
CRIM-GS: Continuous Rigid Motion-Aware Gaussian Splatting from Motion Blur Images

Jungho Lee¹Donghyeong Kim¹Dogyoon Lee¹Suhwan Cho¹Sangyoun Lee¹¹School of Electrical and Electronic Engineering, Yonsei University

{2015142131, 2donghyung87, nemotio, chosuhwan, syleeee}@yonsei.ac.kr

Abstract

Neural radiance fields (NeRFs) have received significant attention due to their high-quality novel view rendering ability, prompting research to address various real-world cases. One critical challenge is the camera motion blur caused by camera movement during exposure time, which prevents accurate 3D scene reconstruction. In this study, we propose continuous rigid motion-aware gaussian splatting (CRIM-GS) to reconstruct accurate 3D scene from blurry images with real-time rendering speed. Considering the actual camera motion blurring process, which consists of complex motion patterns, we predict the continuous movement of the camera based on neural ordinary differential equations (ODEs). Specifically, we leverage rigid body transformations to model the camera motion with proper regularization, preserving the shape and size of the object. Furthermore, we introduce a continuous deformable 3D transformation in the $SE(3)$ field to adapt the rigid body transformation to real-world problems by ensuring a higher degree of freedom. By revisiting fundamental camera theory and employing advanced neural network training techniques, we achieve accurate modeling of continuous camera trajectories. We conduct extensive experiments, demonstrating state-of-the-art performance both quantitatively and qualitatively on benchmark datasets. The project page is publicly available at <https://Jho-Yonsei.github.io/CRIM-Gaussian/>.

1 Introduction

Novel view synthesis has recently garnered significant attention, with Neural Radiance Fields (NeRFs) [28] making considerable advancements in photo-realistic neural rendering. NeRFs take sharp 2D images from multiple views as input to accurately reconstruct 3D scenes, which are crucial for applications such as augmented reality (AR) and virtual reality (VR). NeRFs model radiance and volume density through an implicit neural representation using multi-layer perceptrons (MLPs), based on the coordinates and viewing directions of 3D points. However, the computational complexity of this volume rendering process poses challenges for real-time applications. To address this, 3D Gaussian Splatting (3D-GS) [14] has recently emerged, offering an alternative by explicitly representing 3D scenes and enabling high-quality real-time rendering through a differentiable splatting method.

However, accurately reconstructing 3D scenes in real-world scenarios requires accounting for various types of image degradation (*e.g.*, camera motion blur, image defocus blur). Both NeRFs and 3D-GS rely on sharp images as input, which assumes highly ideal conditions. Obtaining such precise images necessitates a large depth of field (DoF), which in turn requires a very small aperture setting. A smaller aperture significantly reduces the amount of incoming light, leading to necessarily longer

