

Look Gauss, No Pose: Novel View Synthesis using Gaussian Splatting without Accurate Pose Initialization

Christian Schmidt, Jens Piekenbrinck, Bastian Leibe

Abstract—3D Gaussian Splatting has recently emerged as a powerful tool for fast and accurate novel-view synthesis from a set of posed input images. However, like most novel-view synthesis approaches, it relies on accurate camera pose information, limiting its applicability in real-world scenarios where acquiring accurate camera poses can be challenging or even impossible. We propose an extension to the 3D Gaussian Splatting framework by optimizing the extrinsic camera parameters with respect to photometric residuals. We derive the analytical gradients and integrate their computation with the existing high-performance CUDA implementation. This enables downstream tasks such as 6-DoF camera pose estimation as well as joint reconstruction and camera refinement. In particular, we achieve rapid convergence and high accuracy for pose estimation on real-world scenes. Our method enables fast reconstruction of 3D scenes without requiring accurate pose information by jointly optimizing geometry and camera poses, while achieving state-of-the-art results in novel-view synthesis. Our approach is considerably faster to optimize than most competing methods, and several times faster in rendering. We show results on real-world scenes and complex trajectories through simulated environments, achieving state-of-the-art results on LLFF while reducing runtime by two to four times compared to the most efficient competing method. Source code will be available at <https://github.com/Schmiddo/noposegs>.

I. INTRODUCTION

Recently, novel-view synthesis (NVS) methods have emerged as a powerful tool in computer vision, enabling not only the generation of photo-realistic images from unseen viewpoints, but also downstream tasks such as dense 3D scene reconstruction [1], [2], [3] and camera pose estimation relative to a trained model [4], [5], [6]. Solving these tasks benefits various applications, such as computer graphics, robotics, augmented and virtual reality, and 3D scene understanding.

However, most NVS methods depend on accurate camera pose information to build an initial scene representation, limiting the applicability of these approaches in real-world scenarios where acquiring reliable camera poses can be challenging or even impossible.

A promising line of research explores to perform camera pose estimation and 3D reconstruction jointly [7], [8], [6], eliminating the need for perfect aligned camera poses or pose information at all. Additionally, leveraging inherent dependencies between pose and scene information potentially leads to more robust estimations [7], [9]. Despite the significant advancements using novel-view synthesis methods, achieving



Fig. 1. Our proposed approach enables fast, Gaussian Splatting based 3D reconstruction and photo-realistic novel-view synthesis while simultaneously estimating and refining the camera poses, as visualized on this reconstruction of the *horns* scene from LLFF [10]. This reconstruction was performed without camera pose information.

robust pose estimation or joint reconstruction and pose estimation remain an ongoing challenge.

The most prominent methods for NVS are Neural Radiance Fields (NeRFs) [1], [2], which achieve impressive results in terms of visual fidelity by modeling the radiance field as a continuous function using a neural network. The scene can be rendered from any viewpoint using differentiable volumetric ray-marching, which involves shooting a ray for each pixel, querying the radiance field along the ray's path, and accumulating the information using α -compositing. NeRFs are optimized by a photometric reconstruction loss that compares the rendered pixels with the ground-truth, however due to the volumetric ray-marching they can be challenging to optimize and are computationally expensive in both training and rendering.

In contrast to NeRFs that represent a scene as a continuous implicit function, the recently proposed 3D Gaussian Splatting (3DGS) [3] employs an explicit scene representation based on a collection of anisotropic 3D Gaussians. Combined with a differentiable tile-based splatting renderer implemented in CUDA, 3DGS offers highly efficient optimization even on commodity hardware while still delivering photo-realistic novel-view synthesis results in real-time. However, similar to NeRFs, 3DGS requires accurate pose information for every training view. In fact, the Gaussian Splatting algorithm is extremely sensitive to inaccurate camera poses, deteriorating even with very small distortions of the camera

All authors are with the Computer Vision Group, RWTH Aachen University, <lastname>@vision.rwth-aachen.de.

position and orientation.

Previous works in the NeRF literature perform pose estimation [5], [4] as well as joint optimization of camera parameters and scene representation [7], [8], [11], [12] by propagating the gradients from the rendering loss back to the camera parameters. Inspired by this, we propose to extend the 3D Gaussian Splatting framework through differentiable camera pose optimization. As we show, this optimization can directly be integrated into the CUDA rendering kernel, enabling very fast optimization.

While similar in spirit to pose optimization in NeRFs, this approach faces additional challenges for 3DGS. Theoretically, Gaussians have unlimited support; however, this support is usually cut off at some threshold, thus limiting the spatial range over which gradients can influence parts of the model. On the other hand, highly anisotropic Gaussians allow the model to fit training views with high fidelity without accurate modeling of scene geometry. We therefore introduce an anisotropy loss term to avoid overly fast convergence to suboptimal local minima and overfitting to training views. Additionally, we improve the densification and pruning strategy of 3DGS to reduce shape-radiance ambiguities [13], resulting in improved geometry reconstruction and faster optimization and rendering speeds.

As our experiments show, the resulting approach is highly robust to noisy pose initialization, and can perform joint reconstruction and camera pose estimation with minimal assumptions about the input camera poses. We show results for pose estimation as well as joint reconstruction and pose refinement on real-world scenes from the *LLFF* [10] dataset, joint reconstruction and pose refinement on complex trajectories from *Replica* [14], and reconstruction without any known pose information on *Tanks and Temples* [15]. In all evaluated cases, our proposed approach achieves state-of-the-art novel-view synthesis and pose estimation results, while enabling significantly faster runtimes.

In summary, our contributions are: (1) We propose a novel differentiable camera pose estimation approach via Gaussian Splatting that lends itself to a highly efficient CUDA implementation. (2) We improve the robustness of this approach to noisy pose initialization by proposing an anisotropy loss term. (3) We demonstrate experimentally that our approach achieves state-of-the-art NVS and pose estimation / pose refinement results, while requiring only minimal assumptions on the input camera poses.

II. RELATED WORK

Novel View Synthesis. Given a set of posed images of a scene, novel-view synthesis aims to generate realistic images from viewpoints that were not originally captured. In the past years, methods have focused on representing the scene implicitly as radiance field that can be rendered using volumetric ray-marching [1], [16], [2]. These approaches often combine some positional encoding together with an MLP, collectively referred to as Neural Radiance Fields (NeRFs). Each pixel of the image is computed as the approximated integral over a ray through the NeRF volume, which usually

requires many evaluations of the radiance field, making these approaches very slow to train and evaluate. Follow-up methods improved the speed and quality of NeRFs, leading to state-of-the-art methods for novel-view synthesis. These speedups were possible by leveraging spatial data structures with the combination of smaller neural networks [16], [17], [2] or even no neural networks [18].

