UNIVERSITY OF CALIFORNIA
Los Angeles

SparseGS: Real-Time 360° Sparse View Synthesis using Gaussian Splatting

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Computer Science

by

Sairisheek Muttukuru

2024

ABSTRACT OF THE THESIS

SparseGS: Real-Time 360° Sparse View Synthesis
using Gaussian Splatting

by

Sairisheek Muttukuru
Master of Science in Computer Science
University of California, Los Angeles, 2024
Professor Achuta Kadambi, Chair

The problem of novel view synthesis has grown significantly in popularity recently with the introduction of Neural Radiance Fields (NeRFs) and other implicit scene representation methods. A recent advance, 3D Gaussian Splatting (3DGS), leverages an explicit representation to achieve real-time rendering with high-quality results. However, 3DGS still requires an abundance of training views to generate a coherent scene representation. In few shot settings, similar to NeRF, 3DGS tends to overfit to training views, causing background collapse and excessive floaters, especially as the number of training views are reduced. This work proposes a method to enable training coherent 3DGS-based radiance fields of 360° scenes from sparse training views. Depth priors are integrated with generative and explicit constraints to reduce background collapse, remove floaters, and enhance consistency from unseen viewpoints. Experiments show that this method outperforms base 3DGS by 6.4% in LPIPS and by 12.2% in PSNR, and NeRF-based methods by at least 17.6% in LPIPS on the MipNeRF-360 dataset with substantially less training and inference cost. Project website at: `https://tinyurl.com/sparsegs`.

The thesis of Sairisheek Muttukuru is approved.

Jens Palsberg

Bolei Zhou

Achuta Kadambi, Committee Chair

University of California, Los Angeles

2024

To my parents and family

for their unwavering love and support

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This thesis is the culmination of my research experience at UCLA and I would like to express my gratitude to those who assisted and supported me along the way. First, I would like to thank my advisor, Professor Achuta Kadambi, whose invaluable guidance has been a beacon throughout my academic journey. I would also like to extend my gratitude to my thesis commitee members, Professor Bolei Zhou and Professor Jens Palsberg for their willingness to review and support my work.

Furthermore, I wish to acknowledge the significant contributions of the coauthors of this paper: Haolin Xiong, Rishi Upadhyay, and Pradyumna Chari. Their unwavering dedication has been instrumental in shaping the outcome of this research endeavor.

I am also grateful to the PhD students at the lab for their crucial insights and advice at every juncture. Special thanks are due to Rishi and Haolin for their tireless efforts, even during late-night sessions.

Lastly, I am immensely grateful to my parents and family for their unwavering support and love throughout my academic journey.

PREVIOUS PUBLICATIONS

This thesis revises the following publication:

Sairisheek Muttukuru, Haolin Xiong, Rishi Upadhyay, Pradyumna Chari and Achuta Kadambi. SparseGS: Real-Time 360° Sparse View Synthesis using Gaussian Splatting *arXiv preprint arXiv:2312.00206* (2023), *Under Review.*

CHAPTER 1

Introduction

The problem of learning 3D representations from 2D images has long been of interest, but has always been a difficult challenge due to the inherent ambiguities in lifting data to higher dimensions. Neural Radiance Fields (NeRFs) [18, 1, 2] tackle this problem by training a neural network using 2D posed images to predict the color and density of points in 3D space. The simplicity and quality of NeRFs enabled significant advances in novel view synthesis and led to a flurry of follow-up work. 3D Gaussian Splatting [11], a more recent technique, builds off of the ideas of NeRFs but replaces the implicit neural network with with an explicit representation based on 3D gaussians. These gaussians are rendered using point-based splatting, which allows for high rendering speeds, but requires fine-grained hyperparameter tuning. While NeRFs and 3D gaussian splatting both perform well at the task of novel view synthesis, they tend to require large sets of training views, which can be difficult due to reasons such as variable lighting conditions, weather-dependent constraints, and logistical complexities. As a result, interest in techniques for few-shot novel view synthesis has grown.

The few-shot view synthesis problem is especially difficult because the inherent ambiguity in learning 3D structure from 2D images is significantly exacerbated: with few-views, many regions of scene have little to no coverage. Therefore, there are many possible incorrect 3D representations that can represent the 2D training views correctly but will have poor results when rendering novel views due to artifacts such as "floaters" [2], floating regions of high density irregularly positioned through the scene, "background collapse", when the background is represented by similar looking artifacts closer to cameras, or holes in the repre-

sentation. Previous works have tackled this problem through the introduction of constraints that regularize the variation between views or the rendered depth maps. To our knowledge, most recent prior works in this area are built on top of NeRF and as a result are limited by long training times and the inherent black-box nature of neural networks. The lack of transparency poses a notable constraint, preventing a direct approach to the problem at hand. Instead, these challenges are approached indirectly through the use of meticulously designed loss functions and constraints.

In this work, we introduce a technique for few-shot novel view synthesis built on top of 3D Gaussian Splatting. The explicit nature of our underlying representations enables us to introduce a key new operation: direct floater pruning. This operation identifies the parts of a rendered image that suffer due to floaters or background collapse and allows us to **directly edit** the 3D representation to remove these artifacts. As a result, we are afforded greater control over when this operation is applied and how selective it is based on the scene representation during training. We show that this operation provides significant benefits in common view synthesis metrics and allows us to operate on full 360° unbounded scenes, a setting which most current few-shot techniques do not handle. In addition to this new operator, we leverage recent advances in image generation diffusion models to provide supervision in regions with little coverage from training views and apply depth constraints based on a novel technique for rendering depth from 3D gaussians which allows for better depth optimization. Together, we show that these techniques can enable high-quality novel-view synthesis from sparse views that out-performs prior state of the art work.

1.1 Thesis Contributions

In summary, the thesis contributions are as follows:

- A novel technique for training 3D radiance fields from few-views in unbounded 360° settings. The technique provides improvements of up to 6.4% over Base 3DGS in and at least 17.6% over NeRF-based methods in LPIPS.
- A technique to estimate depth from a 3D Gaussian representation that allows for better depth optimization.
- A new explicit, adaptive operator on 3D representations to prune unwanted "floater" artifacts in point-based 3D representations.

CHAPTER 2

Background and Related Work

2.1 Background

The process of volume rendering usually begins by defining a 3D field of optical properties, namely density, $\sigma(\mathbf{x})$ and color $\mathbf{c}(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^3$. To produce 2D maps of visual appearance, these fields are integrated along rays of viewing direction $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$. For ease of notation, the scalar color and density fields are reparameterized to be functions of t along a given ray. The equation commonly used in modern volumetric rendering techniques, which assumes volume with absorption and emission but no scattering is as follows [34]:

$$\boldsymbol{C} = \int_0^D T(t) \cdot \sigma(t) \cdot \mathbf{c}(t)\ dt\ +\ T(D) \cdot \mathbf{c}_{\text{bg}} \tag{2.1}$$

where $\mathbf{c}_{bg}$ is the background color and $T(D)$ is the residual transmittance. Transmittance, $T(t)$ is defined as the differential probability of a ray of light getting to a particle from 0 to t without hitting any other particles. In this context, it is also expressed as:

$$T(0 \to t) = \exp\left(-\int_0^t \sigma(t)\ dt\right) \tag{2.2}$$

This integral is often approximated with intervals δ_i of homogenous media (constant density) along each ray:

$$\boldsymbol{C} = \sum_{i=1}^{N} T_i \cdot (1 - \exp -\sigma_i \delta_i) \cdot \mathbf{c}_i, \quad \text{where} \;\; T_k = \exp \sum_{i=1}^{k-1} -\sigma_i \delta_i \tag{2.3}$$

2.2 Neural Radiance Fields

Radiance Fields, 3D representations of radiance in a scene, were first introduced by Neural Radiance Fields [18] (NeRFs) which represent 3D scenes with a neural volumetric representation that learns a density and color for every point in $\mathbb{R}^3$. The final color of a ray can be written using volumetric rendering principles as a combination of the density and color at the sample points along the ray as follows:

$$\boldsymbol{C} = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i, \;\; \text{where} \;\; T_k = \prod_{i=1}^{k-1} (1 - \alpha_i), \quad \text{and} \;\; \alpha_k = 1 - \exp(-\delta_k \sigma_k) \tag{2.4}$$

which is equation (2.3) rewritten with alpha-compositing weights. NeRFs are then trained using gradient descent on an image reconstruction loss. NeRFs have encouraged a flurry of follow-up work for extending NeRFs with new types of data [17, 12], improving speed [20] or quality [1, 22], or applying them to novel tasks [45, 40]. A set of NeRF follow-ups that are particularly relevant to this topic are Mip-NeRF [1] and Mip-NeRF 360 [2]. Mip-NeRF [1] introduced the concept of anti-aliasing to NeRFs by rendering conical frustums rather than just single rays. As a result, they are able to anti-alias their renders and render at multiple resolutions at high quality. Mip-NeRF 360 was a follow-up work to Mip-NeRF that specifically tackled the problem of representing 360° unbounded scenes. This work also focuses on this setting as it is a challenging setting for sparse view techniques.