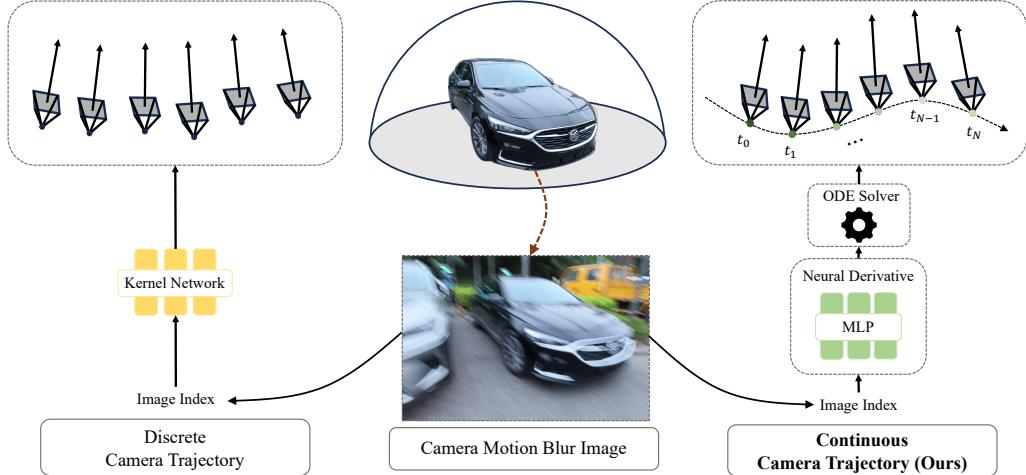


Figure 1: **Differences between CRIM-GS and Existing Methods.** Existing methods obtain discrete blurring kernels through a kernel network composed of MLPs without considering time-continuous dynamics. In contrast, we interpret time-continuous dynamics using neural ODEs and obtain continuous camera motion trajectories over time.

exposure times. This prolonged exposure results in inevitable camera movement during the exposure time. The longer the exposure time, the more complex the camera movement becomes, resulting in images that contain complex motion blur. Consequently, it is critical to develop methods that can handle camera motion blur in input images to achieve precise 3D scene reconstruction. Our work addresses this challenge by aiming to reconstruct accurate 3D scenes from images affected by camera motion blur, thus broadening the applicability of neural rendering to more realistic scenarios.

Recently, several methods have been proposed to achieve sharp novel view rendering from images with camera motion blur. Inspired by traditional blind image deblurring techniques, Deblur-NeRF [25] firstly introduces a method for deblurring 3D scene. This method designs a learnable kernel which intentionally blur images and, during rendering, excludes the learned kernel to render sharp novel view images. Following this approach, various methods have been emerged to more accurately estimate the blurring kernel. Additionally, with the advent of 3D-GS, real-time renderable deblurring methods have developed. Notably, SMURF [19] addresses actual camera motion blur by sequentially modeling camera movement during exposure time, designing continuous camera trajectories. However, SMURF limits the continuous camera movement to the x and y axes of the 2D image plane, which restricts its ability to represent motion in the 3D space comprehensively.

In this paper, we propose a novel approach with three key contributions. First, we model rigid body transformations, as used in Nerfies [32] and DP-NeRF [18], to maintain the shape and size of the subject during camera movement. Second, recognizing that real-world images contain complex and non-linear camera motion information, we introduce a learnable deformable 3D transformation in the $SE(3)$ field to account for subject distortion and refine the rigid body transformation. Third, we apply neural ordinary differential equations (ODEs) [3] to model the continuous camera movement during exposure time as Fig. 1, integrating the two aforementioned transformation methods. In other words, by continuously modeling the camera trajectory in 3D rather than 2D, we present a model that is fundamentally different from existing methods, namely continuous rigid motion-aware gaussian splatting (CRIM-GS). Our model ensures real-time rendering speed by employing 3D-GS [14], a differentiable rasterization-based method, instead of ray-based methods (e.g., NeRF [28], TensoRF [2]). We evaluate and compare our approach on Deblur-NeRF [25] synthetic and real-world scene datasets, achieving state-of-the-art results across the benchmarks. To demonstrate the effectiveness of CRIM-GS, we conduct various ablative experiments on the proposed contributions.

2 Related Work

2.1 Image Deblurring

In the field of computer vision, the phenomenon known as blur is identified as a degradation in images. It frequently occurs during the image capture process in the camera due to factors such as camera shaking, depth of field, and the movement of objects. These factors result in specific types of blur, namely camera motion blur, defocus blur, and object motion blur. Traditionally, image deblurring is simulated as a finding specific blur kernel that restores the degradation of the target image exploiting the knowledge about the blurring process. The blurring process is mathematically formulated as a convolutional equation, denoted as $b = K * s + n$, where b , K , s , and n indicate the blurred image, blur kernel, clean image, and additional noise, respectively. Disentangling this equation involves a categorization into non-blind and blind deblurring, depending on whether the blur kernel K is known or not. Our focus lies on the latter, as real-world images typically lack knowledge of the blur kernel.