Compared to the aforementioned NeRF methods that rely on ray-casting, point-based techniques use an explicit unordered set of geometry as representation of the scene, *e.g.* point clouds, which is rendered using differentiable rasterization or splatting algorithms [19], [20], [21], [22]. Kerbl *et al.* [3] recently proposed 3D Gaussian Splatting (3DGS), a novel point-based method, that utilizes a set of anisotropic Gaussians to represent the scene. A key advantage of 3DGS is its efficient rendering achieved through a differentiable tile-based splatting algorithm that is implemented in CUDA, enabling fast optimization and real-time rendering.

Camera Pose Estimation. Estimating camera poses from images is a prevalent task in computer vision. Existing pose estimation methods typically rely on a known 3D scene to establish 2D-3D correspondences that are used to derive the poses. In recent years, research has explored pose estimation based on the implicit 3D representation of NeRFs. Some approaches have adopted the idea of correspondences and combined them with NeRFs [23], [24]. Another line of work leverages the capabilities of photo-realistic rendering from any viewpoint for a direct comparison with the actual image in order to estimate the camera pose [5], [4]. However, NeRFs can be challenging to optimize and are computationally expensive to train and render [5], [7], [9].

Recently, iComMa [25] is among the first to perform camera pose estimation based on 3DGS. Compared to our work, they use an additional local feature matching model for an end-to-end matching loss, whereas our method is based on the photometric loss.

Joint Reconstruction and Pose Estimation. The aforementioned novel-view synthesis techniques rely on posed images to densely reconstruct the 3D scene. Consequently, Structure from Motion (SfM) methods, *e.g.* COLMAP [26], are frequently employed to estimate camera poses and acquire an initialization for point-based approaches. Despite their prevalence, SfM methods suffer from inaccurate camera pose estimations due to unreliable keypoint matches in regions with low texture or repetitive patterns. We demonstrate in Sec. IV-E that our method with pose refinement achieves comparable or better novel-view synthesis results compared to relying solely on SfM initialization.

For NeRFs, there have been several works that tackle the task of jointly reconstructing 3D geometry and estimating camera poses [8], [7], [11], [9], [12], [27], [28]. Notably, Nope-NeRF [9] showed that reconstructing a scene without prior assumptions on camera poses is possible, using only geometric cues from a monocular depth estimator.

Concurrent to our work, Colmap-Free Gaussian Splatting (CF-3DGS) [6] proposes to estimate the relative poses be-

tween consecutive video frames and iteratively extend the set of Gaussians. While we experiment with a similar initialization scheme for the trajectory, we use a different process to build a consistent Gaussian representation. Notably, our method does not require an iterative estimation of poses and therefore has lower runtime, especially for long video sequences. Additionally, our method can be adjusted to work on unordered image collections, which to the best of our knowledge makes us the first to support this for 3DGS. Consequently, the runtime of our method is not linear in the number of input frames. We compare against CF-3DGS in section Sec. IV-D.

III. METHOD

We first recapitulate the original differentiable Gaussian Splatting approach (Sec. III-A). We then describe our extension to this formulation in Sec. III-B, and details for the proposed optimization procedure in Secs. III-C and III-D.

A. Gaussian Splatting

Gaussian Splatting [3] represents a 3D scene as a set of oriented anisotropic Gaussians \mathcal{G} . Every Gaussian is parametrized by a 3D mean $\boldsymbol{\mu}_i$, a covariance matrix $\boldsymbol{\Sigma}_i$, a set of feature vectors \mathbf{f}_i and a scalar opacity o_i . We denote the corresponding 2D quantities, *i.e.* projected mean and covariance, with $\hat{\boldsymbol{\mu}}_i$ and $\hat{\boldsymbol{\Sigma}}_i$. In order to facilitate gradient based optimization, the 3D covariance matrix is decomposed into an orientation represented as a unit quaternion \mathbf{q}_i and a scale vector \mathbf{s}_i such that

$$\boldsymbol{\Sigma}_i = \mathbf{R}(\mathbf{q}_i) \text{diag}(\mathbf{s}_i) \text{diag}(\mathbf{s}_i)^T \mathbf{R}(\mathbf{q}_i)^T. \quad (1)$$

To render a view from a given camera with intrinsics \mathbf{K} and extrinsics $\mathbf{T} = (\mathbf{R}_c, \mathbf{t}_c)$, the Gaussians are projected onto the image plane and rasterized via alpha blending. More formally, the projected 2D mean of a Gaussian is $\hat{\boldsymbol{\mu}}_i = \pi(\mathbf{K}\mathbf{T}\boldsymbol{\mu}_i)$ where π is the projective transformation $\pi(\mathbf{p}) = (p_x/p_z, p_y/p_z, 1)$. The 2D covariance $\hat{\boldsymbol{\Sigma}}_i$ of a Gaussian as projected onto the viewing plane of camera c is then computed based on a first-order Taylor approximation of π [3],

$$\hat{\boldsymbol{\Sigma}}_i = [\mathbf{J}_i \mathbf{R}_c \boldsymbol{\Sigma}_i \mathbf{R}_c^T \mathbf{J}_i^T]_{2 \times 2}, \quad (2)$$

where \mathbf{J}_i is the Jacobian of the aforementioned approximation, and $[\cdot]_{2 \times 2}$ denotes the upper two-by-two submatrix.

The influence of a splat $(\hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\Sigma}}_i)$ on a pixel \mathbf{p} is determined by the 2D distribution of the splat and its opacity o_i :

$$\alpha_i = o_i \exp\left(-\frac{1}{2}(\mathbf{p} - \hat{\boldsymbol{\mu}}_i)^T \hat{\boldsymbol{\Sigma}}_i^{-1} (\mathbf{p} - \hat{\boldsymbol{\mu}}_i)\right). \quad (3)$$

The color of a splat is modeled via spherical harmonics $\mathbf{c}_i = \text{SH}(\mathbf{f}_i, \mathbf{d}_i^c)$, which calculate view-dependent color using a set of basis functions, learnable parameters \mathbf{f}_i , and the viewing direction $\mathbf{d}_i = \frac{\boldsymbol{\mu}_i - \mathbf{o}_c}{\|\boldsymbol{\mu}_i - \mathbf{o}_c\|}$. Note that $\mathbf{o}_c = -\mathbf{R}_c^{-1} \mathbf{t}_c$ denotes the position of the camera center in world coordinates.

