# A Construct-Optimize Approach to Sparse View Synthesis without Camera Pose

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


**Figure 1: We introduce a sparse view synthesis method, which does not rely on off-the-shelf estimated camera poses. Given the "Family" scene in the Tanks & Temples dataset, we use 6 out of 200 frames as training views and others for testing. Compared with other pose-free methods, including COLMAP-Free 3DGS [Fu et al. 2023] (CF 3DGS) and NoPe-NeRF [Bian et al. 2023], we achieve significant improvements in novel view synthesis both qualitatively and quantitatively. Besides, we also outperform methods which rely on off-the-shelf estimated camera poses, including Instant-NGP [Müller et al. 2022], Gaussian Splatting [Kerbl et al. 2023] (3DGS), and FSGS [Zhu et al. 2023]. Methods marked with a camera rely on off-the-shelf estimated camera poses throughout the paper. The inscription under the statue is emphasized to compare high-frequency details. Image credits by Knapitsch et al. [2017].**

## ABSTRACT

Novel view synthesis from a sparse set of input images is a challenging problem of great practical interest, especially when camera poses are absent or inaccurate. Direct optimization of camera poses and usage of estimated depths in neural radiance field algorithms usually do not produce good results because of the coupling between poses and depths, and inaccuracies in monocular depth estimation. In this paper, we leverage the recent 3D Gaussian splatting method to develop a novel construct-and-optimize method for sparse view synthesis without camera poses. Specifically, we *construct* a solution progressively by using monocular depth and projecting pixels back into the 3D world. During construction, we *optimize* the solution by detecting 2D correspondences between training views and the corresponding rendered images. We develop a unified differentiable pipeline for camera registration and adjustment of both camera poses and depths, followed by back-projection. We also introduce a novel notion of an expected surface in Gaussian splatting, which is critical to our optimization. These steps enable a coarse solution, which can then be low-pass filtered and refined using standard optimization methods. We demonstrate results on the Tanks and Temples and Static Hikes datasets with as few as three widely-spaced views, showing significantly better quality than competing methods, including those with approximate camera pose information. Moreover, our results improve with more views and outperform previous InstantNGP and Gaussian Splatting algorithms even when using half the dataset.



*SIGGRAPH Conference Papers '24, July 27–August 01, 2024, Denver, CO, USA*

ACM ISBN 979-8-4007-0525-0/24/07
https://doi.org/10.1145/3641519.3657427

## CCS CONCEPTS

• **Computing methodologies → Rendering**.

## KEYWORDS

view synthesis, 3D gaussians, camera optimization

**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. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27–August 01, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641519.3657427

## 1 INTRODUCTION

Neural Radiance Field (NeRF) and its several variants [Barron et al. 2023; Kerbl et al. 2023; Mildenhall et al. 2020; Müller et al. 2022] have excelled in novel view synthesis of 3D scenes. However, these methods require densely captured views with accurately labeled camera poses, which is often not feasible in practical scenarios. Often, camera poses are obtained from Structure-from-Motion (SfM) methods like COLMAP [Schönberger and Frahm 2016; Schönberger et al. 2016] as a pre-processing step to NeRF, which is brittle and fails when given sparse views. Even in the cases where COLMAP can successfully register sparse scenes, as shown in Fig. 1, sparse view synthesis is challenging and ill-posed from ambiguity in the 3D scene due to under-sampling. This limitation raises a critical question: Is it possible to perform novel view synthesis from sparse input captures (as little as 3-6 images) with unknown camera poses?

The recent introduction of 3D Gaussian Splatting, denoted as 3DGS [Kerbl et al. 2023], also struggles to deal with sparse view synthesis due to too sparse initialization from SfM. However, the explicit representation, i.e., 3D Gaussians, of 3DGS provides new opportunities to solve that critical question. Different from fitting a solution for sparse view synthesis in NeRFs, we desire to *construct* a solution based on a dense prior, i.e., estimated monocular depth; however *optimization* is still essential, and we therefore call our approach a construct-and-optimize method.

A naive way to construct a solution is by first estimating camera poses and then back-projecting pixels into the scene based on their estimated depths. However, there is a problem in handling camera poses and depth estimation independently – in actual 3D captures, both quantities are tightly coupled and depend upon one another [Kopf et al. 2021] as shown in Fig. 2. Unfortunately, in the case of sparse views, monocular depth estimation algorithms do not take camera pose information into account. Additionally, camera pose estimation algorithms do not leverage and align monocular depth. As a result, back-projection for the same scene across multiple views may be inconsistent. We therefore involve optimization in the construction to solve these issues. Our pipeline shown in Fig. 3 is progressive, i.e. it builds the scene continuously. For the next unregistered view, we first estimate its camera pose in a *registration* stage. Afterwards, we adjust the previous registered camera poses and align monocular depths, which we call *adjustment*. At last, pixels of the next view are back-projected into world space as 3D Gaussians. Therefore, the camera poses are not needed to be known in advance. Finally, we reach a coarse solution, which is then refined using standard optimization [Kerbl et al. 2023] to reproduce details faithfully. Before that, we also apply a low-pass filtering to avoid high-frequency artifacts as shown in Fig. 10 (a).

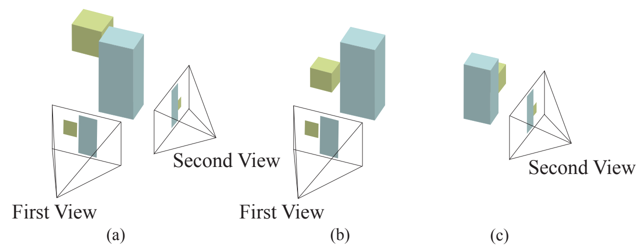


**Figure 2: Example of ambiguity given partial views. Given the scene in (a), there could be different possibilities of scene layouts as shown in (b) and (c), if only the first view or second view is observed. (b) or (c) could be the estimated depth. This ambiguity results in unavoidable error in monocular depth estimation, which necessitates the alignment between camera poses and estimated depths.**

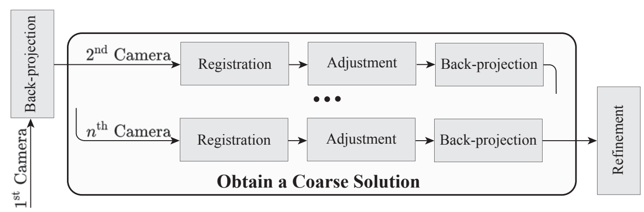


**Figure 3: Overview of our method for sparse view synthesis. We first back-project the first view and sequentially register, adjust and back-project the remaining views in sequence to obtain a coarse solution. This coarse solution is then refined by standard optimization to reproduce fine details.**

To harmonize the monocular depths with the camera poses, we unify *registration* and *adjustment* in a differentiable pipeline for optimization, whose objective is to reproduce training views. However, for sparse views with non-trivial camera movements, commonly used pixel-wise supervision does not lead to effective optimization, since it only considers short-range information. To take long-range information into account, we instead detect 2D correspondences [Sun et al. 2021; Tang et al. 2022] between training views and their corresponding rendered views, and aim to match them in screen-space coordinates as shown in Fig. 4 (a), (b). To optimize over the detected correspondences, we have to render screen-space coordinates of expected surface points, which requires a definition of an expected surface in Gaussian splatting. Existing attempts at estimating the expected surface (e.g., [Chung et al. 2023; Keetha et al. 2023; Zhu et al. 2023]) do not fully respect the shape of Gaussian kernels, leading to ineffective optimization for our problem. We are therefore motivated to develop a more accurate rendering of the expected surface for 3DGS.

Through extensive experiments and comparisons, we show that our method achieves state-of-the-art results when dealing with challenging cases where only a few views (< 5% of total views) are provided and there is non-trivial camera movement between any pair of views, as shown in Fig. 1, 8. The performance of our method also improves as the number of views increases as shown in Fig. 9.

