A Construct-Optimize Approach to Sparse View Synthesis without Camera Pose—*Supplementary Material*

Kaiwen Jiang
kevinjiangedu@gmail.com
University of California, San Diego
United States of America

Yang Fu
yangfuwork@gmail.com
University of California, San Diego
United States of America

Mukund Varma T
mukundvarmat@gmail.com
University of California, San Diego
United States of America

Yash Belhe
yashbelhe2008@gmail.com
University of California, San Diego
United States of America

Xiaolong Wang
xiw012@ucsd.edu
University of California, San Diego
United States of America

Hao Su
haosu@ucsd.edu
University of California, San Diego
United States of America

Ravi Ramamoorthi
ravir@cs.ucsd.edu
University of California, San Diego
United States of America

ACM Reference Format:
Kaiwen Jiang, Yang Fu, Mukund Varma T, Yash Belhe, Xiaolong Wang, Hao Su, and Ravi Ramamoorthi. 2024. A Construct-Optimize Approach to Sparse View Synthesis without Camera Pose—*Supplementary Material*. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3641519.3657427

We discuss implementation details, evaluation details, algorithm derivation and details, and additional evaluation results in the supplementary.

1 IMPLEMENTATION DETAILS

1.1 Details of Back-projecting Pixels into 3D Gaussians

Given a pixel at screen-space coordinates $\mathbf{s}$, it is associated with an estimated depth $l(\mathbf{s})$. Assume that the ray origin and corresponding ray direction are $\mathbf{o}$ and $\mathbf{d}(\mathbf{s})$. From our proposed approximate surface rendering in Sec. 3.3, we want to place a splat such that $D(\mathbf{s}) = l(\mathbf{s})$, where $D(\mathbf{s})$ denotes the rendered depth at $\mathbf{s}$. Trivially placing a splat with small scaling at $\mathbf{o} + l(\mathbf{s})\mathbf{d}(\mathbf{s})$ is plausible but it reduces the benefits provided by 3D Gaussians back to simply using points.

Given our approximate surface rendering algorithm in Sec. 3.3 of the main paper, we can map each splat to its ellipsoid shell. Note that in the following discussion, we talk about how to determine the center and scaling of the ellipsoid shell instead of 3D Gaussians directly. To fill up holes in the space as much as possible without affecting depth rendering at any other pixel, we propose

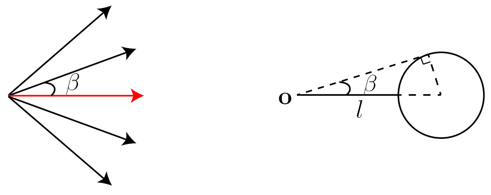

Figure 1: (a) β is the minimum angle between the red ray and its neighbors. (b) We use β and l to set the size and center of the sphere, such that it is as large as possible without intersecting any black rays.

to set the scaling and the center of the ellipsoid shell to $\frac{\sin\beta}{1-\sin\beta}l$, and $\mathbf{o} + \frac{l}{1-\sin\beta}\mathbf{d}(\mathbf{s})$ respectively, as shown in Fig. 1 (b). Here, β is the minimum angle between the current ray and neighboring rays, as shown in Fig. 1 (a). Therefore, both the scaling and center are differentiable functions of l, which itself is obtained from the (differentiable) monocular depth, enabling a fully differentiable framework.

While back-projecting pixels of newly registered views into 3D Gaussians, we are only interested in projecting pixels corresponding to newly observed regions. Therefore, we define a mask M by:

$$M(\mathbf{s}) = [D(\mathbf{s}) == \varnothing] \vee [D(\mathbf{s}) - l(\mathbf{s}) > \tau], \tag{1}$$

where $M(\mathbf{s})$ denotes the value of M at $\mathbf{s}$, $D(\mathbf{s}) == \varnothing$ denotes the emitting ray from $\mathbf{s}$ hits nothing, and $\tau = 10$. Only pixels with $M(\mathbf{s}) == 1$ are back-projected.

1.2 Details of Parameterizing the Optimization over Monocular Depth

As stated in the Sec. 3.1 of main paper, due to the sparsity of correspondences, we have to rely on the scale-consistency assumption in monocular depth estimation. However, this assumption can easily break when there are multiple objects in the scene. Therefore, we relax the constraint by learning a separate scaling and shift for each detected object in the scene. The objects are detected using the method described by Yu et al. [2023].

1.3 Details of Imposing Low-pass Filter

As stated in Sec. 3.4 of main paper, after obtaining a coarse solution, we apply a low-pass filter to it and then conduct refinement. To fill up the holes resulting from the filtering, we expand the scales of the remaining 3D Gaussians based on their spatial density as in [Kerbl et al. 2023].

1.4 Details of Optimization Configuration

Our optimization setup mostly follows Kerbl et al. [2023]. While constructing the coarse solution, we use the following learning rates: 0.01 for the quaternion for each view, 0.1 for the translation for each view and 0.01 for each object in the monocular depth. We only retain correspondence points with confidence over 0.5 for the detected correspondence [Sun et al. 2021; Tang et al. 2022].