Recently, attempts to solve the blind deblurring using deep learning have been explored utilizing data-driven prior involved in paired datasets with network optimization. According to recent deep image deblurring survey [63], several works adopt various network architectures as deblurring networks, such as deep auto-encoder, multi-scale networks, and generative adversarial networks (GANs) [10]. [46, 30] address deep auto-encoder based on U-Net [41] architecture with additional priors. To utilize different scales of the image as complementary information, [49, 9, 61] adopt the multi-scale network and progressively generate high-resolution sharp images utilizing restoration from low-resolution images. Other approaches [15, 16] take GANs to restore the high-frequency details exploiting the high-quality image-to-image translation power of GANs. Although image deblurring shows high-quality restored images, the methods cannot be directly applied to the neural rendering framework as [25] argued since the inconsistency across the given images derived from inherent ill-posed property.

2.2 Neural Rendering

Neural rendering has seen explode in the fields of computer graphics and vision area thanks to the emergence of volumetric rendering and ray tracing-based neural radiance fields (NeRFs) [28], which provide super-realistic rendering quality from 3D scenes. NeRFs have led to a wide range of studies, including 3D mesh reconstruction [53, 52, 21, 48], dynamic scene [39, 32, 33, 20, 50, 22], and human avatars [56, 38, 13]. At the same time, several researches have aimed at overcoming NeRF’s slow training and rendering speed, such as TensoRF [2], plenoxel [8], and Plenoctree [60]. Among these advancements, the recent emergence of 3D Gaussian splatting (3D-GS) [14], which offers remarkable performance along with fast training and rendering speed, has further accelerated research in the neural rendering field. In addition to the aforementioned diverse scenarios for 3D scene modeling, there has been active research focusing on the external non-ideal conditions of given images, such as sparse-view images [29, 58, 51, 57], and the absence of camera parameters [12, 55, 1]. Moreover, there has been significant research attention on non-ideal conditions inherent to the images themselves, such as low-light [27, 35], blur [25, 18, 36, 54, 19, 4, 64, 37], and inconsistent appearance condition [26]. Recently, neural rendering from blurred images, which can easily occur during image acquisition, has attracted attention due to its practical applicability.

2.3 Neural Rendering from Blurry Images

Deblur-NeRF [25] firstly introduced the deblurred neural radiance fields by importing the blind-deblurring mechanism into the NeRF framework. They introduce specific blur kernel in front of the NeRF framework imitating the blind deblurring in 2D image deblurring area. After the emergence of [25], several attempts have been proposed to model the precise blur kernel with various types of neural rendering baseline, such as TensoRF [2], and 3D-GS [14]. DP-NeRF [18] proposes rigid blur kernel that predict the camera motion during image acquisition process as 3D rigid body motion to preserve the geometric consistency across the scene. BAD-NeRF [54] and BAD-GS [64] similarly predict blur kernel as camera motions based on NeRFs [28] and 3D-GS [14], but they assume the linear motion and interpolate them between predicted initial and final camera poses. PDRF [36] proposes progressive blur estimation model with hybrid 2-stage efficient rendering scheme that consists of coarse ray and fine voxel renderer. SMURF [19] adopt Neural ODE [3] to predict more precise continuous camera motions based on TensorRF [2]. BAGS [37] proposes CNN-based multi-scale blur-agnostic degradation kernel with blur masking that indicates the blurred areas.

3 Preliminary

3.1 3D Gaussian Splatting

3D Gaussian Splatting (3D-GS) [14] represents explicit 3D scenes as point clouds using numerous differentiable 3D Gaussians. Each 3D Gaussian primitive is defined by a mean vector $\mu \in \mathbb{R}^3$ and a covariance matrix $\Sigma \in \mathbb{R}^{3 \times 3}$. The basic 3D Gaussian \mathbf{G} is geometrically represented as follows:

$$\mathbf{G}(x) = e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}, \quad (1)$$

where x denotes an arbitrary 3D point. The covariance matrix Σ is decomposed into a scaling vector $\mathbf{S} \in \mathbb{R}^3$ and a rotation matrix $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ obtained by a quaternion $q \in \mathbb{R}^4$ as $\Sigma = \mathbf{R}\mathbf{S}\mathbf{S}^T\mathbf{R}^T$. To render novel view images, the 3D Gaussians in world space are projected into 2D camera space to employ differentiable splatting [59]. Then, the covariance matrix Σ' in camera space is expressed by a viewing transform \mathbf{W} and the Jacobian \mathbf{J} of the affine approximation of the projective transformation as $\Sigma' = \mathbf{J}\mathbf{W}\Sigma\mathbf{W}^T\mathbf{J}^T$.