In summary, the contributions of our work include:

- We propose a unified differentiable pipeline, which leverages correspondences as supervision, for sparse view synthesis without camera poses in Sec. 3.1, 3.2.
- We propose a differentiable approximation of the expected surface in Gaussian splatting for effective correspondence supervision in Sec. 3.3.

## 2 RELATED WORK

*Sparse view synthesis.* The computer vision and graphics community has studied novel view synthesis for decades [Chai et al. 2000; Chen and Williams 1993; Gortler et al. 1996; Levoy and Hanrahan 1996; McMillan and Bishop 1995]. A number of subsequent advances were made in sparse view synthesis [Hedman and Kopf 2018; Kalantari et al. 2016; Mildenhall et al. 2019]. We build on the recent development of neural radiance fields for view synthesis [Mildenhall et al. 2020, 2022].

Sparse view synthesis in NeRFs typically assumes camera poses to be known to simplify the problem. Some works (e.g., [Deng et al. 2022; Kim et al. 2022; Niemeyer et al. 2022; Wang et al. 2023; Yu et al. 2021, 2022]) try to fit an efficient representation (e.g., MLPs [Mildenhall et al. 2020], hash tables [Müller et al. 2022], etc.) with prior knowledge, such as depth or heuristic constraints, to reduce the ambiguity. However, the reconstructed continuous signal still unavoidably results in blurry or noisy images for novel views.

In this paper, we show that constructing a solution for sparse view synthesis with optimization can be easier, in contrast to fitting the complex signal from sparse observations. However, the camera poses and the reconstructed scene should be aligned. In SfM, this goal is achieved by *bundle adjustment* [Snavely et al. 2008]. However, our representation for the scene as 3D Gaussian is different from points. Realistic rendering and the higher degrees of freedom afforded by 3D Gaussians enable correspondence detection between the current reconstructed scene and training views for alignment. This facilitates effective and stable optimization based on differentiable rendering. Therefore, we call this optimization process *adjustment* to distinguish it from conventional bundle adjustment. Furthermore, *registration* of camera poses can also be unified in the same optimization framework. The camera poses are therefore not needed to be known in advance.

*Optimizing camera poses in NeRFs.* Accurate camera poses are vital for realistic view synthesis in NeRFs. Given initially estimated camera poses, which usually come from SfM [Schönberger and Frahm 2016; Schönberger et al. 2016] or sensors, some methods (e.g., [Heo et al. 2023; Lin et al. 2021; Park et al. 2023; Wu et al. 2023]) refine them during the optimization for better view synthesis.

When camera poses are not given, several works [Bian et al. 2023; Fu et al. 2023; Lin et al. 2021; Meuleman et al. 2023; Wang et al. 2021] require a dense capture and gradually register frames by pixel-wise supervision and/or additional priors, such as depth [Birkl et al. 2023; Ranftl et al. 2022] or optical-flow [Teed and Deng 2020]. It is noteworthy that the camera pose of the next unregistered frame is initialized as the camera pose of the last registered frame. However, when the views are sparse and there is non-trivial camera movement between any pair of captured frames, SfM sometimes fails to produce accurate results and none of the previous methods can deal with this scenario well. GNeRF [Meng et al. 2021] tries to tackle the general camera pose querying problem by a generative prior, which is still limited to individual objects rather than scenes.

As discussed in Xing et al. [2022], pixel-wise supervision yields gradients that only consider short-range information. In the case of sparse views, there could be non-trivial camera movements that makes it desirable to have gradients that consider long-range information. Therefore, in this work, we leverage 2D correspondences between the reconstructed scene and training views for effective optimization of camera poses and the reconstructed scene.

*Surface rendering in Gaussian splatting.* In 3DGS, recent progress (e.g., [Chung et al. 2023; Fu et al. 2023; Keetha et al. 2023; Xiong et al. 2023; Yan et al. 2023; Zhu et al. 2023]) has found that an approximate surface is useful for view synthesis. However, their approximation unavoidably assumes that the surface is a constant point inside each Gaussian kernel, which is sub-optimal in our case. Therefore, we propose an approximate anisotropic surface rendering scheme that is more accurate than prior works and results in effective optimization for our task. A variant of our approximate surface rendering is introduced in [Lassner and Zollhöfer 2021], but it is isotropic and does not deal with volume rendering. SuGaR [Guédon and Lepetit 2023] proposes an ideal depth to regularize the current rendered depth. In contrast, we better approximate the expected surface for downstream optimization.

## 3 METHOD

In Sec. 3.1, we first present an overview of our algorithm for sparse view synthesis. In Sec. 3.2, we introduce our differentiable pipeline for registration and adjustment. Next, we introduce a more accurate approximation of surface rendering for 3D Gaussians in Sec. 3.3, which allows us to leverage correspondences as an effective supervision in the differentiable pipeline. After these steps, we have a coarse solution, which we further refine in Sec. 3.4. We present an outline of our pipeline in Fig. 3, and focus on registering, adjusting and back-projecting the $k+1^{\text{th}}$ view in Fig. 4.

### 3.1 Algorithm Overview

Assuming we have an ordered set of consecutively captured $n$ RGB images $\mathcal{I} = \{I_1, I_2, ..., I_n\}$ and their corresponding estimated monocular depths $\mathcal{D} = \{D_1, D_2, ..., D_n\}$, we are interested in novel view synthesis without camera poses, using 3D Gaussians as our representation. As in [Bian et al. 2023], we assume the intrinsic matrix $K$ of the camera is given, and denote the unknown extrinsic matrices for each view as $\mathcal{P} = \{P_1, P_2, ..., P_n\}$.

As shown in Fig. 3, we start with the first view $I_1$ and set its extrinsic matrix $P_1$ to the identity matrix. Next, we back-project each of its pixels into world space as 3D Gaussians, such that the rendered image and depth match $I_1$ and $D_1$ respectively.

Specifically, given the camera pose and depth for the frame, we can construct a particular fully opaque splat for each pixel in our approximate surface rendering scheme (please find details in Sec. 1.1 of the supplementary).