The splats are then sorted by depth and are composited front-to-back via alpha blending in order to find the final

pixel color \mathbf{C}_{xy} :

$$\mathbf{C}_{xy} = \sum_i^N \mathbf{c}_i \alpha_i \prod_j^{i-1} (1 - \alpha_j). \quad (4)$$

The term $T_i = \alpha_i \prod_j^{i-1} (1 - \alpha_j)$ is also called *transmittance*.

The differentiable rasterizer of Kerbl *et al.* [3] efficiently computes gradients for positions, orientations, scalings, opacities, and spherical harmonics of all Gaussians, allowing rapid fitting of 3D scenes given a few posed images. However, it does not provide gradients for the camera extrinsics \mathbf{T} .

B. Camera pose optimization via Gaussian Splatting

We propose to extend the Gaussian Splatting formulation to also compute gradients for the extrinsic camera parameters \mathbf{R}_c and \mathbf{t}_c . To this end, we model the camera pose as an element of $\text{SE}(3)$, the Lie group of 3D rigid body motions, and derive the gradients for group elements in the respective tangent space. This allows us to use the renderer together with existing libraries for tangent space backpropagation, enabling fast and accurate pose estimation and other downstream tasks.

A Lie group is a group that is also a smooth manifold. The lie algebra of the group $\text{SE}(3)$ is denoted $\text{se}(3)$ and is defined as the tangent space at the identity element; it is isomorphic to \mathbb{R}^6 . We use the hat $\hat{\cdot} : \mathbb{R}^6 \rightarrow \text{se}(3)$ and vee $\vee : \text{se}(3) \rightarrow \mathbb{R}^6$ operators to map between \mathbb{R}^6 and $\text{se}(3)$. The *exponential map* $\text{Exp}(\tau) : \mathbb{R}^6 \rightarrow \text{SE}(3)$ and its counterpart, the *logarithmic map* $\text{Log}(\mathcal{X}) : \text{SE}(3) \rightarrow \mathbb{R}^6$ map elements of \mathbb{R}^6 to elements of the Lie group and vice versa. We can then parametrize the camera transform as $\mathbf{T} = \text{Exp}(\boldsymbol{\tau})$, $\boldsymbol{\tau} \in \mathbb{R}^6$. For simplicity of notation, we still denote gradients on $\boldsymbol{\tau}$ as $\frac{\partial \mathcal{L}}{\partial \mathbf{T}}$.

Lie Groups. Given a function $f : \mathcal{M} \rightarrow \mathcal{M}'$ that maps between two Lie groups \mathcal{M} and \mathcal{M}' , it is possible to extend the differential to compute gradients with respect to group elements [29]:

$$Df(\mathcal{X})[\mathbf{v}] = \lim_{t \rightarrow 0} \frac{\text{Log}(f(\text{Exp}(t\mathbf{v})\mathcal{X})f(\mathcal{X})^{-1})}{t}, \quad (5)$$

where \mathbf{v} is an element of the tangent space of \mathcal{X} . Intuitively, \mathbf{v} denotes the direction of the differential. This allows to extend the notion of Jacobians as

$$\left(\frac{\partial f(\mathcal{X})}{\partial \mathcal{X}}\right)_{ij} = \lim_{t \rightarrow 0} \frac{\langle \text{Log}(f(\text{Exp}(te_j)\mathcal{X})f(\mathcal{X})^{-1}), \mathbf{e}'_i \rangle}{t}, \quad (6)$$

where $\mathbf{e}_i, \mathbf{e}'_j$ are orthonormal basis vectors of the input and output spaces of f , respectively, and $\langle \cdot, \cdot \rangle$ denotes the vector product operator. Note that every Euclidean space, together with the usual vector addition, forms a Lie group with its Lie algebra being itself. Thus Eq. (6) generalizes the usual notion of Jacobians to mappings between arbitrary Lie groups.

Deriving the Jacobians. During optimization, our aim is to minimize some loss function \mathcal{L} using gradient descent. For this, we need to determine the gradient of the loss with respect to the camera parameters, $\frac{\partial \mathcal{L}}{\partial \mathbf{T}}$. Three terms contribute to $\frac{\partial \mathcal{L}}{\partial \mathbf{T}}$: the 2D mean $\hat{\boldsymbol{\mu}}_i$, the 2D covariance matrices $\hat{\boldsymbol{\Sigma}}_i$, and

the view-dependent color $\mathbf{c}_i = \text{SH}(\mathbf{f}_i, \mathbf{d}_i^c)$. Using the chain rule, we get

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{T}} &= \sum_i \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{\mu}}_i} \frac{\partial \hat{\boldsymbol{\mu}}_i}{\partial \boldsymbol{\mu}_i^c} \frac{\partial \boldsymbol{\mu}_i^c}{\partial \mathbf{T}} \\ &+ \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{\Sigma}}_i} \left(\frac{\partial \hat{\boldsymbol{\Sigma}}_i}{\partial \mathbf{J}_i} \frac{\partial \mathbf{J}_i}{\partial \boldsymbol{\mu}_i^c} \frac{\partial \boldsymbol{\mu}_i^c}{\partial \mathbf{T}} + \frac{\partial \hat{\boldsymbol{\Sigma}}_i}{\partial \mathbf{R}_c} \frac{\partial \mathbf{R}_c}{\partial \mathbf{T}} \right) + \frac{\partial \mathcal{L}}{\partial \mathbf{c}_i} \frac{\partial \mathbf{c}_i}{\partial \boldsymbol{o}_c} \frac{\partial \boldsymbol{o}_c}{\partial \mathbf{T}}, \end{aligned} \quad (7)$$

where we use $\boldsymbol{\mu}_i^c = \mathbf{T}\boldsymbol{\mu}_i$ to denote the mean of Gaussian i in the reference frame of camera c . The original formulation already computes $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_i^c}$, $\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{\Sigma}}_i}$, and $\frac{\partial \mathcal{L}}{\partial \mathbf{c}_i}$.