2.3 3D Gaussian Splatting

While NeRFs rely on a neural network to represent the radiance field, 3D Gaussian Splatting [11], is a recent technique for view synthesis that replaces the neural network with explicit 3D Gaussians. These gaussians are parameterized by their position, rotation, scaling, opacity, and a set of spherical harmonics coefficients for view-dependent color. Specifically, each gaussian is defined as:

$$G(\mathbf{x}) = \exp\left(\frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x}\right) \tag{2.5}$$

where $\mathbf{x}$ is defined in world space.

They are then rendered by and projecting the gaussians down to the image plane by computing screen space covariances and subsequently multiplying the Gaussian values by their α's and blending the resulting colors using the volume rendering equation. Given a view matrix W and the jacobian of the affine approximation of the projection matrix J the screen space covariance is computed as [49, 50]:

$$\Sigma' = JW\Sigma W^T J^T \tag{2.6}$$

Models are trained using the same reconstruction loss as NeRFs:

$$\mathcal{L} = (1-\lambda)\mathcal{L}_{\text{L1}} + \lambda\mathcal{L}_{\text{D-SSIM}} \tag{2.7}$$

where $\mathcal{L}_{\text{L1}}$ is an L1 loss and $\mathcal{L}_{\text{D-SSIM}}$ is an SSIM loss. Replacing ray-tracing and the neural

network process of NeRF-based methods with splatting and direct representation process offers significant run-time improvements and allows for real-time rendering during inference. In addition, 3D Gaussian representations are explicit rather than the implicit representations of NeRFs, which allows for more direct editing and easier interpretability. This work leverages this property for the technique which identifies and directly deletes floaters. Recent works have built on top of 3D gaussian splatting to perform a variety of downstream tasks including text-to-3d generation [36, 44], dynamic scene representation [15, 41], and animating humans [48]. At the present time, this work is one of the first to provide a technique for few-shot novel view synthesis using 3D gaussians.

2.4 Few Shot Novel View Synthesis

The problem of novel view synthesis from few images has received significant interest as many view synthesis techniques can require a prohibitively high number of views for real-world usage. Many early techniques [37, 46, 31] leverage multi-plane images (MPIs), which represent images by sub-images at different depths, to re-render depth and color from novel view points using traditional transformations. More recent techniques are built on top of the NeRF framework and tend to tackle the problem one of two ways: The first set of methods introduce constraints on the variation between views. An early example of this type of method was DietNeRF [9] which added constraints to ensure that high-level semantic features remained the same from different views since they contained the same object. Another example is RegNeRF [21] which applies both color and depth consistency losses to the outputs at novel views. ViP-NeRF [30] modifies the traditional NeRF framework to additionally compute the visibility of a point and then uses these outputs to regularize the visibility and alpha-blended depth of different views. The second set of methods approach the problem by adding depth priors to novel views to regularize outputs. SparseNeRF [38] falls into this category and uses a pre-trained depth estimation model to get psuedo-ground truth depth

maps which are then used for a local depth ranking loss. They additionally apply a depth smoothness loss to encourage rendered depth maps to be piecewise-smooth. DSNeRF [3] also uses additional depth supervision, but uses the outputs of the SfM pipeline (typically COLMAP [29, 28]) used to get camera poses instead of a pre-trained model. Beyond these two techniques, some prior works approach the problem through recent machine learning techniques. Specifically, Neo 360 [7] uses a tri-plane feature representation to enable their model to reason about 3D features more robustly. A recent technique, FSGS [47], also utilizes 3D Gaussians for sparse-view synthesis. This approach introduces an unpooling operation to densify Gaussians in a novel manner and also incorporates depth priors for enhanced performance. However, FSGS is designed to use 24 views, while all of our results use 12 or fewer. To our knowledge, Neo360 and FSGS are the only other techniques that explicitly tackle the problem of 360°few-shot novel-view synthesis.

CHAPTER 3

Method

The proposed method is comprised of four key components: a depth correlation loss, a diffusion loss, an image re-projection loss, and a floater pruning operation.

3.1 Rendering Depth from 3D Gaussians

Many of the components introduced rely on depth maps rendered from the 3D Gaussian representation. In order to compute these, three different techniques are utilized, each with different properties: alpha-blending, mode-selection, and softmax scaling.

3.1.1 Alpha-blending

Alpha-blending renders depth maps by following the same procedure used for rendering color images. Let $d_{x,y}^{\text{alpha}}$ denote the alpha-blended depth at a pixel x, y, one can calculate it as:

$$d_{x,y}^{\text{alpha}} = \sum_{i=1}^{N} T_i \alpha_i d_i \tag{3.1}$$

where T_i is the accumulated transmittance for the ith gaussian, α_i is the alpha-compositing weight, and d_i is the depth of the gaussian. (In this model, the sum $\sum_{i=1}^{N} T_i \alpha_i$ is almost always 1.)

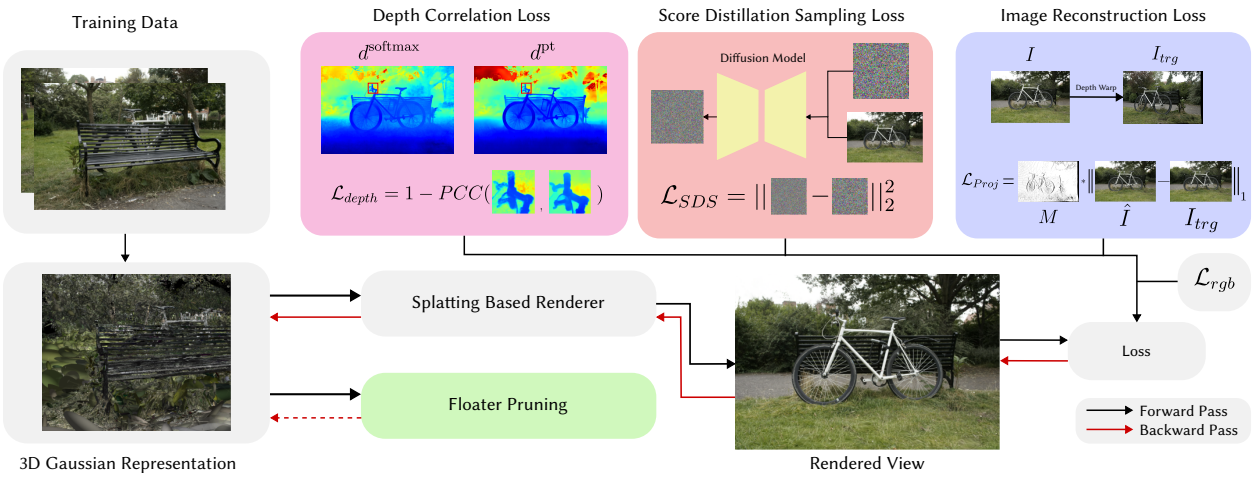


Figure 3.1: **The proposed pipeline incorporates depth priors, diffusion constraints, and a floater pruning technique to improve few-shot novel view synthesis performance.** During training, the softmax depth is rendered and Pearson correlation is used to encourage it to align with d^{pt} as described in section 3.2. Novel views are also generated using the procedure described in section 3.3 and a score distillation sampling loss is incorporated. At pre-set intervals, floaters are pruned as described in section 3.5. New components proposed are colored while the base 3DGS pipeline is in grey.