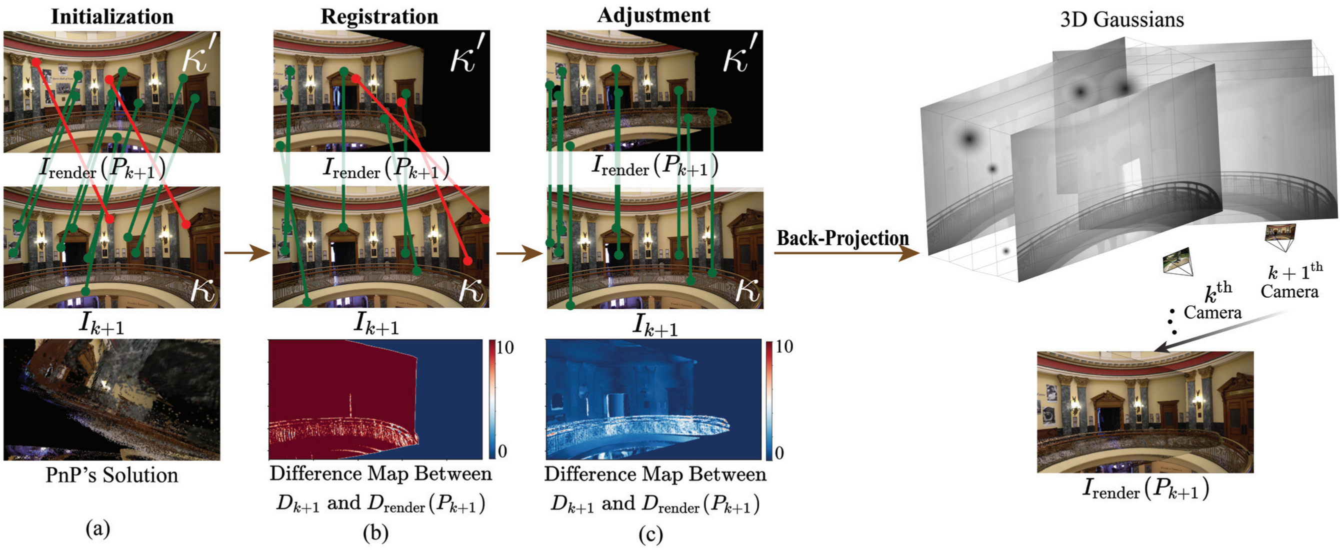


**Figure 4: We assume the first $k$ views have already been registered, and illustrate the registration, adjustment and back-projection of the $k+1^{\text{th}}$ view. (a) We first initialize the camera pose of the $k+1^{\text{th}}$ view, denoted as $P_{k+1}$, as the $k^{\text{th}}$ view's camera pose. 2D correspondences are detected between ground-truth image $I_{k+1}$ and the rendered result $I_{\text{render}}(P_{k+1})$ at $P_{k+1}$. Correspondence points on $I_{\text{render}}(P_{k+1})$ are denoted as $\kappa'$, while those on $I_{k+1}$ are denoted as $\kappa$. Green points denote correct correspondences, while red points denote wrong correspondences. We can use perspective-n-points (PnP) to solve the camera pose but it results in an erroneous solution. (b) We then apply our optimization pipeline (Sec. 3.2) to estimate the camera pose for registration. For now, the monocular depth $D_{k+1}$ of the $k+1^{\text{th}}$ view deviates significantly from the rendered depth $D_{\text{render}}(P_{k+1})$ at $P_{k+1}$. (c) Afterwards, we apply our optimization pipeline (Sec. 3.2) to adjust all previous registered camera poses and monocular depths along with $P_{k+1}$ and $D_{k+1}$. It can be seen that $I_{\text{render}}(P_{k+1})$ and $D_{k+1}$ are much close to $I_{k+1}$ and $D_{\text{render}}(P_{k+1})$. Finally, we back-project pixels in the $k+1^{\text{th}}$ view into world space as 3D Gaussians based on $D_{k+1}$. Images credit by Knapitsch et al. [2017].**

We then assume the first $k$ frames have already been registered and consider the next unregistered view $I_{k+1}$. Its extrinsic matrix $P_{k+1}$ is first initialized as $P_k$ as shown in Fig. 4 (a). We optimize $P_{k+1}$ based on the previous reconstruction to register the new view as shown in Fig. 4 (b). During adjustment, we optimize all previous extrinsic matrices $\{P_1, P_2, ..., P_k\}$ and monocular depths $\{D_1, D_2, ..., D_k\}$ along with $P_{k+1}$ and $D_{k+1}$ as shown in Fig. 4 (c). After that, pixels in $I_{k+1}$ are back-projected based on the aligned depth $D_{k+1}$. After processing all $n$ views, we reach a coarse solution for sparse view synthesis.

## 3.2 Optimization Framework

We achieve *registration* and *adjustment* through an optimization framework. The camera pose is optimized in both *registration* and *adjustment*, but the alignment of depth is achieved through the *adjustment* only. Our optimization aims to reproduce training views, i.e., for each view, the rendered image should match the ground-truth image. Pure pixel-wise supervision does not lead to effective optimization for non-trivial camera movement between consecutive views, since it only considers short-range information. Inspired by but different from optimal transport [Xing et al. 2022], we leverage correspondences, which bootstrap the method, to consider long-range information for optimization.

Assume the view of interest is $I$, and the rendered image given its current extrinsic matrix $P$, is $I_{\text{render}}(P)$. We can leverage off-the-shelf detectors [Sun et al. 2021; Tang et al. 2022] to detect the 2D correspondences between $I$ and $I_{\text{render}}(P)$ per optimization step. The 2D screen-space points on $I$ are $\kappa = \{\kappa^{(1)}, \kappa^{(2)}, ..., \kappa^{(M)}\}$, where $M$ denotes the number of points. The 2D screen-space points on $I_{\text{render}}(P)$ are $\kappa' = \{\kappa'^{(1)}, \kappa'^{(2)}, ..., \kappa'^{(M)}\}$. The optimization goal is then to match $\kappa$ with $\kappa'$, which is visualized in Fig. 4 (a) and (b). For registration only, we can use perspective-n-points (PnP) [Lepetit et al. 2009] to solve camera parameters. However, it is sensitive to mismatches as shown in Fig. 4 (a), and it is hard to balance the number of matches with the threshold of the RANSAC algorithm. On the other hand, we find that our optimization framework is robust and achieves more accurate registration. Therefore, registration and adjustment are unified under the same optimization framework.

In order to back-propagate gradients from the matching between $\kappa$ and $\kappa'$ to the 3D Gaussians that form the surface, we use a differentiable approximate surface renderer, elaborated in Sec. 3.3, to render screen-space coordinates at $\kappa'^{(i)}, i = 1, 2, ..., M$ as $q(c')$. The resulting loss is

$$\mathcal{L}_{\text{corr}} = \sum_{i=1}^{M} ||q(\kappa'^{(i)}) - \kappa^{(i)}||_1. \tag{1}$$

Importantly, $q(\kappa^{'(i)})$ equals $\kappa^{'(i)}$ numerically, but it forms a graph for back-propagating gradients to the underlying representation.

We also find that when $P$ is close to ground-truth, short-range information provided by pixel-wise supervision can help stabilize the optimization. This loss is given by

$$\mathcal{L}_{\text{rgb}} = ||I - I_{\text{render}}(P)||_1. \tag{2}$$

Finally, in the *adjustment*, we adjust the monocular depth $D$ of the current view during the adjustment phase, for later use during back-projection. To align the estimated monocular depths effectively, we would like to use the scale-consistency assumption [Birkl et al. 2023; Ke et al. 2023] but it does not always hold true. To relax the scale-consistency assumption, instead of learning an affine transformation *per view*, we learn a separate affine transformation *per primitive* [Yu et al. 2023]. By denoting the rendered depth given the current extrinsic matrix $P$ as $D_{\text{render}}(P)$, we match $D$ to the current rendered depth $D_{\text{render}}(P)$ for all correspondences. Specifically, the rendered depth at $\kappa^{'}$ is denoted as $b(\kappa^{'})$ and the monocular depth at $\kappa$ is denoted as $d(\kappa)$. The loss term is defined as

$$\mathcal{L}_{\text{depth}} = \sum_{i=1}^{M} ||\text{sg}[b(\kappa^{'(i)})] - d(\kappa^{(i)})||_1. \tag{3}$$

Since we only want to adjust the monocular depth $D$ to make it match the rendered depth $D_{\text{render}}(P)$, we stop the backward propagation of gradients (sg$[\cdot]$) through $b$.

In summary, the optimization objective is defined as

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{corr}} + \lambda_2 \mathcal{L}_{\text{rgb}} + \lambda_3 \mathcal{L}_{\text{depth}}, \tag{4}$$

where $\lambda_1 = 1000, \lambda_2 = 10, \lambda_3 = 1$. The gradients back-propagate to camera parameters and the reconstructed scene.

### 3.3 Differentiable Surface Rendering

Our goal for correspondence matching (see Sec. 3.2) is to propagate long-range gradient information from a 2D screen-space point to its corresponding 3D surface point. In essence, our goal is to map perturbations of the 2D screen-space point to corresponding perturbations of the 3D surface point. However, this raises an important question – where is the 3D surface point?