The Jacobian of a transformation applied to a point $\mathbf{y} = \mathbf{T}\mathbf{x}$ is well-known [29] as

$$\frac{\partial \mathbf{y}}{\partial \mathbf{T}} = (\mathbf{I} | -[\mathbf{y}]^\times), \quad (8)$$

where $[\mathbf{y}]^\times$ denotes the 3-by-3 skew matrix with entries of \mathbf{y} , and $(\cdot | \cdot)$ denotes matrix concatenation; $\frac{\partial \mathbf{y}}{\partial \mathbf{T}}$ is a 3-by-6 matrix relating changes in \mathbf{y} with changes in the tangent space representation of \mathbf{T} . The Jacobian of the group inverse on $\text{SE}(3)$ can be found as [30]

$$\frac{\partial \mathbf{T}^{-1}}{\partial \mathbf{T}} = - \begin{pmatrix} \mathbf{R} & [\mathbf{t}]^\times \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{pmatrix}, \quad \mathbf{T} = (\mathbf{R}, \mathbf{t}). \quad (9)$$

With this, we can compute the gradient contributions of the 2D/3D mean and the spherical harmonics:

$$\frac{\partial \boldsymbol{\mu}_i^c}{\partial \mathbf{T}} = (\mathbf{I} | -[\boldsymbol{\mu}_i^c]^\times) \quad (10)$$

$$\frac{\partial \mathbf{c}_i}{\partial \mathbf{T}} = \frac{\partial \mathbf{c}_i}{\partial \boldsymbol{o}_c} \frac{\partial \boldsymbol{o}_c}{\partial \mathbf{T}} = \frac{\partial \mathbf{c}_i}{\partial \boldsymbol{\mu}_i} (\mathbf{R}_c | [\boldsymbol{o}_c]^\times \mathbf{R}_c), \quad (11)$$

where we used the fact that spherical harmonics are only dependent on the direction from 3D mean $\boldsymbol{\mu}_i$ to the camera center \boldsymbol{o}_c , and $\boldsymbol{o}_c = -\mathbf{R}_c^{-1}\mathbf{t}_c$.

To compute the gradient contribution of the term $\hat{\boldsymbol{\Sigma}}_i = \mathbf{J}_i \mathbf{R}_c \boldsymbol{\Sigma}_i \mathbf{R}_c^T \mathbf{J}_i^T$, we apply Eq. (6) on $f(\mathbf{T}) = \mathbf{R}_c$. Here, f is a function mapping from $\text{SE}(3)$ to $\mathbb{R}^{3 \times 3}$, thus the Jacobian can be written as a matrix in $\mathbb{R}^{3 \times 9}$, relating entries of $\boldsymbol{\tau}$ with entries of \mathbf{R}_c :

$$\frac{\partial f(\mathbf{T})}{\partial \mathbf{T}} = \frac{\partial}{\partial \boldsymbol{\tau}} \Big|_{\boldsymbol{\tau}=\mathbf{0}} \text{Exp}(\boldsymbol{\tau}) \mathbf{R}_c \quad (12)$$

$$= ([\mathbf{e}_1]^\times \mathbf{R}_c | [\mathbf{e}_2]^\times \mathbf{R}_c | [\mathbf{e}_3]^\times \mathbf{R}_c). \quad (13)$$

We integrate these computations with the existing Gaussian Splatting rasterizer using the CUDA programming environment. This allows us to compute accurate gradients on the camera parameters with minimal overhead, leading to significant reductions of runtime compared to other approaches.

C. Splatting with Pose Optimization

For joint reconstruction and pose refinement, we iteratively optimize over all Gaussian parameters and all camera poses. Given a set of initial pose estimates $\mathcal{P} = \{\mathbf{T}_1, \dots, \mathbf{T}_K\}$ corresponding to training images I_1, \dots, I_K , and an initial set of Gaussians \mathcal{G} , we want to find optimal poses \mathcal{P}^* and an optimal scene representation \mathcal{G}^* such that

$$\mathcal{P}^*, \mathcal{G}^* = \arg \min_{\mathcal{P}, \mathcal{G}} \left(\sum_I \mathcal{L}(\hat{I}, I) \right). \quad (14)$$

Our loss \mathcal{L} is a combination of an image-based loss \mathcal{L}_{RGB} and an additional 3D anisotropy loss term \mathcal{L}_{An} to regularize the scene representation:

$$\mathcal{L} = \mathcal{L}_{\text{RGB}} + \mathcal{L}_{\text{An}}. \quad (15)$$

The image-based loss \mathcal{L}_{RGB} is a weighted sum of an L_1 loss and a DSSIM term, as in the original Gaussian Splatting formulation [3]:

$$\mathcal{L}_{\text{RGB}} = (1 - \beta) \mathcal{L}_{L_1} + \beta \mathcal{L}_{\text{DSSIM}}, \quad (16)$$

where we use $\beta = 0.2$. The regularization term \mathcal{L}_{An} helps avoiding overly fast convergence of the Gaussians to suboptimal local minima and overfitting to the training views. It limits the ratio between the major and minor axis of each Gaussian:

$$\mathcal{L}_{\text{An}} = \frac{1}{N} \sum_{i=1}^N [\max\{\mathbf{s}_i\} / \min\{\mathbf{s}_i\} - r]_+, \quad (17)$$

ensuring that Gaussians do not degenerate into arbitrarily thin spikes.

Additionally, we employ an adaptive thresholding scheme to remove transparent Gaussians. The default densification and pruning strategy in 3DGS leads to a large number of semi-transparent Gaussians which contribute little to the actual scene representation while increasing shape-radiance ambiguity [13]. We adapt the opacity threshold so that at most N_{tgt} Gaussians remain after pruning. To make this scheme more effective, we add an L_1 loss on the opacity of each Gaussian during the first 10k steps. The remaining Gaussians cover larger areas and are more opaque, leading to reduced shape-radiance ambiguities, improved geometry reconstruction, and faster rendering.

D. Camera pose estimation

While the focus of our method lies on *pose refinement*, we can also use it to *estimate* an unknown camera pose, e.g., from a trained model or between nearby frames in a video. This allows us to reconstruct scenes with just minimal assumptions on the camera pose distribution. (Essentially, we only require that the cameras show partially overlapping geometry). We show results on videos from the Tanks and Temples dataset [15] in Sec. IV-D. The idea here is to estimate a rough trajectory from neighbouring frames and then to refine this trajectory with our method for joint reconstruction and pose refinement.