For the coarse solution, in the adjustment, as stated in the main paper, we have to fit all seen views. Assuming the $k+1^{\text{th}}$ view has just been registered, we optimize the $k+1^{\text{th}}$ view with 50% chance, and all previous views with the remaining 50% chance. These hyper-parameters depend on the camera movements during the capture; we have reported the ones that worked well in our experiments.

Finally, we further refine the camera parameters when refining the coarse solution using standard method [Kerbl et al. 2023], which is possible as a result of the differentiable forward model. We linearly decay the learning rate for the quaternions from 0.0001 to 0.000001, and for translation from 0.001 to 0.00001. The optimization consumes $\sim$ 1 hour on an NVIDIA RTX 3080 GPU.

2 EVALUATION DETAILS

2.1 Dataset Pre-processing

We use the exact same Tanks&Temples dataset in [Bian et al. 2023], and pre-process the Static Hikes dataset in [Meuleman et al. 2023] by truncating each scene such that 3 training views can cover the entire scene.

2.2 Registration of Testing Views

Since we split all views into training views and testing views, testing views need to be registered after reconstruction of training views.

For methods that rely on off-the-shelf estimated camera poses, we again leverage SfM [Schönberger and Frahm 2016; Schönberger et al. 2016] to register testing views but without bundle adjustment for fairness.

For methods that do not rely on off-the-shelf estimated camera poses, following Bian et al. [2023], we conduct a post-training optimization of the camera poses of the testing views, based on the RGB loss only. The camera pose of the next unregistered testing view is initialized with the corresponding value for the last registered testing view. For our method, we optimize quaternion with a learning rate of 0.001 and translation with a learning rate of 0.01 for the testing views based on the RGB loss only.

3 COARSE SOLUTION CONSTRUCTION ALGORITHM

Our coarse solution construction algorithm is summarized in Alg. 1. The usage of semantic masks in parameterizing optimization of depths is discussed in Sec. 3.2 of main paper but omitted here for simplicity.

Algorithm 1 Coarse solution construction

```
Input: $n$ consecutive frames $\mathcal{I} = \{I^{(1)}, I^{(2)}, ..., I^{(n)}\}$, and their estimated monocular depths $\mathcal{D} = \{D^{(1)}, D^{(2)}, ..., D^{(n)}\}$. Camera intrinsic matrix $K$.
Output: A reconstructed scene $\mathcal{S}$ represented with Gaussian splatting, and the extrinsic matrices for $n$ frames $\mathcal{P} = \{P^{(1)}, P^{(2)}, ..., P^{(n)}\}$.
1: $\mathcal{S} \leftarrow BackProject(\mathcal{S}, I^{(1)}, D^{(1)}, \mathbb{1})$    ▷ Back-project all pixels into 3D Gaussians based on depth
2: for $i \leftarrow 2$ to $n$ do
3:     $P^{(i)} \leftarrow P^{(i-1)}$    ▷ Initialization
4:     $P^{(i)} \leftarrow FitImage(\mathcal{S}, I^{(i)}, P^{(i)})$    ▷ Registration
5:     $\mathcal{I}' \leftarrow \{I^{(1)}, I^{(2)}, ..., I^{(i)}\}$
6:     $\mathcal{P}' \leftarrow \{P^{(1)}, P^{(2)}, ..., P^{(i)}\}$
7:     $\mathcal{P}', \mathcal{S}', D'^{(i)} \leftarrow FitImage(\mathcal{S}, \mathcal{I}', \mathcal{P}', D^{(i)})$    ▷ Adjustment
8:     $M \leftarrow MaskEstimation(\mathcal{S}, I^{(i)}, P^{(i)})$
9:     $\mathcal{S} \leftarrow BackProject(\mathcal{S}, I^{(i)}, D'^{(i)}, M)$    ▷ Back-projection
```

4 NUMERICALLY STABLE APPROXIMATE SURFACE DIFFERENTIABLE RENDERING

We start from a brief simplified review of the color rendering in Kerbl et al. [2023] in Sec. 4.1, and then extend it to surface rendering in Sec. 4.2. Finally, we provide a numerically stable ray-ellipsoid intersection algorithm in Sec. 4.3.

4.1 Review of rendering in Gaussian Splatting

We are interested in computing the color for screen-space point $\mathbf{x} = (x, y)$ [Mildenhall et al. 2022], given by

$$I(\mathbf{x}) = \int_0^\infty \mathbf{c}(\mathbf{x}, t)\sigma(\mathbf{x}, t) \exp\left(-\int_0^t \sigma(\mathbf{x}, s) ds\right) dt. \tag{2}$$

Here, t denotes the third dimension, i.e., depth.