Since the 3D Gaussian representation is volumetric, there are no explicit surfaces; instead, previous works [Chung et al. 2023; Fu et al. 2023; Keetha et al. 2023; Yan et al. 2023] compute the depth of *expected* 3D surface point $D(\mathbf{s})$ corresponding to the 2D screen-space point $\mathbf{s}$ as:

$$D(\mathbf{s}) = \sum_i d_i \alpha_i(\mathbf{s}) \prod_{j=1}^{i-1} (1 - \alpha_j(\mathbf{s})), \tag{5}$$

where $d_i \in \mathbb{R}$ denotes the $z$-axis coordinate for the transformed Gaussian centers in the camera space, and $\alpha_i \in \mathbb{R}$ and $\alpha_j \in \mathbb{R}$ denote the calculated alpha-blending coefficient of the $i^{\text{th}}$ and $j^{\text{th}}$ Gaussian kernel. This is shown as Fig. 5 (a).

Its extension to corresponding *expected* 3D surface point $\Psi(\mathbf{s})$ at $\mathbf{s}$ is given by

$$\Psi(\mathbf{s}) = \sum_i \boldsymbol{\mu}_i \alpha_i(\mathbf{s}) \prod_{j=1}^{i-1} (1 - \alpha_j(\mathbf{s})), \tag{6}$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^3$ denotes the center of the $i^{\text{th}}$ Gaussian kernel.

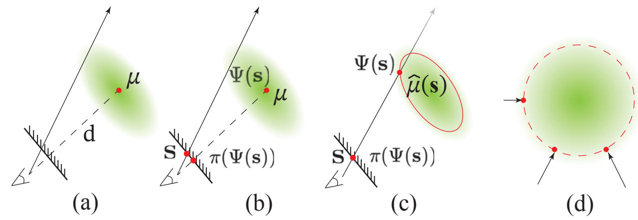


**Figure 5: Illustration of surface rendering in Gaussian splatting. Assume the ray is shot from screen-space coordinates s and $\Psi(\mathbf{s})$ denotes the rendered surface point. $\pi(\cdot)$ denotes projecting 3D points into screen space. (a) Depth rendering of previous methods. The depth $d$ of a Gaussian kernel is defined as the $z$-axis coordinate for the transformed center $\mu$ in the camera space. (b) Extending (a) to render the exact 3D surface point. The surface point of the Gaussian kernel is defined as the center $\boldsymbol{\mu}$. It could result in a mismatch between s and $\pi(\Psi(\mathbf{s}))$. (c) Approximate surface rendering of our method. The surface point $\widehat{\mu}(\mathbf{s})$ of the Gaussian kernel is defined as the intersection point between the ray and an ellipsoid shell. Therefore, our method guarantees a match between s and $\pi(\Psi(\mathbf{s}))$. (d) Surface rendering of our method when considering all the rays passing through the center of a spherical Gaussian kernel. The expected surface points form a shell.**

Unfortunately, their rendering model for $\Psi(\mathbf{s})$ is not consistent with $\mathbf{s}$, i.e., $\Psi(\mathbf{s})$ may not be projected onto $s$, which is essential to our optimization, see Fig. 5 (b).

We would like to fix this rendering model, without breaking any of the original assumptions [Kerbl et al. 2023; Zwicker et al. 2001, 2002] in order to have efficient rendering. Our solution is to replace $\boldsymbol{\mu}_i$ with a better approximation $\widehat{\boldsymbol{\mu}}_i(\mathbf{s})$ for the expected surface point. Different from the earlier rendering model, $\widehat{\boldsymbol{\mu}}_i(\mathbf{s})$ is now dependent on $\mathbf{s}$, as illustrated in Fig. 5 (c).

We expect the relative position $\delta$ between $\widehat{\boldsymbol{\mu}}_i(\mathbf{s})$ and $\boldsymbol{\mu}_i$ to be translation- and rotation-invariant, as shown in Fig. 6 (a). We can therefore consider $\widehat{\boldsymbol{\mu}}_i(\mathbf{s})$ in the canonical form, i.e., the Gaussian kernel is placed canonically at the origin. For a single isotropic 3D Gaussian, we can compute the expected surface point using its free flight distance probability density function. The locus of expected surface points for all rays passing through the center of the 3D Gaussian form a shell, as shown in Fig. 5 (d). We use this shell to approximate the surface of the 3D Gaussian. Similarly, for an anisotropic 3D Gaussian, we use an ellipsoidal shell to approximate the surface. Here, the semi-axes can be computed by an integral which only depends on the anisotropic Gaussian's scale parameters. For efficient gradient calculation, we approximate this integral with a linear function. Our rendering guarantees the "consistency" property, i.e., $\Psi(\mathbf{s})$ is projected to $\mathbf{s}$, as shown in Fig. 5 (c).

In summary, $\widehat{\boldsymbol{\mu}}_i(\mathbf{s})$ is reduced to a ray-intersection test between the current ray and a defined ellipsoid shell of the current Gaussian kernel, illustrated in Fig. 5 (c). Specifically, assume the origin and corresponding direction of the current ray are $\mathbf{o}$ and $\mathbf{d}(\mathbf{s})$, and the intersection distance is $l$, so that $\widehat{\boldsymbol{\mu}}_i(\mathbf{s}) = \mathbf{o} + l\mathbf{d}(\mathbf{s})$. And we skip Gaussian kernels whose approximate surfaces do not intersect with

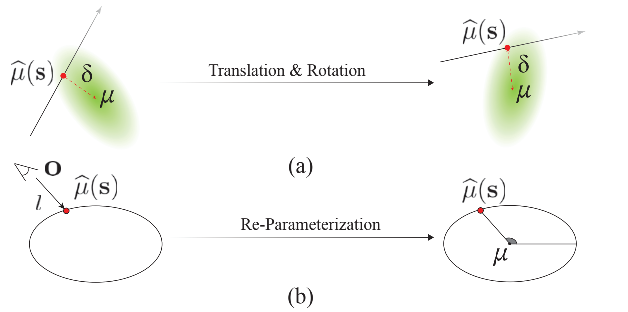


Figure 6: (a) Illustration of the invariance of relative position $\delta$ between the surface point $\widehat{\mu}(\mathbf{s})$, and the center of the Gaussian kernel $\boldsymbol{\mu}$. $\delta$ is expected to be translation- and rotation-invariant. (b) Illustration of re-parameterizing surface point $\widehat{\mu}(\mathbf{s})$ from an intersected point into a point defined on an ellipsoid shell.

the current ray, denoted as $\widehat{\boldsymbol{\mu}}_i(\mathbf{s}) = \emptyset$. The expected world space coordinates of the surface point at $\mathbf{s}$ are therefore given by

$$\Psi(\mathbf{s}) = \sum_{i, \widehat{\boldsymbol{\mu}}_i(\mathbf{s}) \neq \emptyset} \widehat{\boldsymbol{\mu}}_i(\mathbf{s}) \alpha_i(\mathbf{s}) \prod_{j=1, \widehat{\boldsymbol{\mu}}_j(\mathbf{s}) \neq \emptyset}^{i-1} (1 - \alpha_j(\mathbf{s})). \tag{7}$$

By projecting $\widehat{\boldsymbol{\mu}}_i(\mathbf{s})$ into camera space to obtain the $z$-axis coordinate as $z_i(\mathbf{s})$ and into screen space to obtain the screen-space coordinates $\pi(\widehat{\boldsymbol{\mu}}_i(\mathbf{s}))$, we are able to render depth $D(\mathbf{s})$ and screen-space coordinates $q(\mathbf{s})$ of expected surface points, which are given by

$$\begin{aligned} D(\mathbf{s}) &= \sum_{i, \widehat{\boldsymbol{\mu}}_i(\mathbf{s}) \neq \emptyset} z_i(\mathbf{s}) \alpha_i(\mathbf{s}) \prod_{j=1, \widehat{\boldsymbol{\mu}}_j(\mathbf{s}) \neq \emptyset}^{i-1} (1 - \alpha_j(\mathbf{s})) \\ q(\mathbf{s}) &= \sum_{i, \widehat{\boldsymbol{\mu}}_i(\mathbf{s}) \neq \emptyset} \pi(\widehat{\boldsymbol{\mu}}_i(\mathbf{s})) \alpha_i(\mathbf{s}) \prod_{j=1, \widehat{\boldsymbol{\mu}}_j(\mathbf{s}) \neq \emptyset}^{i-1} (1 - \alpha_j(\mathbf{s})). \end{aligned} \tag{8}$$

