussian splats for real-time high-quality relighting. In *ACMMM*, 2024.
 - [44] Saeed Hadadan, Shuhong Chen, and Matthias Zwicker. Neural radiosity. *ACM Transactions on Graphics*, 40(6):1–11, dec 2021. URL: <https://doi.org/10.1145%2F3478513.3480569>, doi:10.1145/3478513.3480569.
 - [45] Saeed Hadadan, Geng Lin, Jan Novák, Fabrice Rousselle, and Matthias Zwicker. Inverse global illumination using a neural radiometric prior. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH ’23, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3588432.3591553.
 - [46] Yuxuan Han, Ruicheng Wang, and Jiaolong Yang. Single-view view synthesis in the wild with learned adaptive multiplane images. In *ACM SIGGRAPH*, 2022.
 - [47] Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. Speedy-splat: Fast 3d gaussian splatting with sparse pixels and sparse primitives. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 21537–21546, June 2025.
 - [48] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models, 2022. URL: <https://arxiv.org/abs/2210.02303>, arXiv:2210.02303.
 - [49] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.
 - [50] Tao Hu, Geng Lin, Zhizhong Han, and Matthias Zwicker. Learning to generate dense point clouds with textures on multiple categories. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2170–2179, January 2021.
 - [51] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024.
 - [52] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5885–5894, October 2021.
 - [53] Jinseo Jeong, Junseo Koo, Qimeng Zhang, and Gunhee Kim. Esr-nerf: Emissive source reconstruction using ldr multi-view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4598–4609, 2024.

- [54] Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. Tensorir: Tensorial inverse rendering. In *CVPR*, 2023.
- [55] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984.
- [56] Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. Layered neural atlases for consistent video editing. *ACM Transactions on Graphics (TOG)*, 40(6):1–12, 2021.
- [57] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, July 2023. URL: <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>.
- [58] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12912–12921, 2022.
- [59] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In *Computer Graphics Forum*, volume 40 (4), pages 29–43. Wiley Online Library, 2021.
- [60] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1440–1449, 2021.
- [61] Yao-Chih Lee, Ji-Ze Genevieve Jang, Yi-Ting Chen, Elizabeth Qiu, and Jia-Bin Huang. Shape-aware text-driven layered video editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- [62] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision*, pages 71–91. Springer, 2024.
- [63] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20775–20785, 2024.
- [64] Peihao Li, Shaohui Wang, Chen Yang, Bingbing Liu, Weichao Qiu, and Haoqian Wang. Nerf-ms: Neural radiance fields with multi-sequence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 18591–18600, October 2023.
- [65] Wenbin Li, Fabio Viola, Jonathan Starck, Gabriel J. Brostow, and Neill D.F. Campbell. Roto++: Accelerating professional rotoscoping using shape manifolds. *ACM Transactions on Graphics (In proceeding of ACM SIGGRAPH’16)*, 35(4), 2016.
- [66] Yin Li, Jian Sun, and Heung-Yeung Shum. Video object cut and paste. *ACM Trans. Graph.*, 24(3):595–600, 2005. doi:10.1145/1073204.1073234.

- [67] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 6498–6508. Computer Vision Foundation / IEEE, 2021. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Li_Neural_Scene_Flow_Fields_for_Space-Time_View_Synthesis_of_Dynamic_CVPR_2021_paper.html, doi:10.1109/CVPR46437.2021.00643.
- [68] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. Gs-ir: 3d gaussian splatting for inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21644–21653, 2024.
- [69] Shanchuan Lin, Andrey Ryabtsev, Soumyadip Sengupta, Brian Curless, Steve Seitz, and Ira Kemelmacher-Shlizerman. Real-time high-resolution background matting. *arXiv*, pages arXiv–2012, 2020.
- [70] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [71] 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 (NeurIPS)*, 2024.
- [72] Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- [73] Yuan Liu, Peng Wang, Cheng Lin, Xiaoxiao Long, Jiepeng Wang, Lingjie Liu, Taku Komura, and Wenping Wang. Nero: Neural geometry and brdf reconstruction of reflective objects from multiview images. In *SIGGRAPH*, 2023.
- [74] Erika Lu, Forrester Cole, Tali Dekel, Weidi Xie, Andrew Zisserman, David Salesin, William T Freeman, and Michael Rubinstein. Layered neural rendering for retiming people in video. In *SIGGRAPH Asia*, 2020.
- [75] Erika Lu, Forrester Cole, Tali Dekel, Andrew Zisserman, William T Freeman, and Michael Rubinstein. Omnimate: Associating objects and their effects in video. In *CVPR*, 2021.
- [76] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7210–7219, 2021.
- [77] Luke Melas-Kyriazi, Iro Laina, Christian Rupprecht, Natalia Neverova, Andrea Vedaldi, Oran Gafni, and Filippos Kokkinos. Im-3d: Iterative multiview diffusion and reconstruction for high-quality 3d generation. *arXiv preprint arXiv:2402.08682*, 2024.