The Gaussian primitive also includes an opacity value $\alpha \in \mathbb{R}$ and spherical harmonic (SH) coefficients $\mathcal{C} \in \mathbb{R}^k$ for the color value. The color C of pixel p is computed by blending \mathcal{N} ordered points overlapping the pixel as follows:

$$C(p) = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (2)$$

where c_i and α_i denote the color and the density of each point, respectively, obtained by a 3D Gaussian with covariance Σ multiplied by SH color coefficients and a learned per-point opacity. In this paper, we ensure real-time rendering by employing the framework of 3D-GS.

3.2 Neural Ordinary Differential Equations

Neural ODEs [3] are first proposed as an approach that interprets neural networks as the derivatives of a ODE systems, where ODEs represent the dynamics inherent to the hidden states. Specifically, neural ODEs are utilized to represent parameterized, time-continuous dynamics in the latent space, providing a unique solution given an initial value and numerical differential equation solvers [23]. Recently, there has been extensive research on neural ODEs. For instance, Latent-ODEs [42] model the continuous dynamics of irregularly sampled time-series data, while Vid-ODEs [34] model a smooth latent space by capturing the underlying continuous dynamics of videos.

Neural ODEs model a continuous and differentiable latent state $\mathbf{z}(t)$. Within an infinitesimally small step limit ϵ in the latent space, the local continuous dynamics are modeled as $\mathbf{z}(t+\epsilon) = \mathbf{z}(t) + \epsilon \cdot \frac{d\mathbf{z}(t)}{dt}$. The derivative of the latent state, $\frac{d\mathbf{z}(t)}{dt}$, is represented by a neural network $f(\mathbf{z}(t), t; \phi)$ parameterized by learnable parameters ϕ . The latent state at any arbitrary time t_s is obtained by solving the ODE from the initial time t_0 :

$$\mathbf{z}(t_s) = \mathbf{z}(t_0) + \int_{t_0}^{t_s} f(\mathbf{z}(t), t; \phi) dt = \text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_s, \phi). \quad (3)$$

The simplest method to solve ODEs is the Euler method [7], which is a fixed-step-size first-order solver. Additionally, the Runge-Kutta [17] methods are preferred as a higher-order solvers as they offer enhanced convergence and stability. Therefore, we adopt the Dormand-Prince-Shampine fifth-order [6] Runge-Kutta solver for all experiments, following [3].

The derivative f modeled by the neural network is expressed as a uniformly Lipschitz continuous non-linear function in \mathbf{z} and t [11]. Therefore, the solution obtained through the solver for any given integration interval (t_i, t_j) is always unique in the integration of the continuous dynamics. We model the non-linear 3D camera trajectory as time-continuous using neural ODEs in the latent space to ensure a continuous representation.