We follow the same framework of forward and backward passes in the rasterizer proposed in [Kerbl et al. 2023], but replace the rendering of color with our defined expected surface. However, the backward pass is a bit different due to the involved ray-intersection test. Specifically, considering the surface point $\widehat{\boldsymbol{\mu}}_i(\mathbf{s})$ for the $i^{\text{th}}$ Gaussian kernel, it is defined as $\mathbf{o} + l\mathbf{d}(\mathbf{s})$, in which $l$ is a function of the center, rotation, and scaling of the $i^{\text{th}}$ Gaussian kernel. This parameterization could result in gradient cancellation when camera parameters are also optimized, illustrated in Fig. 7. Even though the surface point $\widehat{\boldsymbol{\mu}}_i(\mathbf{s})$ is defined through the origin and direction of the current ray, it should be treated as an independent point existing on an ellipsoid shell. Therefore, we propose to re-parameterize $\widehat{\boldsymbol{\mu}}_i(\mathbf{s})$ as a function of the center, rotation and scaling of the ellipsoid shell solely as shown in Fig. 6 (b). The gradients are then back-propagated to these properties of the ellipsoid shell, and finally to the camera parameters and the Gaussian kernel.

## 3.4 Refinement and Implementation Details

After reaching a coarse solution using the algorithm above, we refine the solution using standard optimization techniques [Kerbl et al. 2023] and optimize the camera poses in the optimization. Before that, we first remove error-prone high-frequency reconstructions by applying a low-pass filter, see Fig. 10 (a). To achieve the same effect as a low-pass filter by penalizing dense clusters of Gaussians and promoting more uniformly distributed Gaussians, for each detected object in each view, we only retain 10% of the total back-projected Gaussians, by a farthest point sampling algorithm [Ravi et al. 2020] based on the center of the Gaussians.

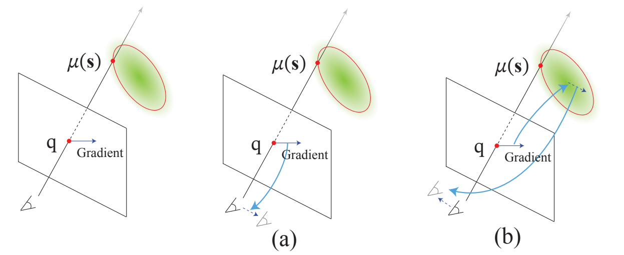


**Figure 7: Illustration of gradient cancellation due to the intersection test. Given the surface point $\mu(\mathbf{s})$ and its projected screen-space coordinates $q$, assume the gradients move $q$ to the right. (a) In one way, since $\mu(\mathbf{s})$ is defined through the ray origin and direction, the gradients are back-propagated to the ray origin to move the viewing position right. (b) In another way, since $\mu(\mathbf{s})$ is defined through the center of the Gaussian kernel, the gradients are back-propagated to the center to move it to the right. In turn, due to the transformation from world space to camera space, the gradients are back-propagated to the viewing position to move it left. Therefore, these two gradients cancel each other and sum to zero occasionally.**

The optimization setup is almost the same as Kerbl et al. [2023]. On an NVIDIA RTX 3080 GPU, the time required to register and adjust a view varies but typically takes minutes and increases as the number of views increases. The refinement with standard 3DGS takes $\sim$ 1 hour. During inference, we enjoy the same speed of [Kerbl et al. 2023] since we still leverage 3D Gaussians as our representation. Please find more details in the supplementary.

# 4 RESULTS

We provide evaluation details below, and conduct comparisons to other methods (Sec. 4.1), and an ablation study of the components of our algorithm (Sec. 4.2). We also encourage readers to look at the supplementary and accompanying video for more results.

*Evaluation Details.* To compare with methods which require initialized camera poses, we use SfM [Schönberger and Frahm 2016; Schönberger et al. 2016] for registration. For fairness, SfM only sees training views for reconstruction, and registers testing views after reconstruction, in which bundle adjustment is not included.

*Datasets.* We evaluate on datasets which contain non-trivial camera movements but ensure that there is overlap between consecutive frames, as has been done by other pose-free methods [Bian et al. 2023; Fu et al. 2023; Meuleman et al. 2023] for evaluation. Following Bian et al. [2023], we use 8 scenes of the **Tanks&Temples** dataset

**Table 1: Quantitative evaluation on the Tanks&Temples dataset. The best score is highlighted in bold throughout the paper. †: FSGS requires multi-view stereo estimation from COLMAP, which fails on 50% of total scenes. We therefore report the averaged metrics of the remaining scenes.**

| Methods | 3 Views | | | 6 Views | | | 12 Views | | |
|---|---|---|---|---|---|---|---|---|---|
| | PSNR ↑ | SSIM ↑ | LPIPS ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
| Instant-NGP 📷 | 15.31 | 0.42 | 0.56 | 17.52 | 0.56 | 0.47 | 20.21 | 0.69 | 0.35 |
| 3DGS 📷 | 15.21 | 0.46 | 0.43 | 20.17 | 0.71 | 0.24 | 23.60 | 0.81 | 0.17 |
| FSGS† 📷 | 19.23 | 0.58 | 0.37 | 23.55 | 0.74 | 0.28 | 26.81 | 0.83 | 0.22 |
| GNT 📷 | 17.80 | 0.57 | 0.29 | 22.52 | 0.77 | 0.18 | 24.56 | 0.82 | 0.14 |
| 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 2: Quantitative evaluation on the Static Hikes dataset.**

| Methods | 3 Views | | | 6 Views | | | 12 Views | | |
|---|---|---|---|---|---|---|---|---|---|
| | PSNR ↑ | SSIM ↑ | LPIPS ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
| LocalRF | 15.97 | 0.33 | 0.47 | 18.32 | 0.47 | 0.43 | **20.13** | **0.54** | 0.41 |
| NoPe-NeRF | 14.85 | 0.25 | 0.67 | 18.59 | 0.34 | 0.57 | 18.19 | 0.34 | 0.59 |
| CF 3DGS | 15.45 | 0.28 | 0.60 | 17.02 | 0.35 | 0.52 | 17.65 | 0.39 | 0.46 |
| Ours | **16.35** | **0.38** | **0.37** | **18.96** | **0.50** | **0.31** | 19.70 | 0.53 | **0.29** |

**Table 3: PSNR Score ↑ on testing views for investigating the effects of number of training views (first row).**

| Methods | 3 📷 | 4 📷 | 6 📷 | 12 📷 | 24 📷 | 60 📷 |
|---|---|---|---|---|---|---|
| Instant-NGP 📷 | 16.46 | 16.94 | 17.19 | 17.86 | 18.57 | 21.10 |
| 3DGS 📷 | 14.99 | 14.99 | 17.76 | 17.12 | 25.35 | 26.95 |
| Ours | **21.54** | **25.36** | **29.00** | **32.09** | **33.73** | **35.93** |

**Table 4: PSNR Score ↑ on testing views for ablation models.**

| Methods | Config-A | Config-B | Config-C | Config-D | Ours |
|---|---|---|---|---|---|
| **PSNR↑** | 16.29 | 17.09 | 19.64 | 17.50 | **23.33** |

[Knapitsch et al. 2017]. Following Meuleman et al. [2023], we also use 5 scenes of the **Static Hikes** dataset [Meuleman et al. 2023]. We estimate the monocular depth per frame with [Ke et al. 2023], and we also show the sparse view synthesis results using another monocular depth estimator [Birkl et al. 2023] in the supplementary. For training, we select $n$ evenly distributed frames and use the others for testing. For example, when $n = 3$, the first, middle and last frames are used for training.