Following several splatting based rendering algorithms [Kerbl et al. 2023; Zwicker et al. 2001, 2002], $\sigma(\mathbf{x}, t)$ is reformulated as

$$\sigma(\mathbf{x}, t) = \sum_{i=1}^{N} \omega_i(\mathbf{x}, t) r_i(\mathbf{x}, t), \tag{3}$$

where r_i denotes the i^{th} reconstruction kernel evaluated at $(\mathbf{x}, t)$ and $\omega_i(\mathbf{x}, t)$ denotes its weight.

Therefore, by swapping the integral and summation, Eqn. 2 can be formulated as

$$\begin{aligned} I(\mathbf{x}) = \sum_{i=1}^{N} \int_0^\infty & \mathbf{c}(\mathbf{x}, t)\omega_i(\mathbf{x}, t) r_i(\mathbf{x}, t) \\ & \prod_{j=1}^{N} \exp\left(-\int_0^t \omega_j(\mathbf{x}, s) r_j(\mathbf{x}, s)) ds\right) dt. \end{aligned} \tag{4}$$

Assumption 1. *Reconstruction kernels are locally supported or approximately truncated and they have no overlap with one another. Specifically, for any depth $\widehat{t}$, there is a single index $k \in \{1, ..., N\}$, such that $\sigma(\mathbf{x}, \widehat{t}) \approx \omega_k(\mathbf{x}, \widehat{t}) r_k(\mathbf{x}, \widehat{t})$.*

Using the above assumption, we have

$$\int_0^{\infty} F(t)\omega_i(\mathbf{x}, t)r_i(\mathbf{x}, t)dt \approx \int_{\alpha_i}^{\beta_i} F(t)\omega_i(\mathbf{x}, t)r_i(\mathbf{x}, t)dt, \quad (5)$$

where α_i, β_i denote the starting and ending points of the local support for the i^{th} reconstruction kernel, and $F(t)$ is any t-dependent function. Furthermore, we sort reconstruction kernels based on their centers along the depth axis such that

$$\forall i, \forall j > i, \alpha_j > \beta_i. \quad (6)$$

This implies that

$$\forall i, \forall j > i, \int_0^{\beta_i} \omega_j(\mathbf{x}, s)r_j(\mathbf{x}, s)ds \approx 0. \quad (7)$$

Plugging Eqn. 5 and Eqn. 7 into Eqn. 4, we have

$$\begin{aligned} I(\mathbf{x}) \approx \sum_{i=1}^{N} \int_0^{\infty} \mathbf{c}(\mathbf{x}, t)\omega_i(\mathbf{x}, t)r_i(\mathbf{x}, t) \\ \prod_{j=1}^{i-1} \exp\left(-\int_0^{t} \omega_j(\mathbf{x}, s)r_j(\mathbf{x}, s)\right)ds)dt. \end{aligned} \quad (8)$$

By expanding the first-order Taylor series of the exponential function, we have

$$I(\mathbf{x}) \approx \sum_{i=1}^{N} \int_0^{\infty} \mathbf{c}(\mathbf{x}, t)\omega_i(\mathbf{x}, t)r_i(\mathbf{x}, t) \prod_{j=1}^{i-1} (1 - \int_0^{t} \omega_j(\mathbf{x}, s)r_j(\mathbf{x}, s))ds)dt. \quad (9)$$

Again plugging Eqn. 5 into Eqn. 9, we can change the upper bound of inner integral for alignment as

$$I(\mathbf{x}) \approx \sum_{i=1}^{N} \int_0^{\infty} \mathbf{c}(\mathbf{x}, t)\omega_i(\mathbf{x}, t)r_i(\mathbf{x}, t) \prod_{j=1}^{i-1} (1 - \int_0^{\infty} \omega_j(\mathbf{x}, s)r_j(\mathbf{x}, s))ds)dt \quad (10)$$

Assumption 2. *Color is constant inside each reconstruction kernel, i.e., $\mathbf{c}(\mathbf{x}, t) = \mathbf{c}_i$.*

From Assumption. 2, we can move $\mathbf{c}(\mathbf{x}, t)$ outside the integral

$$I(\mathbf{x}) \approx \sum_{i=1}^{N} \mathbf{c}_i (\int_0^{\infty} \omega_i(\mathbf{x}, t)r_i(\mathbf{x}, t)dt) \prod_{j=1}^{i-1} (1 - \int_0^{\infty} \omega_j(\mathbf{x}, s)r_j(\mathbf{x}, s))ds) \quad (11)$$

By choosing a Gaussian reconstruction kernel, further convolutions and transformations applied to $I(\mathbf{x})$ can be reduced to the operations on $r_i(\mathbf{x}, t)$. The integral has a closed form solution given by,

$$I(\mathbf{x}) \approx \sum_{i=1}^{N} \mathbf{c}_i \alpha_i(\mathbf{x}) \prod_{j=1}^{i-1} (1 - \alpha_j(\mathbf{x})), \quad (12)$$

where $\alpha_i(\mathbf{x})$ denotes the alpha blending coefficients of i^{th} Gaussian kernel, as in Kerbl et al.[2023].