- [78] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.
- [79] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.
- [80] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics (TOG)*, 43(6):1–19, 2024.
- [81] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. doi:10.1145/3528223.3530127.
- [82] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, July 2022.
- [83] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10349–10358, 2024.
- [84] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [85] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. Radiative backpropagation: An adjoint method for lightning-fast differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 39(4), July 2020. doi:10.1145/3386569.3392406.
- [86] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5589–5599, 2021.
- [87] Julian Ost, Issam Laradji, Alejandro Newell, Yuval Bahat, and Felix Heide. Neural point light fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18419–18429, 2022.
- [88] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016.

- [89] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017.
- [90] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022.
- [91] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering. *ACM Transactions on Graphics*, 4(43), 2024.
- [92] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022.
- [93] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- [94] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 42(4):1–12, 2023.
- [95] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL: <https://arxiv.org/abs/2112.10752>, arXiv:2112.10752.
- [96] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (ToG)*, 41(4):1–14, 2022.
- [97] Sara Sabour, Lily Goli, George Kopanas, Mark Matthews, Dmitry Lagun, Leonidas Guibas, Alec Jacobson, David Fleet, and Andrea Tagliasacchi. Spotlessplats: Ignoring distractors in 3d gaussian splatting. *ACM Transactions on Graphics*, 44(2):1–11, 2025.
- [98] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixel-wise view selection for unstructured multi-view stereo. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part III 14*, pages 501–518. Springer, 2016.
- [99] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [100] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixel-wise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

- [101] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 1, pages 519–528. IEEE, 2006.
- [102] Yahao Shi, Yanmin Wu, Chenming Wu, Xing Liu, Chen Zhao, Haocheng Feng, Jian Zhang, Bin Zhou, Errui Ding, and Jingdong Wang. Gir: 3d gaussian inverse rendering for relightable scene factorization. *IEEE Transactions on Transactions on Pattern Analysis and Machine Intelligence*, 2025.
- [103] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023.
- [104] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7495–7504, 2021.
- [105] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, pages 5459–5469, 2022.
- [106] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. *CVPR*, 2021.
- [107] Stanislaw Szymanowicz, Jason Y. Zhang, Pratul Srinivasan, Ruiqi Gao, Arthur Brussee, Aleksander Holynski, Ricardo Martin-Brualla, Jonathan T. Barron, and Philipp Henzler. Bolt3D: Generating 3D Scenes in Seconds. *arXiv:2503.14445*, 2025.
- [108] Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II*, volume 12347 of *Lecture Notes in Computer Science*, pages 402–419. Springer, 2020. doi:10.1007/978-3-030-58536-5_24.
- [109] Inc Trimble, Oct 2022. URL: <https://3dwarehouse.sketchup.com/>.
- [110] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd E. Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 5481–5490. IEEE, IEEE, 2022. doi:10.1109/CVPR52688.2022.00541.
- [111] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Path replay backpropagation: differentiating light paths using constant memory and linear time. *ACM Trans. Graph.*, 40(4), July 2021. doi:10.1145/3450626.3459804.

- [112] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitry Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. Sv3d: Novel multi-view synthesis and 3d generation from a single image using latent video diffusion. In *European Conference on Computer Vision*, pages 439–457. Springer, 2024.
- [113] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.
- [114] Jianyuan Wang, Nikita Karaev, Christian Rupprecht, and David Novotny. Vggsfm: Visual geometry grounded deep structure from motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21686–21697, 2024.
- [115] Jianyuan Wang, Yiran Zhong, Yuchao Dai, Stan Birchfield, Kaihao Zhang, Nikolai Smolyanskiy, and Hongdong Li. Deep two-view structure-from-motion revisited. *CVPR*, 2021.
- [116] Jue Wang, Pravin Bhat, Alex Colburn, Maneesh Agrawala, and Michael F. Cohen. Interactive video cutout. *ACM Trans. Graph.*, 24(3):585–594, 2005. doi:10.1145/1073204.1073233.
- [117] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [118] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024.
- [119] Tianyu Wang, Xiaowei Hu, Chi-Wing Fu, and Pheng-Ann Heng. Single-stage instance shadow detection with bidirectional relation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–11, June 2021.
- [120] Tianyu Wang, Xiaowei Hu, Qiong Wang, Pheng-Ann Heng, and Chi-Wing Fu. Instance shadow detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [121] Yi Wang, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, Yannick Benezeth, and Prakash Ishwar. Cdnet 2014: An expanded change detection benchmark dataset. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 393–400, 2014. doi:10.1109/CVPRW.2014.126.
- [122] Yuze Wang, Junyi Wang, and Yue Qi. We-gs: An in-the-wild efficient 3d gaussian representation for unconstrained photo collections. *arXiv preprint arXiv:2406.02407*, 2024.
- [123] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36:8406–8441, 2023.