## 4.1 Comparison

We compare with pose-free methods: COLMAP-Free 3DGS (CF 3DGS) [Fu et al. 2023], NoPe-NeRF [Bian et al. 2023], LocalRF [Meuleman et al. 2023]; pose-required reconstruction methods: Instant-NGP [Müller et al. 2022], 3DGS [Kerbl et al. 2023]; a pose-required sparse-view synthesis method: FSGS [Zhu et al. 2023]; and a pose-required generalizable method: GNT [Varma et al. 2023].

*Quantitative Evaluation.* We report the averaged evaluation results of testing views on all 8 scenes of the Tanks&Temples dataset in Table 1. We evaluate on the case $n = 3, 6, 12$ and measure the difference between synthesized results and ground-truth images. For the Static Hikes dataset, since SfM fails in many cases, we compare with pose-free methods only in Table 2 and report the average results of testing views on all 5 scenes. Our method achieves the best metrics compared to other pose-free methods in most cases, and outperforms pose-required methods, which is also shown in Fig. 1. Notice that the Static Hikes dataset comes with stronger ambiguities and non-smooth camera trajectories, resulting in relatively lower metrics of ours compared to those on the Tanks&Temples dataset, which is analyzed in the supplementary. Even though LocalRF achieves higher metrics of PSNR and SSIM in terms of 12 views in the Static Hikes dataset, our method has much lower LPIPS metrics and performs better than it on the Tanks&Temples dataset.

We also report the effect of the number of training views on the PSNR metric of testing views for the “Horse” scene (120 frames in total) in Table 3. We compare with standard view synthesis methods: Instant-NGP and 3DGS. Our method achieves a high metric with sparse views and outperforms alternatives in all cases.

We also evaluate the performance of our method with respect to the ordering of training images. We experiment on the 3 training images case of Tanks & Temples dataset, and randomly shuffle the training images before feeding them into our sparse view synthesis pipeline. Since our pipeline relies on overlapping between two consecutive training images, random shuffling reduces the overlapping compared to the original order and therefore increases the difficulty of registration. In terms of testing views, the metrics of PSNR, SSIM, LPIPS change from 20.37, 0.66, and 0.26 when using the original order into 19.06, 0.59, and 0.30 when using randomly shuffled order.

*Qualitative Evaluation.* We show novel views synthesized from sparse inputs on one scene in Fig. 1 and four scenes in Fig. 8. Fitting a solution from scratch or sparse initialization is ambiguous, resulting in noisy and blurry backgrounds in "Forest", "Family", "Ignatius", "Francis" and "Barn", which also lose high-frequency details such as words under the statue in "Family" in Instant-NGP, NoPe-NeRF, 3DGS, and FSGS. Besides, NoPe-NeRF, GNT, and CF 3DGS fail to handle sparse training views, resulting in the image mismatch in "Francis". In comparison, our method achieves better results in terms of both synthesis quality and pose alignment.

We investigate the effects of the number of training views in Fig. 9. We compare with standard pose-required view synthesis methods: Instant-NGP and 3DGS. Other methods struggle to synthesize high-quality novel view results, with blurry or high-frequency artifacts. Our results are still ambiguous with only 3 views, but are greatly improved with 4 and more views.

## 4.2 Ablation Study

We compare with various baselines to validate our proposed algorithm for achieving a coarse solution. They are all passed through a low-pass filter and refined using the standard method [Kerbl et al. 2023]. To validate the correspondence-based optimization, we set $\lambda_1 = 0$ in Eqn. 4, denoted as "Config-A". To validate the proposed differentiable approximate surface rendering, we replace the proposed rendering with the one in [Chung et al. 2023] and its extension to surface points as "Config-B". To validate the adjustment which aligns the monocular depth, we skip the adjustment step in the Fig. 4 (c) as "Config-C". Besides, we directly back-project pixels into the scene with the original monocular depth and camera poses estimated by SfM, and denote this as "Config-D".

We use the "Museum" scene in Tanks&Temples and evenly select 6 frames as the training views, with the other 94 frames as testing views for evaluation. We report metrics in Table 4. Our full model achieves the best metrics. We show the effects of applying low-pass filtering and refinement in Fig. 10 (a) and qualitative comparison of the different configurations in Fig. 10 (b). The synthesis quality benefits from refinement. Moreover, in comparison, "Config-A", which does not leverage correspondences, and "Config-B" cannot register camera poses successfully, resulting in missing regions. Note that correspondence detection does not fail, but the camera optimization fails in "Config-B". "Config-C" and "Config-D" cannot ensure the alignment between camera poses and monocular depths, resulting in sub-optimal synthesized results.

## 5 CONCLUSIONS, AND FUTURE WORK

Sparse view synthesis is desirable but challenging due to insufficient camera coverage. From the perspective of fitting the signal, the problem is still very ambiguous for under-sampled views despite introducing certain priors, such as depth. Thanks to the explicitness of Gaussian splatting, we propose to construct a coarse solution, where optimization is still involved, for sparse view synthesis. It is then refined for faithful high-frequency details. To effectively reach a coarse solution, we propose to unify registration and adjustment in a fully differentiable pipeline, which leverages long-range information to address the limitation of pixel-wise supervision. A differentiable approximation of the expected surface in Gaussian splatting is also proposed for optimization.

Our method is not without limitations. We can achieve reasonable but not perfect sparse view synthesis for too few training views. The construction of the coarse solution depends on the scale-consistent assumption of estimated monocular depth, which does not hold for complex scenes, such as 360° scenes. By assuming overlapping between consecutive frames, our method also cannot handle unordered collection of images well. Besides, since a Gaussian kernel does not necessarily correspond to a valid surface, a more accurate definition of the surface is welcome. In the future, it would be interesting to explore how to adjust monocular depths more accurately and incorporate novel view constraints to enhance view synthesis quality, and extend our method to unordered collections of images. In conclusion, our work proposes to construct a solution with correspondence-based optimization instead of fitting one from scratch to solve sparse view synthesis without camera poses. We achieve significantly better results than other pose-free methods and even outperform methods which rely on off-the-shelf estimated camera poses. This framework paves the way for future study on sparse view synthesis, few-shot reconstruction, and reconstruction without camera poses.

## ACKNOWLEDGMENTS

This work was supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/ Interior Business Center (DOI/IBC) contract number 140D0423C0076. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI/IBC, or the U.S. Government. We also acknowledge support from NSF grants 2100237 and 2120019 for the Nautilus cluster, gifts from Adobe, Google, Qualcomm and Rembrand, the Ronald L. Graham Chair and the UC San Diego Center for Visual Computing.

## REFERENCES

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2023. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. *ICCV* (2023).

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).

Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. 2000. Plenoptic Sampling. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00).* ACM Press/Addison-Wesley Publishing Co., USA, 307–318.

Shenchang Eric Chen and Lance Williams. 1993. View interpolation for image synthesis. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (Anaheim, CA) *(SIGGRAPH '93).* Association for Computing Machinery, New York, NY, USA, 279–288.

Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. 2023. Depth-Regularized Optimization for 3D Gaussian Splatting in Few-Shot Images. *arXiv preprint arXiv:2311.13398* (2023).

Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. 2022. Depth-supervised NeRF: Fewer Views and Faster Training for Free. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022.* IEEE, 12872–12881.

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).

Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. 1996. The lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. Association for Computing Machinery, New York, NY, USA, 43–54.

Antoine Guédon and Vincent Lepetit. 2023. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. *arXiv preprint arXiv:2311.12775* (2023).

Peter Hedman and Johannes Kopf. 2018. Instant 3D photography. *ACM Trans. Graph.* 37, 4, Article 101 (jul 2018), 12 pages.