4.2 Extension to approximate surface rendering

In previous works, surface rendering is usually achieved by rendering the depth, which is given by

$$D(\mathbf{x}) \approx \sum_{i=1}^{N} d_i \alpha_i(\mathbf{x}) \prod_{j=1}^{i-1} (1 - \alpha_j(\mathbf{x})), \quad (13)$$

where d_i denotes the z-axis coordinate for the center of the Gaussian kernel in the camera space.

This surface rendering model is based on an assumption similar to Assumption. 2, according to which the depth is constant inside each reconstruction kernel. However, it is too strong that it ignores the shape of the Gaussian kernel, making the depth independent of rotation and scaling. A better surface approximation which is both anisotropic and scale-dependent is vital for optimization.

Actually, similar to Eqn. 10, from the definition of expected surface points, we can have

$$\Psi(\mathbf{x}) \approx \sum_{i=1}^{N} (\prod_{j=1}^{i-1} (1 - \int_0^{\infty} \omega_j(\mathbf{x}, s)r_j(\mathbf{x}, s))ds))(\int_0^{\infty} q\omega_i(\mathbf{x}, t)r_i(\mathbf{x}, t)dt), \quad (14)$$

where $q = (\mathbf{x}, t)^T$ and $\Psi(\mathbf{x}) \in \mathbb{R}^3$ denotes the expected surface points for screen-space point $\mathbf{x}$. Therefore, we can see that q only appears in the right integral, excluding effects from reconstruction kernels in front of current one, and q lies exactly on the ray emitting from $\mathbf{x}$.

We then expect to propose a similar assumption to Assumption. 2, such that q is moved out from the integral as a constant, but the constant is carefully chosen such that it still satisfies the property that it lies exactly on the ray emitting from $\mathbf{x}$.

Assumption 3. *Surface point is constant inside each reconstruction kernel but it depends upon the viewing ray.*

By denoting the carefully chosen constant for the i^{th} reconstruction kernel at screen-space coordinates $\mathbf{s}$ as $\mu_i(\mathbf{x})$, we can have the expected surface points formulated as:

$$\Psi(\mathbf{x}) \approx \sum_{i=1}^{N} \mu_i(\mathbf{x})\alpha_i(\mathbf{x}) \prod_{j=1}^{i-1} (1 - \alpha_j(\mathbf{x})) \quad (15)$$

Notice that even though μ_i is a function of $\mathbf{x}$, it actually represents the surface is constant *around* the current ray and filter, but it changes along with the current ray direction.

Since $\mu_i(\mathbf{x})$ is a property belonging to each Gaussian kernel, we look at one Gaussian kernel for defining it. Above all, it is obvious that we can expect two properties for it. As stated in the main paper, 1) The relative position between $\mu_i(\mathbf{x})$ and the center of Gaussian kernel should be fixed when the center and ray both translate by the same distance. 2) The relative position between $\mu_i(\mathbf{x})$ and the center of Gaussian kernel should be fixed when the Gaussian kernel and ray both rotate by the same angle. Therefore, we can put the Gaussian kernel in its canonical form, i.e., placed at the origin with identity rotation, for seeking the definition of $\mu_i(\mathbf{x})$.

For simplicity, we assume the Gaussian kernel is a sphere and then approximately extend it to the ellipsoid case. When the Gaussian kernel is not too big, rays emitted from the camera passing through it can be assumed as roughly passing through the center. Therefore, as the ray moves, the exact surface points form a shell

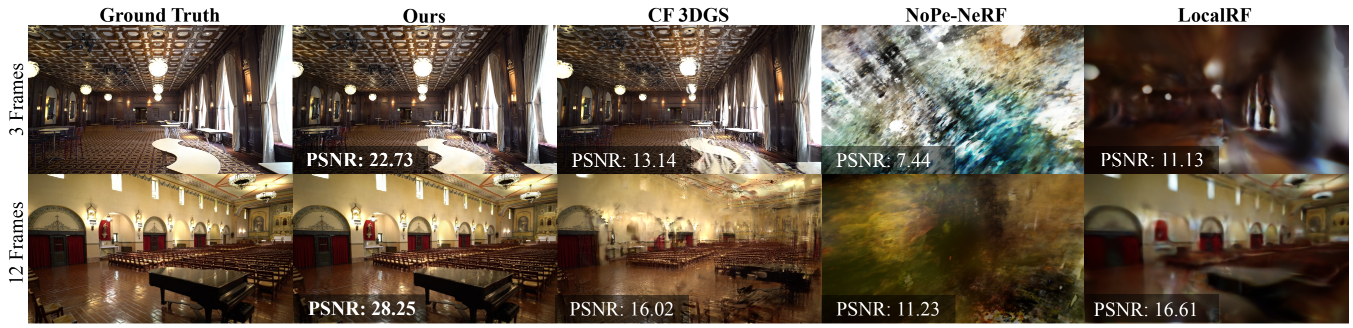

Figure 2: Qualitative comparison of pose-free methods for sparse view synthesis. From top to bottom, we use 3 and 12 frames as training views and others for testing. The scenes are, from the top to bottom: Ballroom, Church from Tanks&Temples. Our method enjoys the best visually pleasing results and highest PSNR scores. Images credit by Knapitsch et al. [2017].