For two nearby frames I_t, I_{t+1} of an input video stream, we can estimate the relative pose between them as follows. Using a pretrained off-the-shelf monocular depth estimator D , we compute a relative depth map $D(I_t)$. We then use this depth map and the known camera intrinsics \mathbf{K} to unproject a subset of pixels into a point cloud P_t . We interpret these points as isotropic Gaussians and set their color to the respective pixel color and their scale to the mean 3D distance to the three nearest neighbours in the point cloud, yielding a set of 3D Gaussians \mathcal{G}_t . We can then optimize \mathcal{G}_t to better depict the input frame:

$$\mathcal{G}_t^* = \arg \min_{\mathcal{G}} \left(\mathcal{L}_{\text{RGB}}(\hat{I}_t, I_t) \right). \quad (18)$$

This optimization converges very quickly (typically in less than 100 steps). Due to the efficient rasterizer and the limited number of 3D points, this initialization takes less than a second per frame. After initializing \mathcal{G}_t , we estimate the local transformation \mathbf{T}_{t+1} between frames I_t and I_{t+1} such that it minimizes the rendering loss of \mathcal{G}_t compared to frame I_{t+1} :

$$\mathbf{T}_t^* = \arg \min_{\mathbf{T}} \mathcal{L}_{\text{RGB}}^M(\hat{I}_t, I_{t+1}). \quad (19)$$

Here, we use a confidence-masked loss function $\mathcal{L}_{\text{RGB}}^M$. In particular, we apply a mask M on the rendered image before computing the L_1 loss for pose estimation in order to evaluate the loss only on pixels with high accumulated alpha. This makes the optimization more robust to moving objects and gaps in the rendering due to missing geometry. We define the mask as a pixel-wise threshold on the accumulated transmittance:

$$M_{x,y} = \begin{cases} 1 & \text{if } \sum_i^N T_i > 0.99 \\ 0 & \text{otherwise.} \end{cases} \quad (20)$$

The masked loss is then

$$\mathcal{L}_{L_1}^M(\hat{I}, I) = \sum_{x,y} |M_{x,y}(I_{x,y} - \hat{I}_{x,y})|. \quad (21)$$

IV. EXPERIMENTS

We perform experiments on three tasks: *camera pose estimation* (Sec. IV-B), *joint reconstruction and pose refinement* (Sec. IV-C), and *reconstruction without pose information* (Sec. IV-D).

A. Experimental Setup

We compare against several previous works [9], [7] as well as a concurrent similar approach [6].

Datasets. We evaluate our method on *LLFF*, *Replica*, and *Tanks and Temples*. *LLFF* [10] is a set of 8 forward-facing real scenes captured with a handheld smartphone camera. The ground-truth poses in this dataset are estimated with COLMAP [26]. As in previous works, we reserve the last 10% of images for the NVS test set [7].

Replica [14] contains high-quality renderings of indoor scenes. In particular, we evaluate our method on the trajectories generated by [31], which correspond to 5 offices and 3 apartment rooms. We subsample the trajectories to 400 frames and use every eighth frame for testing.

Tanks and Temples [15] consists of several real-world scenes captured with a high-resolution camera. Similar to previous work [9], we take every eighth image for testing and train on the remaining images. We use the camera poses estimated by COLMAP as ground truth. Compared to *LLFF*, *Tanks and Temples* contains much larger scenes with longer and more complex trajectories.

Metrics. We evaluate our method on the tasks of novel view synthesis (NVS) and camera pose estimation. For NVS, we report peak-signal-to-noise ratio (PSNR), structural similarity measure (SSIM), and learned perceptual image patch similarity (LPIPS), as commonly done in previous works [9], [7], [28], [11]. For pose estimation, we use two slightly different sets of metrics. For *pose refinement*, we use

the absolute rotation and translation error between a pose \mathbf{T}_i and the corresponding ground-truth pose \mathbf{T}_i^* . For *pose estimation*, we compute the relative rotation and translation error (RPE_r and RPE_t, respectively), as well as the absolute trajectory error (ATE). For all methods and all tasks, we compute pose estimation metrics after aligning the predicted and ground-truth trajectories using Procrustes alignment.

Implementation Details. We integrate the gradient computation of the camera parameters into the Gaussian Splatting CUDA kernel and implemented the remaining parts of our method in PyTorch.

Pose Estimation. We optimize the reference models with poses estimated by COLMAP [26], using our anisotropy loss and adaptive opacity thresholding. We set $r = 10$ and $N_{\text{igt}} = 256'000$.

Joint Reconstruction and Pose Refinement. For the pose refinement task, we initialize the point cloud by unprojecting points from the depth map of an off-the-shelf monocular depth estimator and treating them as Gaussians as described in Sec. III-C. In order to be comparable to previous works we use DPT [32], in particular the same model and weights as NopeNeRF [9]. Note that we use depth only for initialization; we do not use ground-truth depth at any point. We start the camera learning rate at 10^{-2} and decay to 10^{-4} , using a cosine decay schedule. In the anisotropy regularizer, we set the maximum ratio to $r = 10$. We adopt the majority of hyperparameter settings from the original 3D Gaussian Splatting paper [3], with the following exceptions: We start with a positional learning rate of 1.6×10^{-2} , decaying exponentially to 1.6×10^{-4} . Contrary to the original implementation, the learning rate is not scaled by the scene extent, since we have no (reliable) information on poses.

Similar to previous work [7], [11], [28], [9], we perform test-time optimization of camera poses in order to minimize the influence of suboptimal pose estimation on NVS performance. We optimize the pose of each test frame for up to 200 steps using the same hyperparameters as for the relative pose estimation during the online phase.

For the comparison to pose-free methods on video clips, we estimate an initial trajectory using a simple per-frame optimization scheme. From each frame, we unproject 50'000 points and optimize them to fit the frame for 100 steps. We optimize the relative pose between consecutive frames for 200 steps, using a learning rate of 10^{-3} decaying to 10^{-4} using a cosine schedule. This scheme is very similar to the initial phase of CF-3DGS [6]. However, our method requires much less time to estimate the relative poses due to our efficient CUDA integration. Additionally, the main part of our method is independent of this simple per-frame scheme; our optimization scheme works based on any initialization with sufficiently small pose errors. For example, we are able to optimize noisy poses of subsampled trajectories of the Replica dataset [14] (see Sec. IV-C), with substantial camera motion between subsequent frames. In contrast to this, CF-3DGS explicitly proposes an iterative expansion scheme.

We conduct all experiments on a single RTX 3090 GPU.