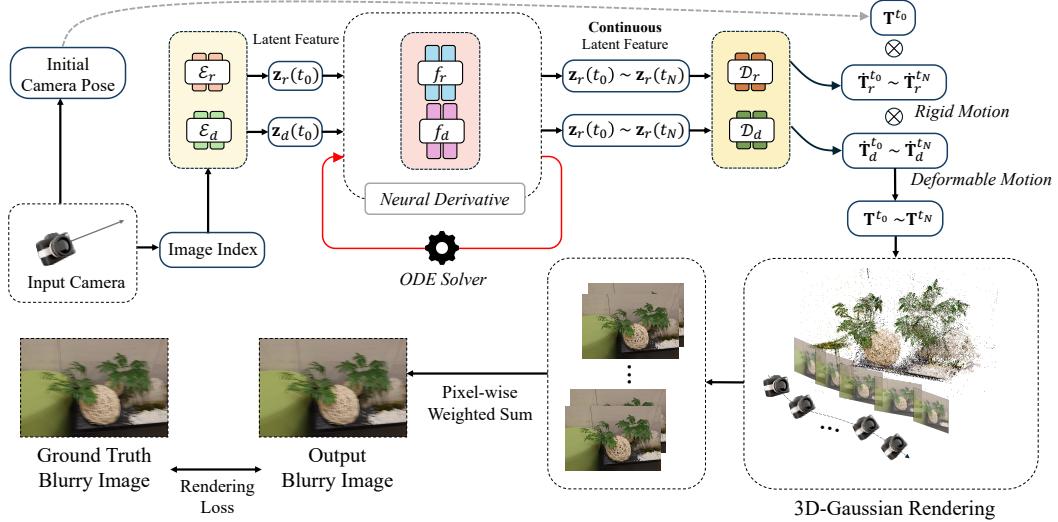


Figure 2: **Pipeline of CRIM-GS.** The input camera contains the image index and the initial camera pose information. CRIM-GS iteratively solves the ODE using the encoded image index and the neural derivative presented in Sec. 4.2 and Sec. 4.3, and obtains N transformed camera poses after decoding. Then, N images are rendered through 3D-GS, and a blurry image is obtained through the pixel-wise weighted-sum presented in Sec. 4.4

4 Method

4.1 CRIM-GS Framework

Our goal is to reconstruct a sharp 3D scene using only images with camera motion blur as input, thereby obtaining deblurred novel view images. Inspired by image blind deblurring methods, we follow the approach of Deblur-NeRF [25], intentionally learning a kernel that blurs images and excluding this kernel during rendering to produce sharp rendered images. As illustrated in Fig. 1, our blurring kernel consists of camera poses along the camera motion trajectory, generated continuously in time order through neural ODEs [3]. Each pose is composed of a rigid body transformation $\dot{\mathbf{T}}_r = [\dot{\mathbf{R}}_r | \dot{\mathbf{t}}_r]$ (Sec. 4.2), which maintains the shape and size of the subject, and a deformable body transformation $\dot{\mathbf{T}}_d = [\dot{\mathbf{R}}_d | \dot{\mathbf{t}}_d]$ (Sec. 4.3), which accounts for distortions that may occur during actual image acquisition. Both transformations exist in a dense $SE(3)$ field, where $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ and $\mathbf{t} \in \mathbb{R}^3$ represent multiplicative rate of change of the rotation matrix and translation vector, respectively. Given these two transformation matrices $[\dot{\mathbf{R}}_r | \dot{\mathbf{t}}_r]$ and $[\dot{\mathbf{R}}_d | \dot{\mathbf{t}}_d]$, the subsequent pose $\mathbf{T}^{t_s} \in SE(3)$ in the camera motion is derived as follows:

$$\mathbf{T}^{t_s} = \mathbf{T}^{t_0} \dot{\mathbf{T}}_r^{t_s} \dot{\mathbf{T}}_d^{t_s}, \text{ where } \mathbf{T} = [\mathbf{R} | \mathbf{t}] = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \in SE(3). \quad (4)$$

$\mathbf{T}^{t_0} = [\mathbf{R}^{t_0} | \mathbf{t}^{t_0}]$ and $\mathbf{T}^{t_s} = [\mathbf{R}^{t_s} | \mathbf{t}^{t_s}]$ denote the camera poses at the initial state and the time t_s . By rendering N images from the obtained N camera poses and computing their pixel-wise weighted sum (Sec. 4.4), we obtain the final blurry image. Our whole framework is shown in Fig. 2