Methods	3 Views			6 Views			12 Views		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Instant-NGP 📷	15.54	0.43	0.56	17.62	0.57	0.46	20.44	0.71	0.33
LocalRF	16.06	0.49	0.70	16.31	0.50	0.67	18.68	0.54	0.61
NoPe-NeRF	12.05	0.35	0.76	15.64	0.45	0.65	18.12	0.49	0.60
CF 3DGS	14.91	0.43	0.43	16.71	0.50	0.41	18.62	0.59	0.36
Ours	**20.37**	**0.66**	**0.26**	**25.18**	**0.81**	**0.16**	**28.65**	**0.88**	**0.10**

Table 1: Quantitative evaluation on the Tanks&Temples dataset in terms of novel view synthesis where SfM sees all frames for reconstruction, yielding clean camera poses. The best score is highlighted in bold. Methods marked with a camera denote that they rely on off-the-shelf estimated camera poses.

around the Gaussian kernel. In this case, it is sufficient to consider a 1D Gaussian kernel to determine the location of surface.

Specifically, assume the "scaling" of this 1D Gaussian kernel is s and a 1D ray is passing through it from infinite left to infinite right, the density can be given as $\sigma(x) = e^{-\frac{x^2}{2s^2}}$. Therefore, based on the corresponding defined free-flight distribution, the located surface is given by

$$\Psi = \int_{-\infty}^{\infty} x\sigma(x)e^{-\int_{-\infty}^{x}\sigma(s)ds}dx \tag{16}$$

We numerically solve this equation, but for efficient calculation, we propose to use $\Psi = 2s$ to approximate it. We then extend the conclusion to an ellipsoid Gaussian kernel by forming an ellipsoid shell whose length of each axis depends on the scaling of the same axes.

Therefore, in summary, $\mu_i(\mathbf{x})$ is reduced to a ray-ellipsoid intersection test. Each Gaussian kernel is associated with a unique ellipsoid, which shares the same center and rotation with the kernel, but the length of axes is a function of the scaling of the kernel.

In the forward pass, the calculated $\Psi(\mathbf{x})$ has the desirable property that it is promised to be projected to $\mathbf{x}$ in the screen space.

As to the backward pass, as discussed in the main paper, the gradients should not be propagated to the ray origin and direction even though the surface point is defined through the ray-ellipsoid intersection test, which has the form $\mu_i(\mathbf{x}) = \mathbf{o} + l\mathbf{d}$, where $\mathbf{o}, \mathbf{d}$ denote the ray origin and direction, and l denote the length between ray origin and intersected point. By defining the surface point through an approximated ellipsoid, we can actually parameterize the surface point as a function of center, rotation and scaling of Gaussian kernels. This awards us with another benefit.

Considering the gradients propagated to the reconstructed scene through the rendered surface $\frac{\partial \mathcal{L}}{\partial \mu_i(\mathbf{x})}$, if $\mu_i(\mathbf{x})$ is parameterized as $\mathbf{o} + l\mathbf{d}$, $\frac{\partial \mathcal{L}}{\partial l} = \frac{\partial \mathcal{L}}{\partial \mu_i(\mathbf{x})} \cdot \frac{\partial \mu_i(\mathbf{x})}{\partial l} = \frac{\partial \mathcal{L}}{\partial \mu_i(\mathbf{x})} \cdot \mathbf{d}$. Therefore, the gradient is biased towards the ray direction, and it is even possible that $\frac{\partial \mathcal{L}}{\partial \mu_i(\mathbf{x})}$ is perpendicular to $\mathbf{d}$, resulting in stuck at zero gradients. In contrast, by parameterizing $\mu_i(\mathbf{x})$ as $\mu_i + R_i f(s_i)$, where μ_i, R_i, s_i denote the center, rotation and scaling of i^{th} Gaussian kernel and f denotes the function of determining the shape of ellipsoid shell, $\frac{\partial \mathcal{L}}{\partial \mu_i} = \frac{\partial \mathcal{L}}{\partial \mu_i(\mathbf{x})}$. It is free of the problem mentioned above.

However, there is no denying that a density function represented as the composition of Gaussian kernels does not necessarily corresponds to a valid surface. It would be interesting to explore how to use splatting to represent SDF, UDF, and etc. function for more accurate surface rendering.

4.3 Numerically stable ray-ellipsoid intersection algorithm summary

In our introduced approximate surface rendering algorithm, the ray-ellipsoid intersection has the numerical issues when the ray origin is far from the ellipsoid. We summarize the details of a more numerically stable version in Alg. 2.