3.1.2 Mode-selection

In mode-selection, the depth of the gaussian with the largest contributing $w_i = T_i\alpha_i$ represents the depth of that pixel. The mode-selected depth of a pixel can be written as:

$$d^{\text{mode}}_{x,y} = d_{\arg\max_i(w_i)} \tag{3.2}$$

Intuitively, the mode-selected depth can be thought of as choosing the first high opacity gaussian while the alpha blended depth considers almost all gaussians along an imaginary ray from the camera. A key insight that is relied on later to build the pruning operator is that d^{mode} and d^{alpha} are not always the same even though they should be in the ideal case. Consider the toy setting on the left of fig. 3.2. d^{mode} is the depth of the second gaussian from the left since that is the one with the highest w_i, but d^{alpha} appears slightly behind this gaussian due to low weight gaussians with higher depths. This difference can lead to

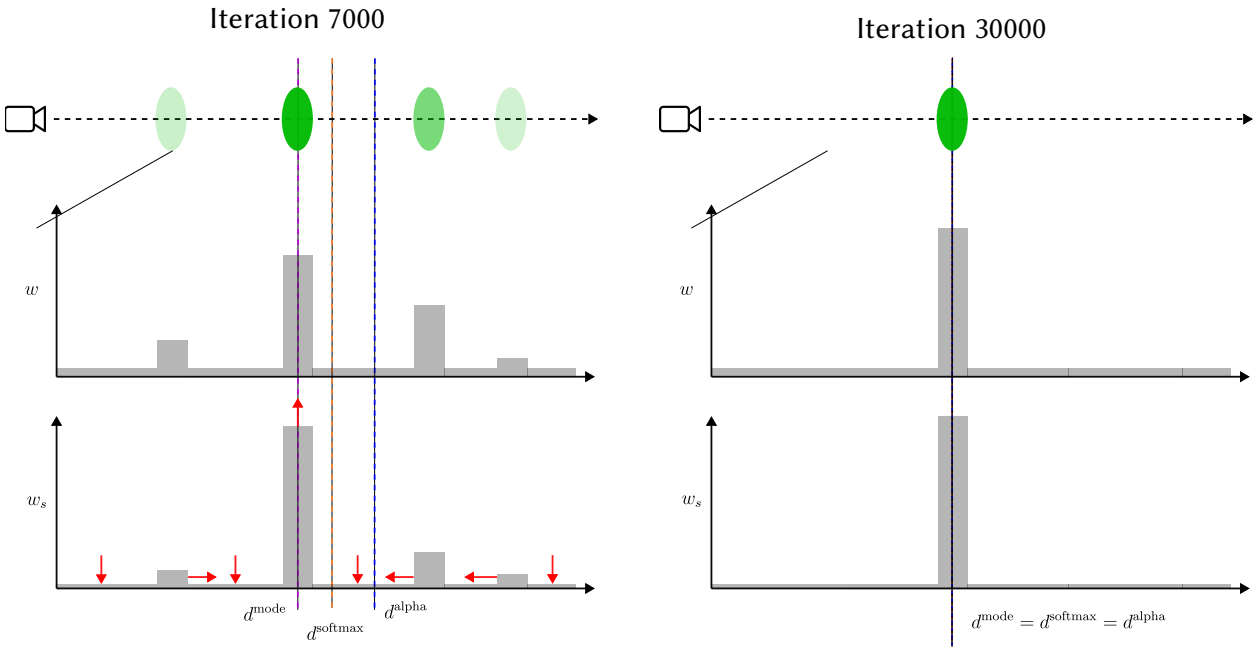


Figure 3.2: **A toy example that demonstrates the different kinds of depth: mode, alpha-blended, and softmax used in our technique and how depth correlation loss affects gaussian weights w_i.** w_i are shown on top and the weights after applying softmax are below. ***Left:*** Gaussian and weight distribution after 7000 iterations of training. Although there is a single gaussian with the highest weight, there are other low-weight gaussians nearby that affect both the rendering quality and the calculated depths. Shown in red, the loss encourages these smaller peaks to come closer to the tallest peak while also encouraging empty regions to go close to 0. ***Right:*** After training is complete, the depth correlation loss has consolidated low-weight gaussians into a single gaussian at the correct depth. All of the depth variations now agree.

ambiguity about what the true depth is. This ambiguity is interpreted as a measure of the uncertainty of 3DGS at a point and is used to inform the pruning operator introduced later.

3.1.3 Softmax-scaling

Although the mode depth works quite well for this purpose of identifying ambiguity in depth, the presence of the arg max operator in its equation means that any backward gradients only flow to one gaussian, the one with the highest w_i. This is not ideal, as it is optimal to influence all gaussians along this imaginary ray to converge on the correct depth. In order to overcome

this limitation of the mode depth, a new method is introduced to render depth that is a mix of the both alpha-blending and mode-selection, called the softmax depth:

$$d_{x,y}^{\text{softmax}} = \log\left(\frac{\sum_{i=1}^{N} w_i e^{\beta w_i} d_i}{\sum_{i=1}^{N} w_i e^{\beta w_i}}\right) \tag{3.3}$$

where the β parameter allows us to modulate the softmax temperature, thereby choosing a desired amplification of highly weighted gaussians. Note that:

$$\lim_{\beta \to 0} d_{x,y}^{\text{softmax}} = \log(d_{x,y}^{\text{alpha}}) \tag{3.4}$$

$$\lim_{\beta \to \infty} d_{x,y}^{\text{softmax}} = \log(d_{x,y}^{\text{mode}}) \tag{3.5}$$

Compared to mode depth, with the softmax approach, the mode depth can be approximated while still propagating gradients to gaussians surrounding the mode. Additionally, a psuedo-ground truth depth map is computed using a pre-trained depth estimation model on top of the training views. Monodepth [16, 32] is used in this work, but any pre-trained depth estimation model [26, 25] can work. The depth from this pre-trained model is referred to as d^{pt}.

3.2 Patch-based Depth Correlation Loss

Since the monocular estimation model predicts relative depth while the alpha-blended and mode-based depths are COLMAP-anchored depth, directly applying a loss such as mean squared error (MSE) would not work well. One option is to attempt to estimate scale and shift parameters and use those to align the two depth maps in metric space. However, the transformation between the depth maps is not guaranteed to be constant at all points, meaning a naive alignment could introduce additional unwanted distortion. Instead, Pearson correlation across image-patches is proposed to compute a similar metric between depth

maps. This is similar to the depth ranking losses proposed by prior work [38], but instead of comparing two selected points per iteration, entire patches can be compared, meaning larger portions can be affected of the image at once and can learn more local structure. The Pearson correlation coefficient is closely related to normalized cross-correlation; therefore, employing this loss encourages patches at the same location in both depth maps to have high cross-correlation, irrespective of the variations in depth value ranges.

At each iteration, N non-overlapping patches are randomly sampled to compute the depth correlation loss as:

$$\mathcal{L}_{depth} = \frac{1}{N}\sum_{i}^{N} 1 - PCC(p_i^{\text{softmax}}, p_i^{\text{pt}}) \tag{3.6}$$

$$PCC(X,Y) = \frac{\mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]}{\sqrt{\mathbb{E}[Y^2] - \mathbb{E}[Y]^2}\sqrt{\mathbb{E}[X^2] - \mathbb{E}[X]^2}} \tag{3.7}$$

where $p_i^{\text{softmax}} \in \mathbb{R}^{S^2}$ denotes the ith patch of $d^{\text{softmax}}$and $p_i^{\text{pt}} \in \mathbb{R}^{S^2}$ denotes the ith patch of d^{pt}, with the patch size S being a hyper-parameters. Intuitively, this loss works to align the softmax depth maps of the gaussian representation with the depth map of monodepth while avoiding the problem of inconsistent scale and shift.