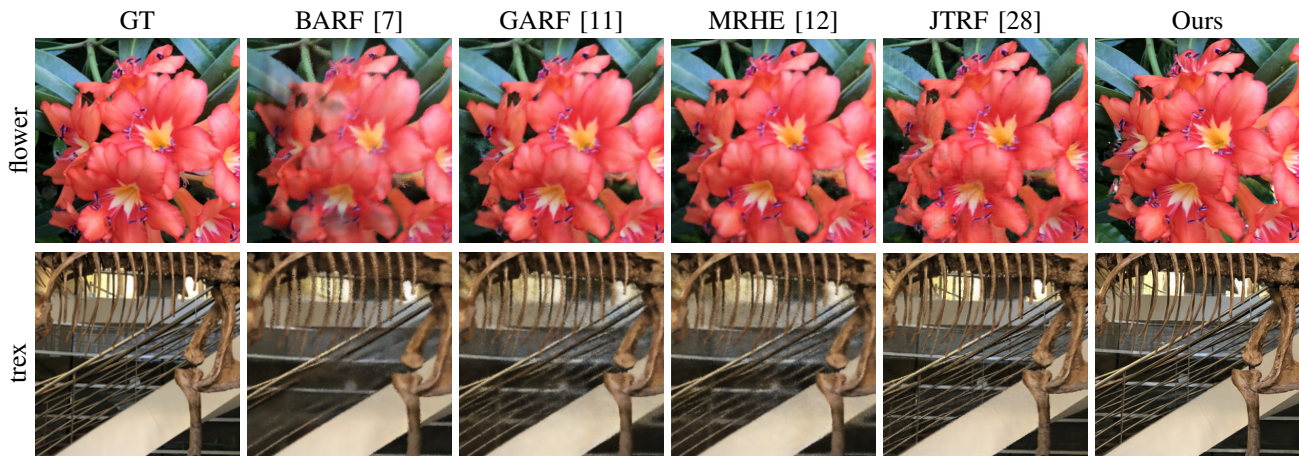


Fig. 2. Qualitative results for pose-free reconstruction and NVS on the *LLFF* dataset [10]. Our method achieves superior reconstruction quality compared to previous techniques. Notably, BARF, GARF and MRHE struggle to capture fine details, as exemplified by the banister in the *trex* scene. Additionally, their rendered images often exhibit blurriness. While JTRF appears sharper at first glance, it suffers from significant pixelation artifacts, particularly evident in the leaves of the flower scene. In contrast, our method produces crisp and artifact-free renderings.

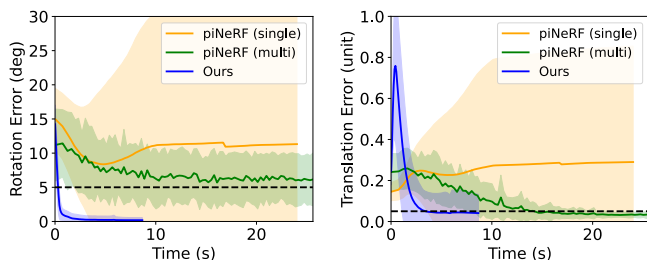


Fig. 3. Convergence behavior of our method compared to piNeRF [4] on a subset of the *LLFF* dataset, *i.e.* fern, flower, fortress, horns and room. The dashed black line represents the thresholds of 5deg for rotation and 0.05 for translation. The colored bold line is the mean of the error over all scenes. Moreover, the \pm std is shown as the colored area around the mean line. Our method uses only a single pose hypothesis, but converges much faster than even the multi-hypothesis baseline.

B. Camera pose estimation from trained models

We first investigate the convergence speed of our method on the task of 6-DoF camera pose estimation from monocular images. Given a trained model of a scene, a single RGB image, and an initial pose estimate, we want to find the correct position and orientation of the camera corresponding to the image. We follow previous work [4] and generate noisy poses by rotating the camera pose sequentially around each axis, sampling rotational noise uniformly from the range $[-15^\circ, 15^\circ]$ and translating it along the world axes by a random offset sampled from $[-0.15, 0.15]$ units. We optimize poses for up to 1000 steps, although most trials converge much earlier. Due to our efficient CUDA implementation, this process takes just a couple of seconds on an NVIDIA RTX 3090. We evaluate our method on real-world scenes from the *LLFF* dataset [10] and compare against piNeRF [4], a recent approach which uses a multi-hypothesis optimization strategy based on Instant-NGP [2]. A visual comparison is shown in Fig. 3, and quantitative results are shown in Tab. I. Compared to the piNeRF baseline, our approach converges much faster on the rotation error, even though piNeRF uses

TABLE I
MEAN ABSOLUTE ROTATION AND TRANSLATION ERROR FOR CAMERA POSE ESTIMATION ON *LLFF* [10].

Scene	Pos	Rot	Rot@5	Pos@0.05
Fern	0.013	0.009	1.000	1.000
Flower	0.024	0.035	1.000	1.000
Fortress	0.146	0.868	0.933	0.800
Horns	0.020	0.015	1.000	0.975
Room	0.004	0.000	1.000	1.000
Mean	0.041	0.185	0.987	0.955
[4] single	0.311	11.054	0.680	0.656
[4] multi	0.038	5.428	0.808	0.760

a highly performant hashgrid representation and optimizes multiple pose hypotheses in parallel. We hypothesize that the initial increase in translational error is due to the fact that we optimize camera poses as elements of $SE(3)$; this leads to a coupling between camera position and camera orientation, as also noted in [4], which optimizes poses in $TxSO(3)$. However, our method still shows rapid convergence after the initial spike. We leave a thorough investigation of the effect of different Lie Group parametrizations and potential multi-hypothesis setups of our method for future work.

C. Joint Reconstruction and Pose Refinement.

Real-World Scenes. We investigate the performance of our method on *LLFF*, a set of forward-facing real-world scenes captured with a handheld smartphone camera. In this setting, we assume *unknown* camera poses and initialize all camera poses to identity. We compare against several state-of-the-art methods for joint reconstruction and pose estimation [7], [28], [12] in Tab. II. Qualitative results are shown in Fig. 2. Our method outperforms all other methods in visual quality, and performs well in terms of camera registration accuracy, while being several times faster to optimize.

Complex Trajectories. We evaluate our method on the

TABLE II
NOVEL VIEW SYNTHESIS AND POSE ESTIMATION RESULTS ON
LLFF [10], TRAINING/OPTIMIZATION TIME OF ONE SCENE.

Method	Novel View Synthesis			Pose Estimation		Time
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	Rot. \downarrow	Trans. \downarrow	
BARF [7]	23.09	0.678	0.275	1.580	0.721	300m
GARF [11]	24.55	0.745	0.216	0.280	0.269	600m
MRHE [12]	24.79	0.772	0.197	0.384	0.258	30m
JTRF [28]	25.27	0.827	-	0.709	0.325	180m
Ours	25.19	0.838	0.123	1.529	0.314	10m

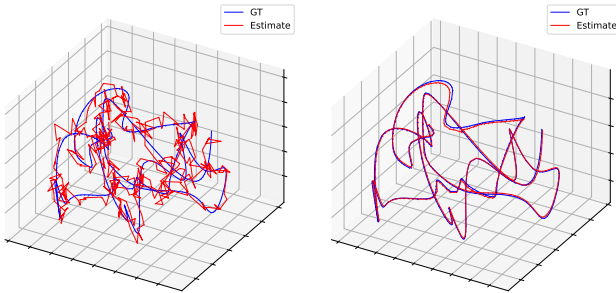


Fig. 4. Qualitative results on `office0` from *Replica* [14]. Left: trajectory at the beginning of optimization. Right: trajectory after optimization. Processing takes around 30 minutes on a single RTX 3090.