Algorithm 2 Numerically Stable Ray-ellipsoid Intersection Test

```
Input: Ray origin $\mathbf{o} \in \mathbb{R}^3$ and ray direction $\mathbf{d} \in \mathbb{R}^3$. The properties
    of an ellipsoid: center $\mathbf{e}_c \in \mathbb{R}^3$, scaling $\mathbf{e}_s = (\mathbf{e}_{s_x}, \mathbf{e}_{s_y}, \mathbf{e}_{s_z})^T \in$
    $\mathbb{R}^3_+$ and rotation $\mathbf{e}_r \in \mathfrak{so}(3)$.
Output: Whether the ray intersect the ellipsoid. If yes, return the
    world space coordinates of the intersection point.
1:  $\mathbf{o}' \leftarrow \mathbf{e}_r^T(\mathbf{o} - \mathbf{e}_c)$
2:  $\mathbf{d}' \leftarrow \frac{\mathbf{e}_r^T \mathbf{d}}{||\mathbf{e}_r^T \mathbf{d}||_2}$
3:  $\mathbf{t}_0 \leftarrow \mathbf{d}' \oslash \mathbf{e}_s$
4:  $\mathbf{t}_1 \leftarrow \mathbf{o}' \oslash \mathbf{e}_s$
5:  $\mathbf{t}_2 \leftarrow \frac{1}{\mathbf{e}_{s_x}\mathbf{e}_{s_y}\mathbf{e}_{s_z}}(\mathbf{d}' \times \mathbf{o}') \odot \mathbf{e}_s$
6:  if $||\mathbf{t}_0||_2 < ||\mathbf{t}_2||_2$ then
7:      return No Intersection, $\varnothing$.
8:  else
9:      $t \leftarrow -(\frac{\mathbf{t}_0}{||\mathbf{t}_0||_2} \cdot \frac{\mathbf{t}_1}{||\mathbf{t}_0||_2}) - \frac{1}{||\mathbf{t}_0||_2}\sqrt{1 - \frac{||\mathbf{t}_2||_2}{||\mathbf{t}_0||_2}}\sqrt{1 + \frac{||\mathbf{t}_2||_2}{||\mathbf{t}_0||_2}}$.
10:     return Intersected, $\mathbf{o} + t\mathbf{d}$.
```

Methods	**3**	**6**	**12**
Instant-NGP 📷	15.31	17.52	20.21
3DGS 📷	15.21	20.17	23.60
GNT 📷	<u>17.80</u>	<u>22.52</u>	<u>24.56</u>
LocalRF	16.06	16.31	18.68
NoPe-NeRF	12.05	15.64	18.12
CF 3DGS	14.91	16.71	18.62
Ours + MiDaS [2023]	*19.22*	*22.65*	*25.17*
Ours + MariGold [2023]	**20.37**	**25.18**	**28.65**

Table 2: PSNR Score ↑ on testing views with different number of training views (3, 6, 12) on the Tanks&Temples dataset, including our method using different depth estimators as backbones. We highlight the best score in bold, italicize the second best score, and underline the third best score.

Methods	CF 3DGS	NoPe-NeRF	Ours
RPE$_t$ ↓	7.1888	5.5087	**0.0000**
RPE$_r$ ↓	2.4940	2.3741	**0.0882**
ATE ↓	0.0981	0.0610	**0.0000**

Table 3: Quantitative evaluation of pose accuracy on the Tanks&Temples dataset with 12 training views. The unit of RPE$_r$ is in degrees, and RPE$_t$ is scaled by 100. ATE is in the ground truth scale. The best score is highlighted in bold.

5 ADDITIONAL EVALUATION RESULTS

5.1 Evaluation on Pose Accuracy

Pose accuracy measurement is non-trivial for sparse views. The reason is that even though ground-truth is given, the poses are relative, therefore, we have to estimate a transformation between produced camera poses from some pose-free methods and the ground-truth. However, it is very ambiguous in the case of sparse views. For example, given 3 views which are distributed on the x-axis with equal interval, even though a pose-free method registers views as three almost equal poses, the estimated transformation can align the registered poses and ground-truth well. In this case, the pose-free method cannot synthesize high quality novel view results due to failure of registration but has very high pose accuracy.

Therefore, we choose to evaluate pose accuracy on the 12 training views case on the Tanks&Temples dataset. We follow the NoPe-NeRF [Bian et al. 2023] to use the Absolute Trajectory Error (ATE) and Relative Pose Error (RPE), which consists of relative rotation error (RPE$_r$) and relative translation error (RPE$_t$), to measure the estimated pose accuracy of training views. The ground truth camera poses are estimated using SfM, which sees all views (including both training and testing views) for accuracy.

As shown in Table. 3, our method outperforms other baselines in all three metrics.