3.3 Score Distillation Sampling Loss

In the few-shot setting, the sparsity of the input training data likely results in incomplete coverage of the scene. Therefore, inspired by recent advancements in image generative models [36, 27, 5, 23, 8, 43], the utilization of a pre-trained generative diffusion model is proposed to refine the 3D Gaussian representation via Score Distillation Sampling (SDS) as introduced in previous work [24].This allows refinement of plausible details for regions lacking perfect coverage in the training views and facilitates the generation of more complete 3D representations [14, 6, 13, 33]. The implementation begins by generating novel camera poses within the camera-to-world coordinate system. Random views are sampled, targeting the scene center

from an elliptic cylinder that best fits the training view positions. The SDS loss can then be formulated as:

$$\hat{I} = \mathcal{N}(\sqrt{\hat{\alpha}}I', (1-\hat{\alpha})\mathbf{I}) \tag{3.8}$$

$$\mathcal{L}_{SDS} = \nabla_G \mathbb{E}[(\epsilon_\phi(\hat{I}_p; \tilde{I}_p) - \epsilon)\frac{\partial \hat{I}_p}{\partial G}] \tag{3.9}$$

where G represents the parameters of our gaussian representation, $\hat{a}$ represents the cumulative product of one minus the variance schedule and ϵ is the noise introduced by the SDS, $\epsilon_\phi(\cdot)$ is the predicted noise by Stable Diffusion, $\hat{I}_p$ is the rendered image at camera pose p with added noise, and $\tilde{I}_p$ is the image denoised by Stable Diffusion. Intuitively, the SDS loss first finds the error between the true noise added to an image and the error the diffusion model estimates and then takes the gradient of this error with respect to the parameters of the gaussian model, i.e. the means, scalings, rotations, and opacities of the gaussians. Note that the usage of this SDS loss differs greatly from most prior uses of diffusion models for sparse view synthesis. While most prior techniques leverage diffusion models to fill in large blank regions or hallucinate new views that are not present in the original images (e.g., if only a front view is available and diffusion is used to hallucinate a back view), diffusion models and the SDS loss are used solely as a refinement step to enhance renders, which are anticipated to already encompass most of the correct structure of the 3D scene.

3.4 Image Re-projection Loss

Most 3D reconstruction models operate by fitting to the training views in the hope that the model can learn the underlying geometry of the scene. However, this approach becomes problematic in settings with extremely sparse input, as the model tends to overfit excessively [9, 21]. This overfitting leads to background collapse even with minimal changes in the viewing angle, undermining the model's ability to generalize and accurately reconstruct

the scene from new perspectives. In this study, image re-projection is employed, utilizing established depth warping techniques [19, 42, 3] to augment the available training data by re-projecting images to novel viewpoints. It is impractical to perform depth warping directly using outputs from the Monodepth model because they are in a relative scale. However, a solution to this limitation has been discovered by scaling the range to match the minimum and maximum values of rendered alpha-blended depth maps. As a result, the quality of the warping is highly dependent on the estimation of $\min_i d_i^{alpha}$ and $\max_i d_i^{alpha}$; however, in practical applications, it has been observed that the warping results are quite stable, provided that the depth Pearson loss has converged to a reasonable level. Mathematically, image re-projection can be defined as follows: For pixel $p_i(x_i, y_i)$ in training image I_{src}, the warping to the corresponding pixel $p_j(x_j, y_j)$ at an unseen viewpoint I_{trg} can be formulated as:

$$p_j = K_{trg} T (K_{src}^{-1} Z_i p_i) \tag{3.10}$$

where Z_i is the monodepth outputs scaled to the range of $\min_i d_i^{alpha}$ and $\max_i d_i^{alpha}$, K_{trg} and K_{src} are the intrinsic matrices of the source and the target camera, and T refers to the transformation between camera poses from viewpoint I_{src} to I_{trg}. A warp mask M, which marks which pixels were validly warped, is generated along with the process. The image re-projection loss L_{Proj} is formulated as:

$$\mathcal{L}_{Proj} = M * L_1(\hat{I}, I_{trg}) \tag{3.11}$$

3.5 Floater Removal

Although the model is trained with the depth correlation loss, optimizing the softmax depth alone does not solve the problem of "floaters". An example image with floaters is shown

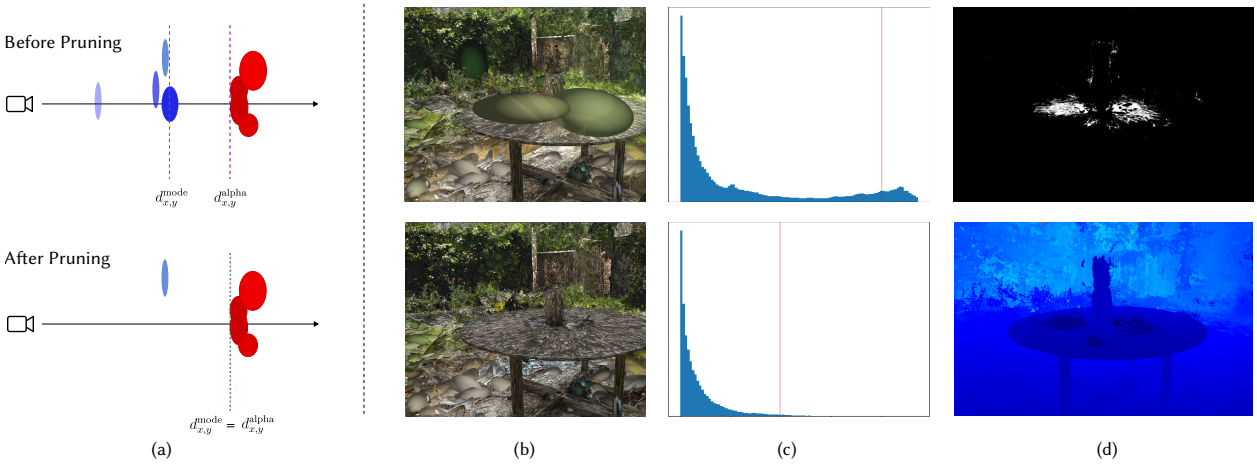


Figure 3.3: **The proposed floater pruning technique removes gaussians at inaccurate depths.** A toy example (a) demonstrates the pruning technique: before pruning, there are floaters (blue) in front of the gaussians at the object surface (red) and therefore, $d^{\text{mode}}_{x,y}$ $d^{\text{alpha}}_{x,y}$ are not aligned. The pruning technique removes all gaussians on that pixel before the mode and as a result, $d^{\text{mode}}_{x,y} = d^{\text{alpha}}_{x,y}$. Applied to the garden scene (b), the pruning technique removes large floaters in the center and left of the scene. There exists a link between the bimodality of the histogram of relative differences before and after the pruning operator in (c), with the appropriate pruning cutoff points indicated by the red vertical line. (d) top: The floater mask F_i is obtained from thresholding the relative differencing of the mode depth d^{mode} (bottom) at the cutoff points.

in fig. 3.3. To remove floaters, a novel operator is proposed, leveraging the explicit representation of 3D gaussians to eliminate floaters and encourage the model to re-learn those regions of the training views correctly. In our model, "floaters" often manifest themselves as relatively low-opacity Gaussians positioned close to the camera plane. Although they do not appear prominently when rendering the softmax depth of a scene due to being "averaged out," they are prominent in the mode-selected depth. This difference is leveraged to generate a floater mask F for each training view. Having identified this mask in image space, all Gaussians up to and including the mode Gaussian are selected and pruned. Specifically, F for a single training view i is computed as follows: first, Δ_i, the relative difference between the mode-selected depth and alpha-blended depth, is computed. When visualizing the distributions of Δ_i, it was observed that images with many floaters had bi-modal histograms, as floaters tend to be far from the true depth, while images without floaters tended to be