- [124] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7467–7477, 2020.
- [125] Jay Zhangjie Wu, Yuxuan Zhang, Haithem Turki, Xuanchi Ren, Jun Gao, Mike Zheng Shou, Sanja Fidler, Zan Gojcic, and Huan Ling. Difx3d+: Improving 3d reconstructions with single-step diffusion models. *arXiv preprint arXiv:2503.01774*, 2025.
- [126] Jiaye Wu, Saeed Hadadan, Geng Lin, Matthias Zwicker, David Jacobs, and Roni Sengupta. Gani: Global and near field illumination aware neural inverse rendering, 2024. URL: <https://arxiv.org/abs/2403.15651>, arXiv:2403.15651.
- [127] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. D²nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. In *Advances in Neural Information Processing Systems*, 2022.
- [128] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.
- [129] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 9421–9431. Computer Vision Foundation / IEEE, 2021. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Xian_Space-Time_Neural_Irradiance_Fields_for_Free-Viewpoint_Video_CVPR_2021_paper.html, doi:10.1109/CVPR46437.2021.00930.
- [130] Haolin Xiong. *SparseGS: Real-time 360° sparse view synthesis using Gaussian splatting*. University of California, Los Angeles, 2024.
- [131] Jiacong Xu, Yiqun Mei, and Vishal Patel. Wild-gs: Real-time novel view synthesis from unconstrained photo collections. *Advances in Neural Information Processing Systems*, 37:103334–103355, 2024.
- [132] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5438–5448, 2022.
- [133] Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. *arXiv preprint arXiv:2501.13928*, 2025.
- [134] Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

- [135] Yifan Yang, Shuhai Zhang, Zixiong Huang, Yubing Zhang, and Mingkui Tan. Cross-ray neural radiance fields for novel-view synthesis from unconstrained image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15901–15911, 2023.
- [136] Ziyi Yang, Yanzhen Chen, Xinyu Gao, Yazhen Yuan, Yu Wu, Xiaowei Zhou, and Xiaogang Jin. Sire-ir: Inverse rendering for brdf reconstruction with shadow and illumination removal in high-illuminance scenes. *arXiv preprint*, 2023.
- [137] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European conference on computer vision (ECCV)*, pages 767–783, 2018.
- [138] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5525–5534, 2019.
- [139] Yao Yao, Jingyang Zhang, Jingbo Liu, Yihang Qu, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. Neilf: Neural incident light field for physically-based material estimation. In *ECCV*, 2022.
- [140] Yuxuan Yao, Zixuan Zeng, Chun Gu, Xiatian Zhu, and Li Zhang. Reflective gaussian splatting. In *ICLR*, 2025.
- [141] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Proceedings of the Thirty-Fifth Conference on Neural Information Processing Systems (NeurIPS)*, 2021.
- [142] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. Baked sdf: Meshing neural sdf for real-time view synthesis. *arXiv preprint*, 2023. [arXiv:2303.13485](https://arxiv.org/abs/2303.13485).
- [143] Vickie Ye, Zhengqi Li, Richard Tucker, Angjoo Kanazawa, and Noah Snavely. Deformable sprites for unsupervised video decomposition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.
- [144] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019.
- [145] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoc-trees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5752–5761, 2021.
- [146] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021.

- [147] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19447–19456, 2024.
- [148] Zehao Yu and Shenghua Gao. Fast-mvsnet: Sparse-to-dense multi-view stereo with learned propagation and gauss-newton refinement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1949–1958, 2020.
- [149] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 43(6):1–13, 2024.
- [150] Dongbin Zhang, Chuming Wang, Weitao Wang, Peihao Li, Minghan Qin, and Haoqian Wang. Gaussian in the wild: 3d gaussian splatting for unconstrained image collections. In *European Conference on Computer Vision*, pages 341–359. Springer, 2024.
- [151] Jiawei Zhang, Jiahe Li, Xiaohan Yu, Lei Huang, Lin Gu, Jin Zheng, and Xiao Bai. Cor-gs: sparse-view 3d gaussian splatting via co-regularization. In *European Conference on Computer Vision*, pages 335–352. Springer, 2024.
- [152] Jingyang Zhang, Yao Yao, Shiwei Li, Jingbo Liu, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. Neif++: Inter-reflectable light fields for geometry and material estimation. *International Conference on Computer Vision (ICCV)*, 2023.
- [153] Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. Modeling indirect illumination for inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18643–18652, 2022.
- [154] Zhe Zhang, Rui Peng, Yuxi Hu, and Ronggang Wang. Geomvsnet: Learning multi-view stereo with geometry perception. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21508–21518, 2023.
- [155] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 37, 2018. URL: <https://arxiv.org/abs/1805.09817>.
- [156] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *European conference on computer vision*, pages 145–163. Springer, 2024.
- [157] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings of IEEE Visualization 2001*, pages 29–538, 2001. doi:10.1109/VISUAL.2001.964490.
- [158] Matthias Zwicker, Jussi Räsänen, Mario Botsch, Carsten Dachsbacher, and Mark Pauly. Perspective accurate splatting. In *Proceedings of Graphics Interface*, volume 2004, pages 247–254, 01 2004.