Hwan Heo, Taekyung Kim, Jiyoung Lee, Jaewon Lee, Soohyun Kim, Hyunwoo J Kim, and Jin-Hwa Kim. 2023. Robust camera pose refinement for multi-resolution hash encoding. In *International Conference on Machine Learning*. PMLR, 13000–13016.

Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. 2016. Learning-Based View Synthesis for Light Field Cameras. *ACM Trans. Graph.* 35, 6, Article 193 (2016), 10 pages.

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]

Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. 2023. SplaTAM: Splat, Track & Map 3D Gaussians for Dense RGB-D SLAM. *arXiv preprint* (2023).

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).

Mijeong Kim, Seonguk Seo, and Bohyung Han. 2022. InfoNeRF: Ray Entropy Minimization for Few-Shot Neural Volume Rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 12902–12911.

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).

Johannes Kopf, Xuejian Rong, and Jia-Bin Huang. 2021. Robust Consistent Video Depth Estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Christoph Lassner and Michael Zollhöfer. 2021. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1440–1449.

Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. 2009. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal Of Computer Vision* 81 (2009), 155–166.

Marc Levoy and Pat Hanrahan. 1996. Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. Association for Computing Machinery, New York, NY, USA, 31–42.

Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. 2021. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5741–5751.

Leonard McMillan and Gary Bishop. 1995. Plenoptic modeling: an image-based rendering system. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. Association for Computing Machinery, New York, NY, USA, 39–46.

Quan Meng, Anpei Chen, Haimin Luo, Minye Wu, Hao Su, Lan Xu, Xuming He, and Jingyi Yu. 2021. Gnerf: Gan-based neural radiance field without posed camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6351–6361.

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, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. 2019. Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. *ACM Transactions on Graphics (TOG)* (2019).

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

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.

Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. 2022. RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 5470–5480.

Keunhong Park, Philipp Henzler, Ben Mildenhall, Jonathan T Barron, and Ricardo Martin-Brualla. 2023. CamP: Camera preconditioning for neural radiance fields. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–11.

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. 2022. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 3 (2022).

Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. 2020. Accelerating 3D Deep Learning with PyTorch3D. *arXiv:2007.08501* (2020).

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)*.

Noah Snavely, Steven M. Seitz, and Richard Szeliski. 2008. Modeling the World from Internet Photo Collections. *Int. J. Comput. Vis.* 80, 2 (2008), 189–210.

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).

Zachary Teed and Jia Deng. 2020. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 402–419.

Mukund T Varma, Peihao Wang, Xuxi Chen, Tianlong Chen, Subhashini Venugopalan, and Zhangyang Wang. 2023. Is Attention All That NeRF Needs?. In *The Eleventh International Conference on Learning Representations*.

Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. 2023. SparseNeRF: Distilling Depth Ranking for Few-shot Novel View Synthesis. *IEEE/CVF International Conference on Computer Vision (ICCV)* (2023).

Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. 2021. NeRF−−: Neural Radiance Fields Without Known Camera Parameters. *arXiv preprint arXiv:2102.07064* (2021).

Xiuchao Wu, Jiamin Xu, Xin Zhang, Hujun Bao, Qixing Huang, Yujun Shen, James Tompkin, and Weiwei Xu. 2023. ScaNeRF: Scalable Bundle-Adjusting Neural Radiance Fields for Large-Scale Scene Rendering. *ACM Trans. Graph.* 42, 6, Article 261 (dec 2023), 18 pages.

Jiankai Xing, Fujun Luan, Ling-Qi Yan, Xuejun Hu, Houde Qian, and Kun Xu. 2022. Differentiable Rendering using RGBXY Derivatives and Optimal Transport. *ACM Trans. Graph.* 41, 6, Article 189 (dec 2022), 13 pages.

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

Chi Yan, Delin Qu, Dong Wang, Dan Xu, Zhigang Wang, Bin Zhao, and Xuelong Li. 2023. GS-SLAM: Dense Visual SLAM with 3D Gaussian Splatting. *arXiv preprint arXiv:2311.11700* (2023).

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. 2021. pixelNeRF: Neural Radiance Fields From One or Few Images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 4578–4587.

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 Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. 2022. MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (Eds.).

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.