ALGORITHM 1: Algorithm to prune floaters

Function *prune_floaters*(G, P, a, b)
- **Input :** Gaussian Representation G
- **Input :** Camera Poses P
- **Input :** Curve Parameters a, b
- $\bar{D} \leftarrow 0.0$
- **foreach** $p \in P$ **do**
  - $d^{\text{alpha}} \leftarrow \text{alpha\_blend\_depth}(G, p)$
  - $d^{\text{mode}} \leftarrow \text{mode\_select}(G, p)$
  - $\Delta_p \leftarrow \frac{d^{\text{mode}} - d^{\text{alpha}}}{d^{\text{alpha}}}$
  - $\bar{D} \leftarrow \bar{D} + \text{dip\_test}(\Delta_p)$
- **end**
- $\bar{D} \leftarrow \bar{D}/|P|$
- **foreach** $p \in P$ **do**
  - $\tau_p \leftarrow \text{percentile}(\Delta_p, ae^{b\bar{D}})$
  - $F_p \leftarrow \mathbb{1}[\Delta_p > \tau]$
  - $g_0, g_1, ..., g_n \leftarrow \text{mask\_to\_gaussian}(F_p)$
  - $\text{remove\_gaussians}(g_0, g_1, ..., g_n)$
- **end**

end

more uni-modal. This phenomenon is visualized in fig. 3.3. Leveraging this, the dip test [4], a measure of uni-modality, is performed on the distribution. The uni-modality score is then averaged across all training views for a scene, since the number of floaters is generally a scene-wide metric, and used to select a cutoff threshold for the relative differences. The dip statistic to threshold conversion process is performed using an exponential curve with parameters a and b. These parameters are estimated by manually examining Δ_i and F_i for various scenes from different datasets and real-world captures. This process was carefully designed to allow floater pruning to be adaptive: in some cases, 3D scenes are already of quite high quality and, as a result, do not have many floaters. In that situation, setting a predefined threshold for the percent of pixels to prune would force detail deletion from the scene. Similarly, some scenes are especially difficult to learn (due to the 3D structure or distribution of training images), and in those cases, pruning should remove more floaters than normal. The average dip score $\bar{D}$ provides a proxy measure of how many floaters a

scene contains and allows adjustment of techniques in response. The full pruning process is visualized in fig. 3.3 and shown in algorithm 1.

3.6 Full Loss

Combined, the full loss is:

$$\mathcal{L} = \mathcal{L}_{rgb}(\hat{I}, I) + \lambda_{depth}\mathcal{L}_{depth}(d^{\text{softmax}}, d^{\text{pt}}) + \lambda_{SDS}\mathcal{L}_{SDS}(\hat{I}') + \lambda_{Proj}\mathcal{L}_{Proj}(\hat{I}_{trg}, I_{trg}) \quad (3.12)$$

where $\hat{I}$ is the rendered image, I is the ground truth image, and $\hat{I}'$ is a rendered image from a random novel viewpoint (not a training view). $\mathcal{L}_{rgb}$ is the same loss as is used to train base 3D Gaussian Splatting.

CHAPTER 4

Results

4.1 Mip-NeRF 360 Dataset

Since this technique is focused on unbounded, 360° scenes, evaluation is conducted on the Mip-NeRF 360 dataset designed specifically for this use case. The full dataset contains 9 scenes, but 6 of them are selected for having true 360° coverage. Comparison is made against Mip-NeRF 360 [2], DSNeRF [3], SparseNeRF [38], RegNeRF [21], and base 3D Gaussian Splatting [11]. DSNeRF, RegNeRF, and SparseNeRF are specifically designed for the few-shot setting, but are focused on frontal scenes. Mip-NeRF 360 and 3D Gaussian Splatting are designed for the general view synthesis case, but are also designed to handle 360° scenes. Thus, they are used as comparison benchmarks in the evaluation. For this setting, 12 training views are used. Results on PSNR, SSIM, and LPIPS are shown in table 4.1. Additionally, the runtimes of all methods are compared. The evaluation is split into two tables due to limitations in memory: running DSNeRF at higher resolutions was not possible within these constraints. On the higher resolution comparison (against Mip-NeRF 360, SparseNeRF, etc.), all baselines are outperformed on SSIM, LPIPS, and PSNR. Notably, Base 3DGS is outperformed by 12.2% in PSNR and 14.8% in SSIM. For LPIPS specifically, it is 17.6% lower than the next best performing technique, Mip NeRF 360. A similar pattern is observed for the comparison against DSNeRF, where DSNeRF is outperformed by 29.7% in LPIPS. Additionally, the technique trains in less than an hour and can run inference in real-time (100+ FPS). However, it is slightly outperformed by DSNeRF on PSNR. This is believed to be a result of the specific failure modes of the technique: NeRF based methods

Model	PSNR ↑	SSIM ↑	LPIPS ↓	Train Time (h)	Render FPS
SparseNeRF	11.5638	0.3206	0.6984	4	1/120
RegNeRF	11.7379	0.2266	0.6892	4	1/120
ViP-NeRF	11.1622	0.2291	0.7132	4	1/120
Mip-NeRF 360	<u>17.1044</u>	<u>0.4660</u>	0.5750	3	1/120
Base 3DGS	15.3840	0.4415	<u>0.5061</u>	0.5	120+
Ours	**17.2558**	**0.5067**	**0.4738**	0.75	120+

(a)

Model	PSNR ↑	SSIM ↑	LPIPS ↓	Train Time (h)	Render FPS
DSNeRF	**17.4934**	0.3620	0.6242	4	1/120
Base 3DGS	15.2437	<u>0.3914</u>	<u>0.4824</u>	0.5	120+
Ours	<u>16.7740</u>	**0.4341**	**0.4390**	0.75	120+

(b)

Table 4.1: **This model out-performs previous SOTA on SSIM and LPIPS for few-shot novel view synthesis on the Mip-NeRF 360 dataset.** Results in (a) were rendered at 1/2x while Results in (b) were at 1/4x resolution due to memory constraints with DSNeRF. All models in the same table were run at the same resolution for fairness. Training times were recorded on 1 RTX3090

tend to face the problem of having outputs that are too smooth, which was what originally prompted the introduction of the positional encoding [18, 35]. As a result, in these few-shot settings, the renders from NeRF exhibit an overly smooth appearance. Conversely, the 3D gaussian representation gravitates towards positioning isolated gaussians in regions of empty space. As a result, when the model does not have a good representation of a scene region, high-frequency artifacts arise. PSNR tends to tolerate overly smooth outputs while heavily penalizing high-frequency artifacts [39], leading to the poor performance of the model when compared to NeRF baselines. On a perceptual metric such as LPIPS, the model significantly outperforms the next closest baseline.

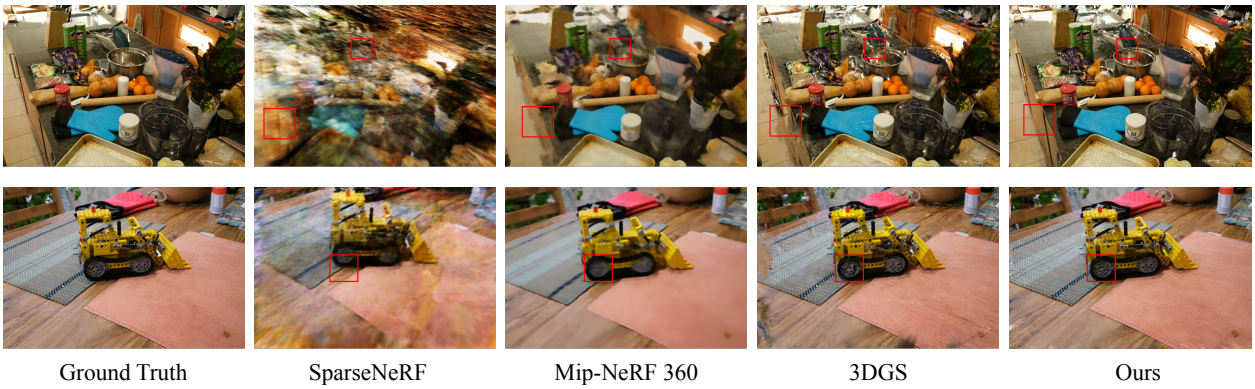


Figure 4.1: **Qualitative results from the Mip-NeRF 360 dataset show the model produces images that are sharper and perceptually similar to ground truth.** Red boxes show specific regions of interest.