5.2 Quantitative Evaluation with Different Depth Estimators

Following the evaluation scheme in the main paper, we evaluate sparse view synthesis on the Tanks&Temples dataset [Knapitsch et al. 2017] with different depth estimators. Specifically, besides the MariGold [Ke et al. 2023] which we use in the main paper, we also test another popular estimator, i.e., MiDaS [Birkl et al. 2023], which is assumed to be less detailed and accurate compared to MariGold.

As shown in Table 2, our method with MiDaS achieves the second best results, outperforming all other baselines except for our method with MariGold, which demonstrates the robustness of our method. FSGS [Zhu et al. 2023] is excluded here as it relies on multi-view stereo estimation which fails on 50% of all tested cases.

5.3 Quantitative Evaluation with Clean Cameras

Following the evaluation scheme in the main paper, we evaluate sparse view synthesis on the Tanks&Temples dataset [Knapitsch et al. 2017]. We compare with pose-free methods: COLMAP-Free 3DGS (denoted as "CF 3DGS") [Fu et al. 2023], NoPe-NeRF [Bian et al. 2023], LocalRF [Meuleman et al. 2023]; and an efficient pose-required reconstruction method: Instant-NGP [Müller et al. 2022]. We test methods with clean cameras, where SfM sees all views for estimating camera poses. However, in this case, due to the stronger reliance between 3DGS, FSGS and SfM, we choose not to test 3DGS and FSGS for fairness. GNT fails in the case of clean cameras where SfM sees all the views, because SfM typically estimates poses in a much larger scale compared to the noisy camera case where SfM only sees a few views, and GNT cannot handle camera poses in a large scale.

We report the averaged results of all scenes in Table 1 for Tanks & Temples dataset. It is clear that even by using clean camera poses, the pose-required method still cannot outperform ours in all cases.

5.4 Qualitative Comparison

We show additional qualitative results on the Tanks&Temples dataset in Fig. 2. We compare our methods to other pose-free methods: COLMAP-Free 3DGS (denoted as "CF 3DGS") [Fu et al. 2023], NoPe-NeRF [Bian et al. 2023], and LocalRF [Meuleman et al. 2023]. Clearly, our method synthesizes the best results both qualitatively and quantitatively.

We show qualitative results on the Static Hikes dataset in Fig. 3, and compare our method with LocalRF [Meuleman et al. 2023] for illustration. Notice that LocalRF keeps updating the camera poses of testing views during the training, leading to occasionally more aligned results at the cost of longer optimization. Scenes in the Static Hikes dataset contain many high-frequency details, such as leaves, and the camera movements are not smooth. Therefore, in the case of sparse views, the production of testing views can be ambiguous as in the forest and garden case. For example, in the case of 3 and 12 views, regions emphasized by arrows synthesized by our method are reasonable but do not necessarily correspond to the ground-truth images. In this case, blurry results in LocalRF are even preferred for better metrics. Besides, scenes in the Static Hikes dataset are large, and sparse views cannot necessarily cover all regions. In the case of 3 views, for the university2 and university3 scenes, regions emphasized by arrows synthesized by our method are missing because they are not covered. In contrast, LocalRF can fill up the missing regions, which even though are inaccurate, resulting in better metrics. We show PSNR scores and it can be seen that even though LocalRF synthesizes much more blurry results, synthesized results of our method do not always enjoy higher PSNR scores due to the above mentioned challenges. However, the metrics improve as the number of views increases and we can achieve better metrics than other pose-free methods. Our method can also synthesize visually pleasing results in all cases.

REFERENCES

Wenjing Bian, Zirui Wang, Kejie Li, Jiawang Bian, and Victor Adrian Prisacariu. 2023. NoPe-NeRF: Optimising Neural Radiance Field with No Pose Prior. *CVPR*.

Reiner Birkl, Diana Wofk, and Matthias Müller. 2023. MiDaS v3.1 – A Model Zoo for Robust Monocular Relative Depth Estimation. *arXiv preprint arXiv:2307.14460* (2023).

Yang Fu, Sifei Liu, Amey Kulkarni, Jan Kautz, Alexei A Efros, and Xiaolong Wang. 2023. COLMAP-Free 3D Gaussian Splatting. *arXiv preprint arXiv:2312.07504* (2023).

Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. 2023. Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation. arXiv:2312.02145 [cs.CV]

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023).

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM Transactions on Graphics* 36, 4 (2017).

Andreas Meuleman, Yu-Lun Liu, Chen Gao, Jia-Bin Huang, Changil Kim, Min H. Kim, and Johannes Kopf. 2023. Progressively Optimized Local Radiance Fields for Robust View Synthesis. In *CVPR*.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2022. NeRF: representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2022), 99–106.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.* 41, 4 (2022), 102:1–102:15.

Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*.

Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. 2021. LoFTR: Detector-Free Local Feature Matching with Transformers. *CVPR* (2021).

Shitao Tang, Jiahui Zhang, Siyu Zhu, and Ping Tan. 2022. QuadTree Attention for Vision Transformers. *ICLR* (2022).

Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. 2023. Convolutions Die Hard: Open-Vocabulary Segmentation with Single Frozen Convolutional CLIP. In *NeurIPS*.

Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. 2023. FSGS: Real-Time Few-Shot View Synthesis using Gaussian Splatting. arXiv:2312.00451 [cs.CV]

Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. 2001. Surface Splatting. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 371–378.

M. Zwicker, H. Pfister, J. van Baar, and M. Gross. 2002. EWA Splatting. *IEEE Transactions on Visualization and Computer Graphics* 8, 3 (07/2002-09/2002 2002), 223–238.

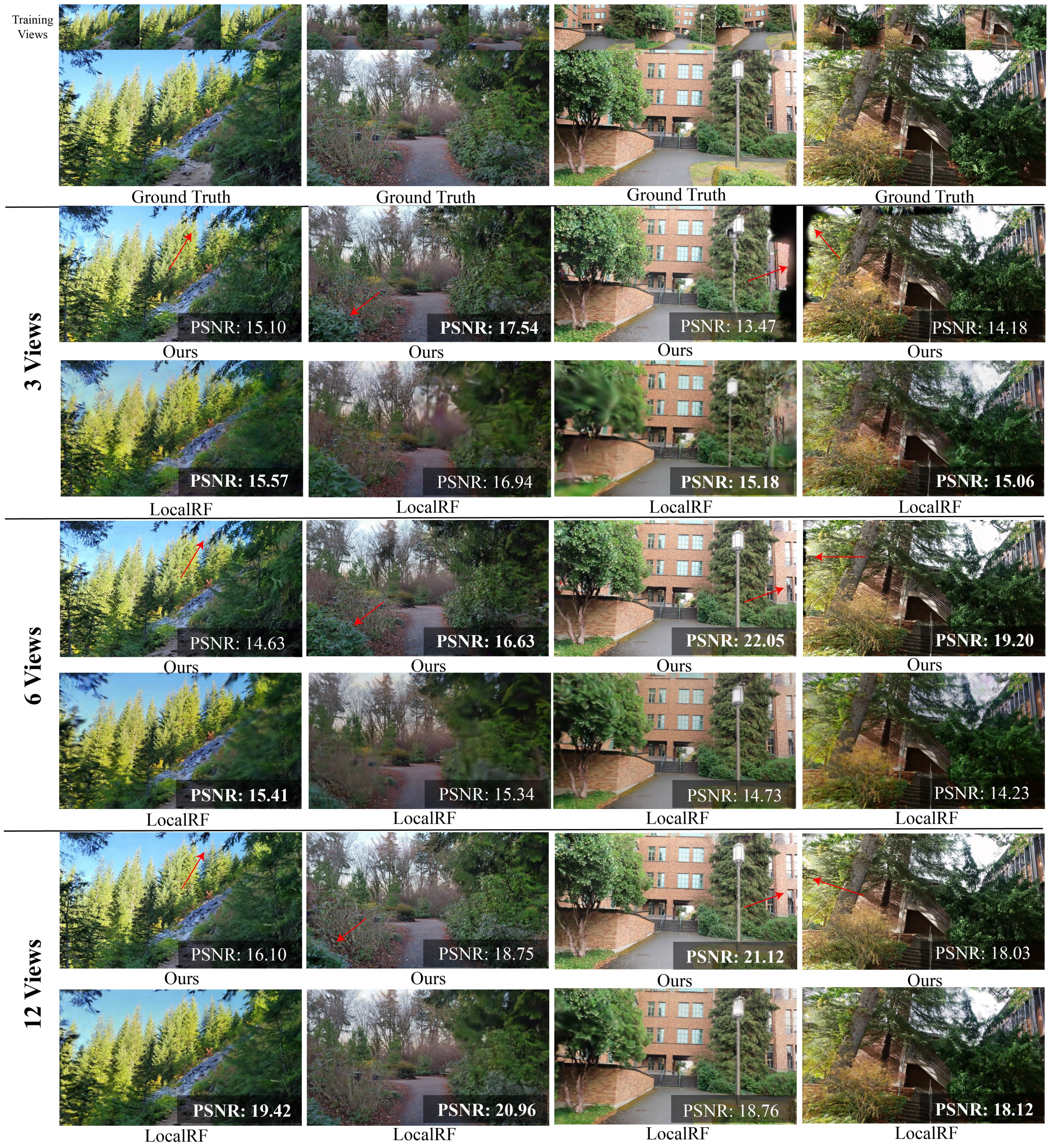

Figure 3: Illustration of results of a testing view on the Static Hikes [Meuleman et al. 2023]. The scenes are, from the left to right: forest, garden, university2, university3. We show synthesized results with 3, 6, 12 frames of our method and LocalRF. We also show the training frames in the 3 frames case at the first row. Regions of interest are emphasized by arrows. Please zoom in for better details. Images credit by Meuleman et al. [2023].