Replica dataset [14], using trajectories generated by random walks [31]. *Replica* contains 8 room-scale scenes with challenging geometry, lighting, and camera trajectories. We subsample the trajectories by a factor of 5, and use every eight frame of that for testing. In total, that yields 350 training views and 50 test views for each of the 8 scenes. Here, we perturb the poses by sampling gaussian noise in the tangent space with a standard deviation of 0.05. We show qualitative results in Fig. 4 and quantitative results for NVS and camera pose estimation in Tab. III.

D. Reconstruction without Pose Information

We evaluate NVS performance and pose estimation on a subset of the *Tanks and Temples* benchmark, as proposed in [9]. We compare our method against previous NeRF-based approaches that can be trained without known poses, in particular Nope-NeRF [9] and BARF [7]. In addition, we compare to CF-3DGS [6], a concurrent work that also uses Gaussian Splatting for this task. For all methods, pose estimation performance is evaluated after Procrustes alignment.

We report the results in Tab. IV. Compared to the NeRF baselines, our method achieves consistently better novel view quality, and a similar visual fidelity as CF-3DGS. Notably, our method is more than 40 \times faster than the NeRF baselines, and about 4 \times faster than CF-3DGS (*i.e.*, 20+ hrs/scene for Nope-NeRF, around 2 hrs/scene for CF-3DGS, and less than 30 minutes/scene for our method). Again, our method achieves comparable results to state-of-the-art methods, with a fraction of the runtime.

TABLE III
NOVEL VIEW SYNTHESIS AND POSE ESTIMATION RESULTS ON
REPLICA [14].

Scene	Novel View Synthesis			Pose Estimation		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	RPE $_t$ \downarrow	RPE $_r$ \downarrow	ATE
Room0	28.169	0.858	0.180	1.968	0.364	0.056
Room1	30.906	0.899	0.170	2.316	0.493	0.045
Room2	31.535	0.928	0.161	1.368	0.357	0.034
Office0	42.354	0.982	0.078	0.173	0.047	0.009
Office1	40.510	0.970	0.149	0.652	0.176	0.041
Office2	33.245	0.938	0.193	0.687	0.256	0.061
Office3	35.478	0.957	0.128	0.573	0.093	0.034
Office4	32.487	0.938	0.177	2.458	0.479	0.125
Mean	34.335	0.934	0.154	1.274	0.283	0.051

TABLE IV
COMPARISON ON TANKS AND TEMPLES [15] AGAINST
STATE-OF-THE-ART METHODS FOR JOINT 3D RECONSTRUCTION AND
POSE ESTIMATION. * DENOTES CONCURRENT WORK.

Method	Novel View Synthesis			Pose Estimation		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	RPE $_t$ \downarrow	RPE $_r$ \downarrow	ATE \downarrow
Nerf- [8]	22.50	0.59	0.54	1.735	0.477	0.123
SCNerf [33]	23.76	0.65	0.48	1.890	0.489	0.129
BARF [7]	23.42	0.61	0.54	1.046	0.441	0.078
Nope-NeRF [9]	26.34	0.74	0.39	0.080	0.038	0.006
CF-3DGS* [6]	31.28	0.93	0.09	0.041	0.069	0.004
Ours	31.24	0.92	0.12	0.075	0.069	0.009

E. Ablations

Comparison against 3DGS+COLMAP. We report the results of vanilla Gaussian Splatting with poses estimated by COLMAP in Sec. IV-E. The visual quality of our reconstruction is comparable to splatting using SfM poses, in some cases even better (*e.g.*, the *Family* and *Horse* sequences of *Tanks and Temples*, see Sec. IV-E). This is likely due to a suboptimal reconstruction by COLMAP, leading to slightly wrong camera poses or misplacement of initial 3D points for the initialization of 3DGS. Similarly, we observe a higher visual quality for the forward-facing *LLFF* scenes. The high visual quality and low pose error make our method interesting for postprocessing of, *e.g.*, noisy SLAM trajectories.

V. DISCUSSION

We have proposed a novel approach that integrates camera pose estimation into 3D Gaussian Splatting by extending the high-performance CUDA rendering kernel. We verified the efficacy of our method for camera pose estimation and joint reconstruction and pose refinement on several well-known benchmark datasets, achieving state-of-the-art results in novel view synthesis quality and pose estimation accuracy, while being several times faster than competing methods.

3D Gaussian Splatting approaches are rapidly gaining traction in many application areas where NeRFs have previously been successful, due to their extremely fast differentiable rendering capabilities. Our proposed approach enables those approaches to perform more robustly with less requirements

TABLE V

COMPARISON BETWEEN OUR METHOD, TRAINED WITHOUT POSE INFORMATION, AND GAUSSIAN SPLATTING USING POSES ESTIMATED WITH COLMAP. LEFT: RESULTS ON TANKS AND TEMPLES [15]. RIGHT: RESULTS ON LLFF [10].

Scene	Ours			3DGS+COLMAP			Scene	Ours			3DGS+COLMAP		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow		PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
Ballroom	32.73	0.96	0.05	34.66	0.97	0.03	Fern	26.313	0.867	0.111	23.322	0.790	0.232
Barn	31.21	0.89	0.15	32.11	0.96	0.07	Flower	25.297	0.834	0.120	26.802	0.834	0.226
Church	28.64	0.89	0.16	30.00	0.94	0.09	Fortress	30.215	0.924	0.080	29.530	0.872	0.180
Family	32.02	0.94	0.10	28.43	0.93	0.12	Horns	22.210	0.880	0.123	24.464	0.837	0.237
Francis	32.66	0.91	0.17	32.23	0.92	0.16	Leaves	18.957	0.686	0.217	17.480	0.586	0.305
Horse	33.20	0.96	0.07	21.60	0.80	0.22	Orchids	18.201	0.627	0.205	18.780	0.624	0.259
Ignatius	28.56	0.87	0.18	30.27	0.93	0.09	Room	35.354	0.980	0.040	32.272	0.942	0.114
Museum	30.86	0.92	0.11	34.73	0.97	0.05	Trex	25.007	0.909	0.084	23.828	0.869	0.227
Mean	31.24	0.92	0.12	30.50	0.93	0.10	Mean	25.194	0.838	0.123	24.560	0.794	0.223

on accurate pose initialization, while preserving the supreme runtime speed that makes 3DGS attractive.