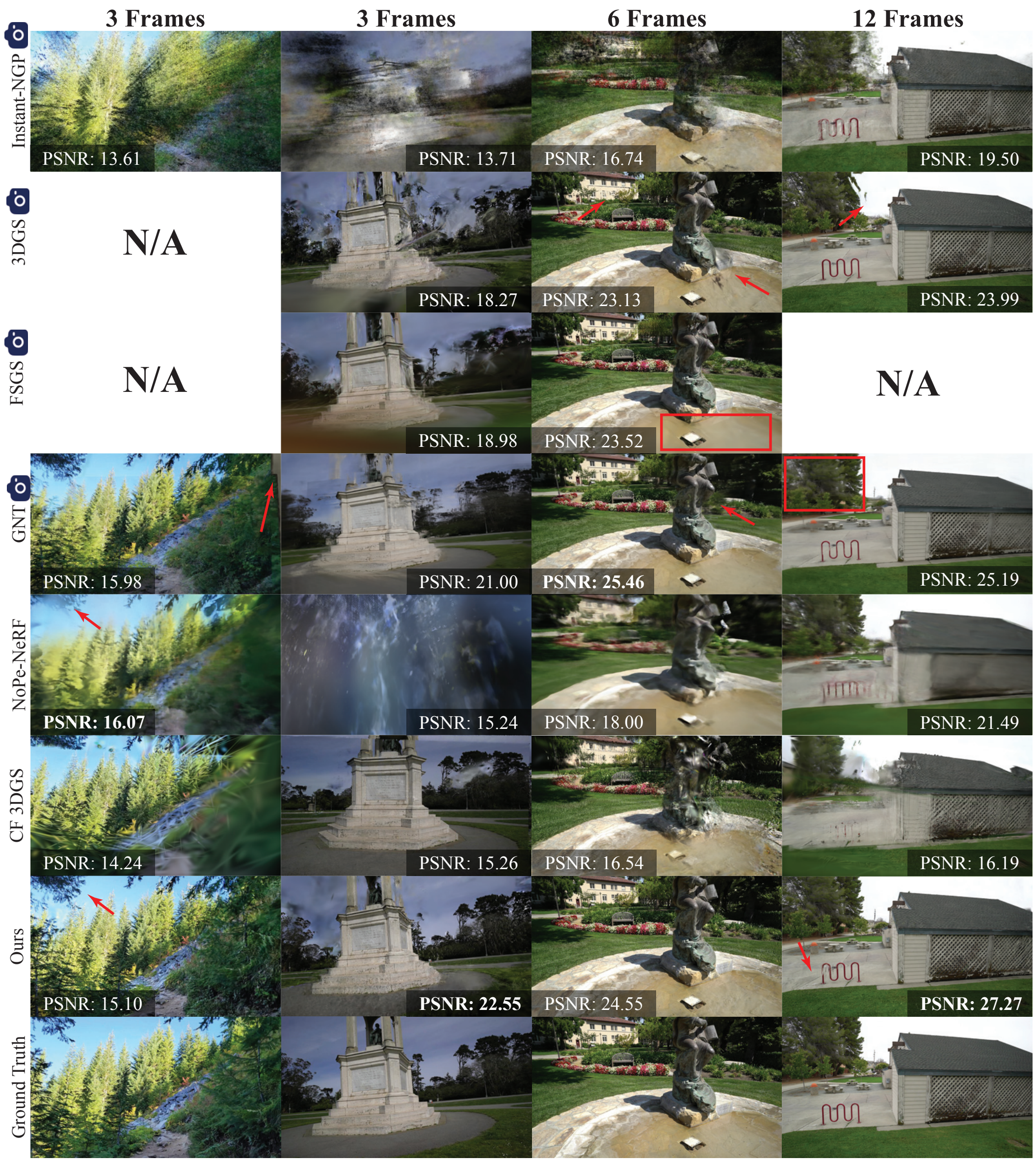


**Figure 8: Qualitative comparison of different methods for sparse view synthesis. From left to right, we use 3, 3, 6 and 12 frames as training views and others for testing. The scenes are, from the left to right: Forest from Static Hikes; Francis, Ignatius, Barn from Tanks&Temples. Some subtle differences in quality are highlighted by arrows/rectangles. Since COLMAP fails for "Forest" (3 views) and multi-view stereo estimation fails for "Barn" (12 views), methods cannot handle these cases are denoted by "N/A". For Instant-NGP and GNT, we show their results on "Forest" (3 views) by giving ground-truth poses which are estimated given both training and testing views. In "Forest", due to its complexity (analyzed in the supplementary), our method cannot achieve high PSNR score despite much more visually pleasing and sharp results. In "Francis", Instant-NGP and 3DGS contain visible artifacts, while FSGS is too smooth on the grass. NoPe-NeRF and CF 3DGS cannot process sparse views well, leading to complete failure or misalignment. In "Ignatius", Instant-NGP, 3DGS and GNT contain blurry artifacts pointed out by arrows, despite slightly higher metric of GNT than ours. FSGS and NoPe-NeRF are overly smooth, while CF 3DGS cannot align the camera pose. In "Barn", Instant-NGP, NoPe-NeRF and CF 3DGS cannot produce sharp rendered results, with blurs around red pillars. 3DGS cannot synthesize faithful background with missing trees and the telegraph pole. GNT synthesizes blurry trees. Our synthesized result also has certain artifacts around red pillars, but enjoys the best PSNR score. Images credit by Meuleman et al. [2023] and Knapitsch et al. [2017].**

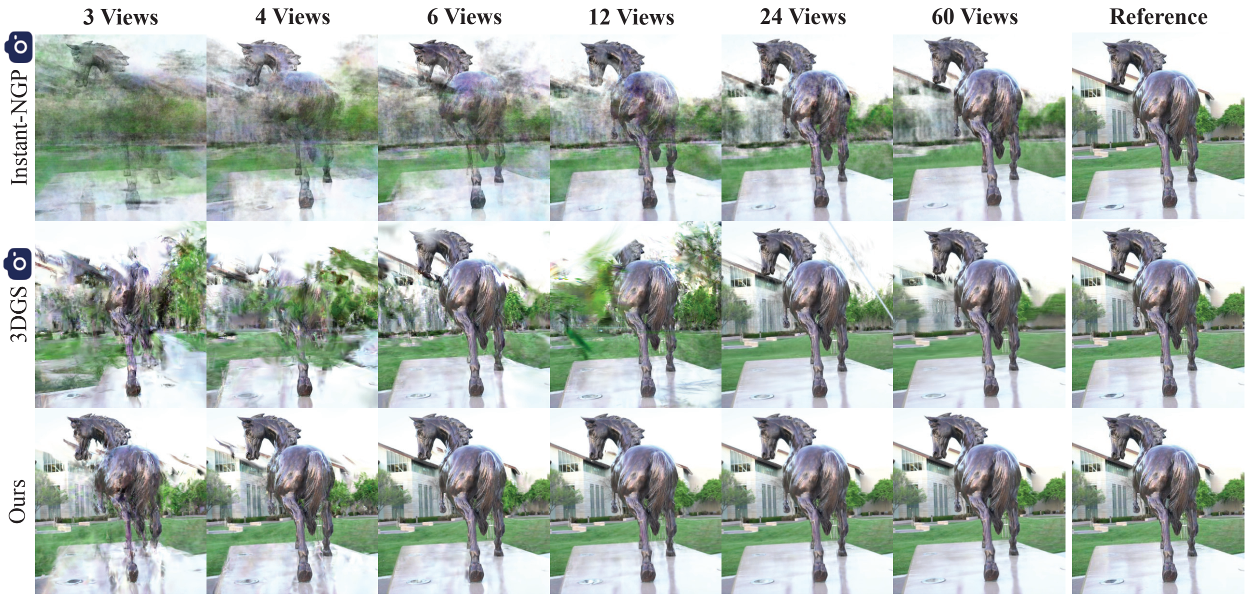


**Figure 9: Qualitative comparison of the effects of the number of training views. For the "Horse" scene in Tanks&Temples (120 frames in total), from left to right, we use 3, 4, 6, 12, 24 and 60 frames as training views and compare the results on the same testing view. Using only 3 views, our results are still somewhat ambiguous, but our method can synthesize faithful results with 4 views, and they improve in accuracy with more views. In comparison, other methods struggle to synthesize faithful results until 60 views, where the background is still not accurate. Instant-NGP features blurry artifacts, while 3DGS features high-frequency artifacts, resulting in occasional failure as in the 12 views case and lower PSNR at very sparse views in Table 3. Image credits by Knapitsch et al. [2017].**

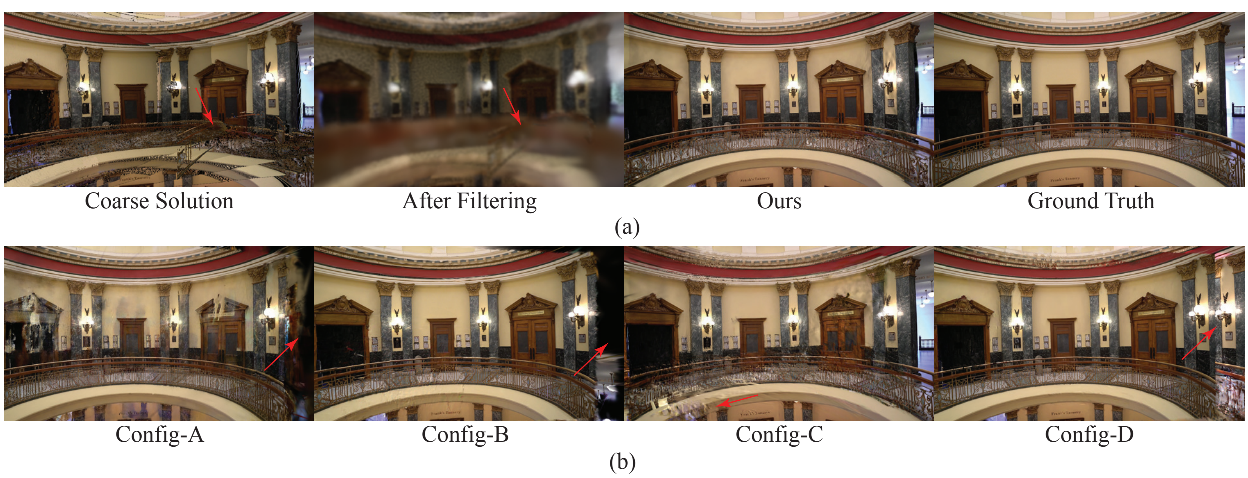


**Figure 10: Given the "Museum" scene in Tanks&Temples (100 frames in total), we use 6 frames as training views and others for testing. We show the synthesized results on the same testing view for different configurations. Regions of interest are emphasized by arrows. (a) We show the effects of applying a low-pass filter and refinement. As pointed by the arrow, even though we manage to align the monocular depths, coarse solution still has high-frequency errors due to inaccuracy of monocular depths. After applying a low-pass filter, error-prune high-frequency information is removed and then faithfully reproduced by refinement. (b) We show the results of different ablation models. The ground-truth is the same with that in (a). "Config-A" and "Config-B" cannot register the camera poses successfully. Therefore, when the testing pose deviates from training poses, there are missing regions, pointed out by arrows. "Config-C" and "Config-D" cannot ensure the alignment between camera poses and monocular depths, leading to artifacts in "Config-C" and wrong synthesized results in "Config-D", pointed out by arrows. Image credits by Knapitsch et al. [2017].**