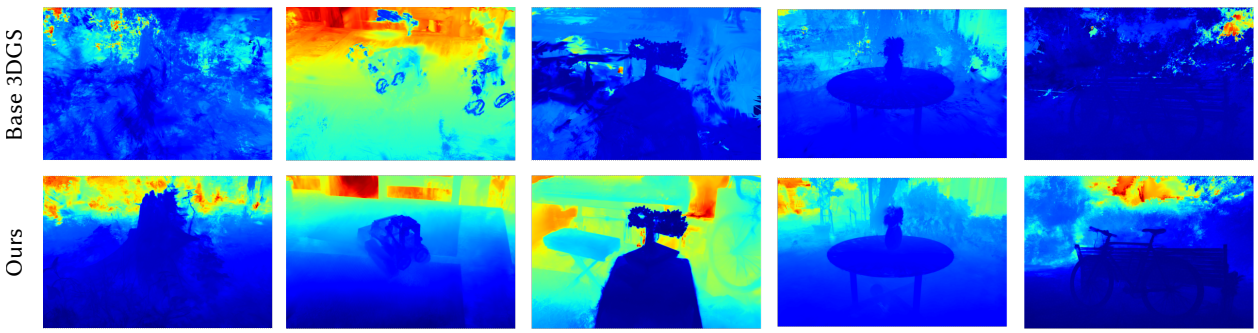


Figure 4.2: **Depth renders from Base 3DGS (top) and our model (bottom) for 5 scenes from the MipNeRF-360 dataset:** Stump, Kitchen, Bonsai, Garden, and Bicycle. Base 3DGS depths are generally incorrectly concentrated close to the camera view while our model learns more varied and detailed depths.

Model	PSNR ↑	SSIM ↑	LPIPS ↓
RegNeRF	18.89	0.745	0.190
SparseNeRF	19.55	0.769	0.201
DSNeRF	16.90	0.570	0.450
Base 3DGS	14.18	0.6284	0.301
Ours	18.89	0.702	0.229

Table 4.2: **Results on forward-facing scenes from the DTU dataset.** Our method is competitive with SOTA despite being designed for a different problem setting (unbounded 360 scenes). All models are trained with 3 input images.

Model	PSNR ↑	SSIM ↑	LPIPS ↓
Base 3DGS	15.05	0.497	0.518
+ Depth Correlation	16.52	0.587	0.438
+ Diffusion Loss	16.79	0.585	0.452
+ Depth Warping	16.93	0.598	0.438
+ Floater Pruning	**17.18**	**0.602**	**0.437**

Table 4.3: **An ablation study on the bonsai scene shows our full model performs best.** Each component individually provides benefits that combine to enable the performance of our full model. The best result is bolded.

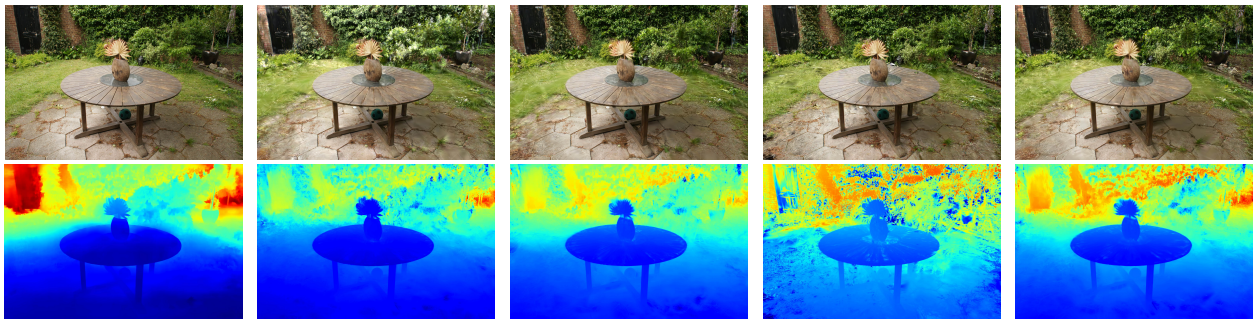
4.2 DTU Dataset

In addition to Mip-NeRF 360, comparisons are provided on forward-facing scenes from the DTU dataset [10]. Although the technique is designed for unbounded 360° scenes, these comparisons are provided because a large set of prior work on the task of few-shot novel view synthesis has been focused on the forward-facing setting. Results can be seen in table 4.2. The model consistently out-performs Base 3DGS and is competitive with RegNeRF and SparseNeRF despite the fact that the method is designed for a different problem setting.

4.3 Ablation Studies

4.3.1 Overall Ablation

Ablation studies are conducted to examine the effects of each of the individual components introduced. Specifically, the depth correlation loss, the score distillation sampling loss, the image re-projection loss, and floater pruning are ablated. Results are shown in table 4.3 and in fig. 4.3. As can be seen in table 4.3, the full model performs best on all three metrics, and the depth correlation loss, based on the novel depth formulation, provides the most benefit among the three components of the technique (drop of approx. 1.5 PSNR when removing



Ground Truth | w/o Floater Pruning | w/o Diffusion Loss | w/o Correlation Loss | Full Model (Ours)

Figure 4.3: **The full model performs best in terms of image renders and depth maps.** The full model combines the best of different subcomponents to generate high-quality depth maps (e.g. it retains the smoothness of the depth correlation loss while learning background details from floater pruning and diffusion). For depth maps, the ground truth is the output of the Monodepth model.

it). fig. 4.3 highlights the benefits of the depth correlation loss, particularly for the depth maps. Without the depth correlation loss, there are many discontinuities and sharp features in the depth map, likely because the model is excessively overfitting to the limited training views.

4.3.2 Softmax Depth Ablation

In table 4.4 a quantitative comparison between alpha-blended depth and softmax depth is provided. Two models are trained, both with only the depth correlation loss. One optimizes alpha-blended depth for this loss while the other optimizes softmax depth. On the full Mip-NeRF 360 dataset, the model trained with softmax depth correlation loss outperformed both base 3DGS and alpha-blended depth correlation loss, suggesting softmax depth can help the model learn better representations.

Model	PSNR ↑	SSIM ↑	LPIPS ↓
Base 3DGS	15.3840	0.4415	0.5061
Base + Alpha-Blended Depth	15.6749	0.4557	0.4998
Base + Softmax Depth	**16.5051**	**0.4972**	**0.4744**

Table 4.4: **Ablation study on the relative effectiveness of softmax depth vs alpha-blended depth.**

4.3.3 Training View Ablation

fig. 4.4 provides a study of how the model performs relative to base 3DGS as the number of training views is changed. We test 8, 10, 12, 14, 16, and 18 views. Across all three metrics, PSNR, SSIM, and LPIPS, the model consistently outperforms Base 3DGS in all settings. The model also performs well in the challenging 8-view setting, achieving results that are comparable to base 3DGS and better than most NeRF based methods at 12-view.