Acknowledgements. Christian Schmidt was funded by BMBF project bridgingAI (16DHBKI023). Jens Piekenbrink was funded by Bosch Research as part of the Bosch-RWTH Lighthouse collaboration “Context Understanding for Autonomous Systems”. The authors would like to thank Alexander Hermans for his valuable feedback and discussions.

REFERENCES

- [1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis,” in *ECCV*, 2020.
- [2] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding,” *ACM TOG*, vol. 41, no. 4, pp. 102:1–102:15, July 2022.
- [3] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3D Gaussian Splatting for Real-Time Radiance Field Rendering,” *ACM TOG*, vol. 42, no. 4, July 2023.
- [4] Y. Lin, T. Müller, J. Tremblay, B. Wen, S. Tyree, A. Evans, P. A. Vela, and S. Birchfield, “Parallel Inversion of Neural Radiance Fields for Robust Pose Estimation,” in *ICRA*, 2023.
- [5] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, “iNeRF: Inverting Neural Radiance Fields for Pose Estimation,” in *IROS*, 2021.
- [6] Y. Fu, S. Liu, A. Kulkarni, J. Kautz, A. A. Efros, and X. Wang, “Colmap-free 3d gaussian splatting,” *arXiv:2312.07504*, 2023.
- [7] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, “BARF: Bundle-Adjusting Neural Radiance Fields,” in *ICCV*, 2021.
- [8] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, “NeRF—: Neural Radiance Fields Without Known Camera Parameters,” *arXiv:2102.07064*, 2021.
- [9] W. Bian, Z. Wang, K. Li, J.-W. Bian, and V. A. Prisacariu, “Nope-nerf: Optimising neural radiance field with no pose prior,” in *CVPR*, 2023, pp. 4160–4169.
- [10] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, “Local light field fusion: Practical view synthesis with prescriptive sampling guidelines,” *ACM TOG*, vol. 38, no. 4, pp. 1–14, 2019.
- [11] S.-F. Chng, S. Ramasinghe, J. Sherrah, and S. Lucey, “Gaussian activated neural radiance fields for high fidelity reconstruction and pose estimation,” in *ECCV*, 2022.
- [12] H. Heo, T. Kim, J. Lee, J.-W. Lee, S. Kim, H. Kim, and J.-H. Kim, “Robust Camera Pose Refinement for Multi-Resolution Hash Encoding,” *arXiv:2302.01571*, 2023.
- [13] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, “Nerf++: Analyzing and improving neural radiance fields,” *arXiv:2010.07492*, 2020.
- [14] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, “The Replica dataset: A digital replica of indoor spaces,” *arXiv:1906.05797*, 2019.
- [15] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, “Tanks and temples: Benchmarking large-scale scene reconstruction,” *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.
- [16] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, “Tensorf: Tensorial radiance fields,” in *ECCV*. Springer, 2022, pp. 333–350.
- [17] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann, “Point-nerf: Point-based neural radiance fields,” in *CVPR*, 2022, pp. 5438–5448.
- [18] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, “Plenoxels: Radiance Fields without Neural Networks,” in *CVPR*, 2022, pp. 5501–5510.
- [19] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, “Ewa volume splatting,” in *Visualization*. IEEE, 2001, pp. 29–36.
- [20] —, “Surface splatting,” in *SIGGRAPH*, 2001, pp. 371–378.
- [21] L. Ren, H. Pfister, and M. Zwicker, “Object space ewa surface splatting: A hardware accelerated approach to high quality point rendering,” in *Computer Graphics Forum*, vol. 21, no. 3, 2002, pp. 461–470.
- [22] C. Lassner and M. Zollhofer, “Pulsar: Efficient sphere-based neural rendering,” in *CVPR*, 2021, pp. 1440–1449.
- [23] G. Avraham, J. Straub, T. Shen, T.-Y. Yang, H. Germain, C. Sweeney, V. Balntas, D. Novotny, D. DeTone, and R. Newcombe, “Nerfels: Renderable neural codes for improved camera pose estimation,” in *CVPR*, 2022, pp. 5061–5070.
- [24] F. Li, S. R. Vutukur, H. Yu, I. Shugurov, B. Busam, S. Yang, and S. Ilic, “Nerf-pose: A first-reconstruct-then-regress approach for weakly-supervised 6d object pose estimation,” in *ICCV*, 2023, pp. 2123–2133.
- [25] Y. Sun, X. Wang, Y. Zhang, J. Zhang, C. Jiang, Y. Guo, and F. Wang, “icomma: Inverting 3d gaussians splatting for camera pose estimation via comparing and matching,” *arXiv preprint arXiv:2312.09031*, 2023.
- [26] J. L. Schonberger and J.-M. Frahm, “Structure-from-motion Revisited,” in *CVPR*, 2016, pp. 4104–4113.
- [27] S. Liu, S. Lin, J. Lu, S. Saha, A. Supikov, and M. Yip, “BAA-NGP: Bundle-Adjusting Accelerated Neural Graphics Primitives,” *arXiv:2306.04166*, 2023.
- [28] B.-Y. Cheng, W.-C. Chiu, and Y.-L. Liu, “Improving Robustness for Joint Optimization of Camera Poses and Decomposed Low-Rank Tensorial Radiance Fields,” in *AAAI*, 2024.
- [29] J. Sola, J. Deray, and D. Atchuthan, “A micro lie theory for state estimation in robotics,” *arXiv:1812.01537*, 2018.
- [30] Z. Teed and J. Deng, “Tangent space backpropagation for 3d transformation groups,” in *CVPR*, 2021, pp. 10338–10347.
- [31] E. Sucar, S. Liu, J. Ortiz, and A. Davison, “iMAP: Implicit mapping and positioning in real-time,” in *ICCV*, 2021.
- [32] R. Ranftl, A. Bochkovskiy, and V. Koltun, “Vision transformers for dense prediction,” in *ICCV*, 2021, pp. 12179–12188.
- [33] Y. Jeong, S. Ahn, C. Choy, A. Anandkumar, M. Cho, and J. Park, “Self-calibrating neural radiance fields,” in *ICCV*, 2021.