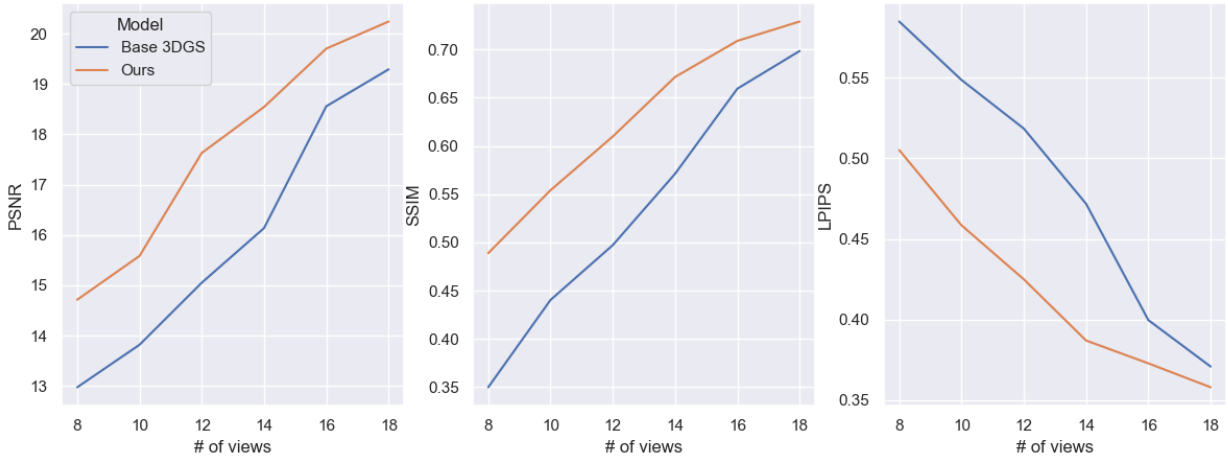


Figure 4.4: **Ablation study on the number of views used to train models.** The model consistently outperforms Base 3DGS on PSNR, SSIM, and LPIPS when we use as few as 8 or as many as 18 images for training.

Figure 4.5: **Qualitative ablation study on the benefits of the SDS loss.** Images from models without the SDS loss (top) exhibit high-frequency artifacts and dis-coloration. Images from models with the SDS loss (bottom) smooth these artifacts out and have higher quality reconstructions.

4.3.4 SDS Loss Ablation

fig. 4.5 provides qualitative examples of the model trained with and without our SDS loss. While the models trained without the SDS loss generate reasonable images, they exhibit many high-frequency artifacts, likely the result of many small highly anisotropic gaussians. Models trained with the SDS loss are able to largely avoid this issue. This highlights the design of the SDS loss: the diffusion model is not used to generate new details or unseen views, and instead refines regions of the scene representation and alleviates some of the limitations of the 3D Gaussian representation.

4.4 Implementation Details

The training procedure is built on top of the procedure introduced in 3D Gaussian Splatting [11]. The model is trained for 30k iterations with learning rates for the gaussian representation components taken from [11]. The depth correlation loss is applied every iteration with a patch size of 128 pixels and select 50% of all patches per iteration. The diffusion loss is applied randomly for 20% of iterations during the last 10k steps. Additionally, the floater pruning technique is applied at 20k iteration with $a = 97$ and $b = -8$.$\lambda_{depth} = 0.1$ and $\lambda_{SDS} = 5 \times 10^{-4}$. At 5k iteration, the point where the depth correlation loss has reasonably converged, alpha-blending depth maps are rendered for image re-projection. For each training camera, 4 extra warping viewpoints are generated by rotating it around the scene's center. The projection weight λ_{Proj} is set to 0.3.

CHAPTER 5

Discussion and Conclusion

5.1 Conclusion

One key limitation of the method is that because it is built on top of 3D Gaussian Splatting, it is heavily reliant on the initial point cloud provided by COLMAP. If this point cloud is inaccurate or lacks detail, the model may struggle to compensate adequately, leading to under-reconstructions, especially in areas distant from the scene center. This limitation is especially prominent in the sparse view setting, as our initial training views lack substantial coverage. As a result, the input point clouds are relatively small, with many tested scenes being initialized with fewer than 20 points in total. To address this challenge, future efforts could involve investigating point cloud densification techniques as data augmentations.

In this paper, a novel 3D gaussian splatting based technique for few-shot novel view synthesis is introduced. The explicit nature of the 3D gaussian representation is leveraged to introduce a novel pruning operator designed to reduce and remove "floater" artifacts. Applied to the Mip-NeRF 360 dataset, it is shown that the technique can achieve state-of-the-art performance on few shot novel view synthesis for 360° unbounded scenes.

CHAPTER 6

Appendix

6.1 Appendix Contents

This supplementary is organized as follows:

- section 6.2 shows the **gradient derivation** for the softmax depth formulation.
- section 6.3 provides additional **qualitative** comparisons on the Mip-NeRF 360 dataset.

6.2 Derivation of the Softmax Depth Gradient

In this section, we derive the gradients used for the softmax depth implemented in the rasterizer. We assume we are rendering a single pixel and that the gaussians G_i along the camera ray are ordered from closest to the camera plane to the farthest. (i.e. α_1 corresponds to the gaussian closest to the camera, while α_N corresponds to the farthest)

If we let $w_k = T_k \alpha_k$ and $T_k = \prod_{i=1}^{k-1}(1 - \alpha_i)$

$$d_{x,y}^{\text{softmax}} = \log\left(\frac{\sum_{i=1}^{N} w_i e^{\beta w_i} d_i}{\sum_{i=1}^{N} w_i e^{\beta w_i}}\right) \tag{6.1}$$

The derivative w.r.t the gaussian's camera z value is given below:

$$\frac{\partial d_{x,y}^{\text{softmax}}}{\partial d_k} = \frac{w_k e^{\beta w_k}}{\sum_{i=1}^{N} w_i e^{\beta w_i} d_i} \tag{6.2}$$

The derivative w.r.t the gaussian's alpha value is given below:

$$\frac{\partial d_{x,y}^{\text{softmax}}}{\partial \alpha_k} =$$

$$\frac{(1+\beta w_k) T_k e^{\beta w_k} d_k - \frac{1}{1-\alpha_k} \sum_{j=k+1}^{N} (1+\beta w_j) w_j e^{\beta w_j} d_j}{\sum_{i=1}^{N} w_i e^{\beta w_i} d_i} \tag{3}$$

$$- \frac{(1+\beta w_k) T_k e^{\beta w_k} - \frac{1}{1-\alpha_k} \sum_{j=k+1}^{N} (1+\beta w_j) w_j e^{\beta w_j}}{\sum_{i=1}^{N} w_i e^{\beta w_i}} \tag{4}$$

The sums in the denominators are retained during the forward pass and subsequently transferred to the backward pass. To avoid an $O(n^2)$ blowup, an accumulator strategy is used as in [11].

Let $\mathcal{A}_k^l, \mathcal{A}_k^r$ be the accumulators pertaining to $\frac{\partial d_{x,y}^{\text{softmax}}}{\partial \alpha_k}$ for expressions (3) and (4) respectively.

$$\mathcal{A}_0^l = \mathcal{A}_0^r = \alpha_0 = 0$$
$$\mathcal{A}_k^l = \alpha_{k-1}(1+\beta w_{k-1}) e^{\beta w_{k-1}} d_{k-1} + (1-\alpha_{k-1}) \mathcal{A}_{k-1}^l$$
$$\mathcal{A}_k^r = \alpha_{k-1}(1+\beta w_{k-1}) e^{\beta w_{k-1}} + (1-\alpha_{k-1}) \mathcal{A}_{k-1}^r$$

It can be shown that

$$\frac{\partial d_{x,y}^{\text{softmax}}}{\partial \alpha_k} = T_k \left(\frac{(1+\beta w_k) e^{\beta w_k} d_k - \mathcal{A}_k^l}{\sum_{i=1}^{N} w_i e^{\beta w_i} d_i} - \frac{(1+\beta w_k) e^{\beta w_k} - \mathcal{A}_k^r}{\sum_{i=1}^{N} w_i e^{\beta w_i}} \right)$$

6.3 Additional Comparisons on Mip-NeRF 360 Dataset

In this section, we provide more extensive qualitative comparisons on the Mip-NeRF 360 dataset. The images in fig. 6.1 show that our models render images that are more consistent (fewer floaters and color artifacts) while also including high-frequency details.

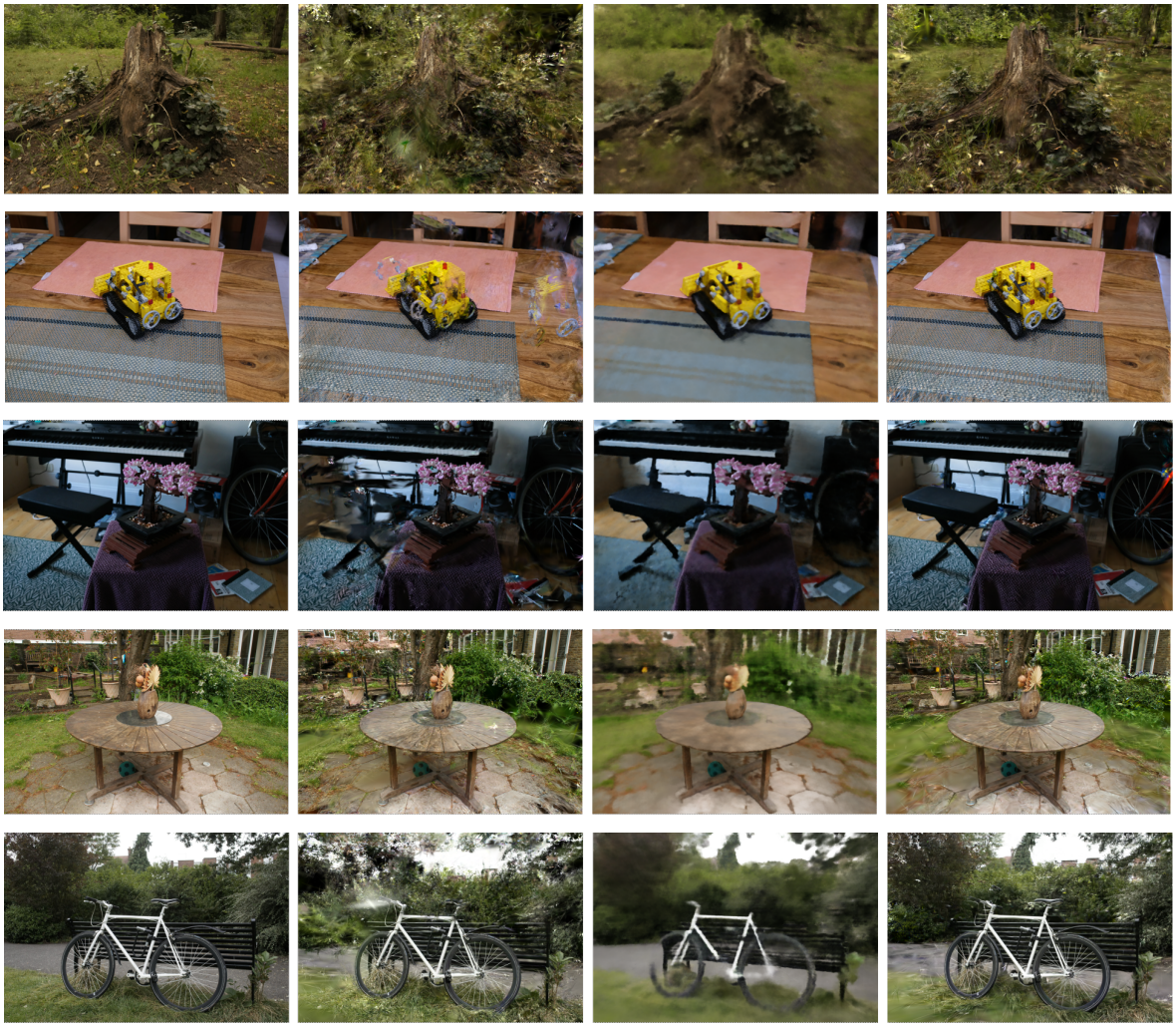


Figure 6.1: **Qualitative Results on the MipNeRF-360 dataset.** Scenes are 'Stump', 'Kitchen', 'Bonsai', 'Garden', and 'Bicycle'.

REFERENCES

[1] 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. *ICCV*, 2021.

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022.

[3] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[4] J. A. Hartigan and P. M. Hartigan. The Dip Test of Unimodality. *The Annals of Statistics*, 13(1):70 – 84, 1985.

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[6] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023.

[7] Muhammad Zubair Irshad, Sergey Zakharov, Katherine Liu, Vitor Guizilini, Thomas Kollar, Adrien Gaidon, Zsolt Kira, and Rares Ambrus. Neo 360: Neural fields for sparse view synthesis of outdoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9187–9198, 2023.

[8] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. 2022.

[9] 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.

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

[11] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

[12] Aviad Levis, Pratul P Srinivasan, Andrew A Chael, Ren Ng, and Katherine L Bouman. Gravitationally lensed black hole emission tomography. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 19841–19850, 2022.

[13] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Zexiang Xu, Hao Su, et al. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *arXiv preprint arXiv:2306.16928*, 2023.

[14] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023.

[15] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023.

[16] S. Mahdi H. Miangoleh, Sebastian Dille, Long Mai, Sylvain Paris, and Yağız Aksoy. Boosting monocular depth estimation models to high-resolution via content-adaptive multi-resolution merging. In *Proc. CVPR*, 2021.

[17] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P. Srinivasan, and Jonathan T. Barron. NeRF in the dark: High dynamic range view synthesis from noisy raw images. *CVPR*, 2022.

[18] 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.

[19] Yuji Mori, Norishige Fukushima, Toshiaki Fujii, and Masayuki Tanimoto. View generation with 3d warping using depth information for ftv. In *2008 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, pages 229–232, 2008.

[20] 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, July 2022.

[21] 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.

[22] Henry Peters, Yunhao Ba, and Achuta Kadambi. pcon: Polarimetric coordinate networks for neural scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[23] Ryan Po, Wang Yifan, Vladislav Golyanik, Kfir Aberman, Jonathan T Barron, Amit H Bermano, Eric Ryan Chan, Tali Dekel, Aleksander Holynski, Angjoo Kanazawa, et al. State of the art on diffusion models for visual computing. *arXiv preprint arXiv:2310.07204*, 2023.

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

[25] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *ArXiv preprint*, 2021.

[26] 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 (TPAMI)*, 2020.

[27] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[28] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[29] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[30] Nagabhushan Somraj and Rajiv Soundararajan. Vip-nerf: Visibility prior for sparse input neural radiance fields. August 2023.

[31] Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. *CVPR*, 2019.

[32] Gabriela Ben Melech Stan, Diana Wofk, Scottie Fox, Alex Redden, Will Saxton, Jean Yu, Estelle Aflalo, Shao-Yen Tseng, Fabio Nonato, Matthias Muller, et al. Ldm3d: Latent diffusion model for 3d. *arXiv preprint arXiv:2305.10853*, 2023.

[33] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[34] Andrea Tagliasacchi and Ben Mildenhall. Volume rendering digest (for nerf), 2022.

[35] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[36] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023.

[37] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 551–560, 2020.

[38] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[39] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[40] Chung-Yi Weng, Brian Curless, Pratul P. Srinivasan, Jonathan T. Barron, and Ira Kemelmacher-Shlizerman. HumanNeRF: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16210–16220, June 2022.

[41] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023.

[42] Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Humphrey Shi, and Zhangyang Wang. Sinnerf: Training neural radiance fields on complex scenes from a single image. 2022.

[43] Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Yi Wang, and Zhangyang Wang. Neurallift-360: Lifting an in-the-wild 2d photo to a 3d object with 360° views. 2022.

[44] Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *arXiv preprint arXiv:2310.08529*, 2023.

[45] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (ToG)*, 40(6):1–18, 2021.

[46] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[47] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting, 2023.

[48] Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. Drivable 3d gaussian avatars. *arXiv preprint arXiv:2311.08581*, 2023.

[49] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, pages 29–538, 2001.

[50] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001.