4.2 Continuous Rigid Body Motion

According to [24], using rigid body transformations in the $SE(3)$ field is effective when the shape and size of the object remain unchanged. Assuming no distortion in the images during camera motion, we first obtain the unit rotation axis $\hat{\omega} \in \mathbb{R}^3$ and the rotation angle θ about this axis, as well as the 3D column vector $v \in \mathbb{R}^3$ required for translation. These are the components of the unit screw axis $\mathcal{S} = (\hat{\omega}, v)$. Considering that the form of blur varies for each image in a scene, we embed the image index of the scene to obtain different unit screw axis \mathcal{S} for each image. The embedded features are then passed through a parameterized one-layer encoder \mathcal{E}_r , transforming them into the latent state $\mathbf{z}_r(t_0)$, which represents the latent features for $\hat{\omega}$, θ , and v of \mathcal{S} . Inspired by the continuity of the

screw axis in rigid body motion [24], we continuously model this screw axis in the latent space using a neural derivative $f(\cdot, \cdot; \phi)$. Specifically, the derivative of the latent features of the screw axis is expressed as $\frac{d\mathbf{z}_r(t)}{dt} = f(\mathbf{z}_r(t), t; \phi)$, and the latent features at an arbitrary time t_s can be obtained by numerically integrating f from t_0 to t_s :

$$\mathbf{z}_r(t_s) = \mathbf{z}_r(t_0) + \int_{t_0}^{t_s} f(\mathbf{z}_r(t), t; \phi) dt. \quad (5)$$

To obtain N poses along the camera trajectory, we uniformly sample N time points within the given exposure time from the image metadata. We then transform the N latent features obtained through the ODE solver into the unit screw axis \mathcal{S} using a simple MLP decoder \mathcal{D}_r . In Nerfies [32] and DP-NeRF [18], the angular velocity $\omega = \hat{\omega}\theta$ is modeled and then decomposed into the unit rotation axis $\hat{\omega}$ and the rotation angle θ , making the two elements dependent on each other. However, since θ represents the rotation amount about the axis, these two elements should be independent. Therefore, we model $\hat{\omega}$ and θ independently through the decoder \mathcal{D}_r :

$$\mathcal{D}_r(\mathbf{z}(t_s)) = (\hat{\omega}^{t_s}, \theta^{t_s}, v^{t_s}), \text{ where } \|\hat{\omega}^{t_s}\| = 1. \quad (6)$$

According to [24], the screw axis $\mathcal{S}^{t_s} = (\hat{\omega}^{t_s}, v^{t_s})$ is a normalized twist, so the transformation matrix $[\mathcal{S}^{t_s}] \in \mathbb{R}^{4 \times 4}$ is represented as follows:

$$[\mathcal{S}^{t_s}] = \begin{bmatrix} [\hat{\omega}^{t_s}] & v^{t_s} \\ 0 & 0 \end{bmatrix} \in \mathfrak{se}(3), \quad (7)$$

where $[\hat{\omega}^{t_s}] \in \mathfrak{so}(3)$ is a 3×3 skew-symmetric matrix of vector $\hat{\omega}^{t_s}$. To derive the infinitesimal transformation matrix $[\mathcal{S}^{t_s}]\theta^{t_s} \in \mathfrak{se}(3)$ on the Lie algebra to the transformation matrix $\dot{\mathbf{T}}_r^{t_s} \in SE(3)$ in the Lie group, we use the matrix exponential $e^{[\mathcal{S}^{t_s}]\theta^{t_s}}$, whose rotation matrix and translation vector are $\dot{\mathbf{R}}_r^{t_s} = e^{[\hat{\omega}^{t_s}]\theta^{t_s}}$ and $\dot{\mathbf{t}}_r^{t_s} = G(\theta^{t_s})v^{t_s}$, respectively. These matrices are expressed as follows using Rodrigues' formula [40] through Taylor expansion:

$$e^{[\hat{\omega}^{t_s}]\theta^{t_s}} = I + \sin \theta^{t_s} [\hat{\omega}^{t_s}] + (1 - \cos \theta^{t_s}) [\hat{\omega}^{t_s}]^2 \in SO(3) \quad (8)$$

$$G(\theta^{t_s}) = I\theta^{t_s} + (1 - \cos \theta^{t_s}) [\hat{\omega}^{t_s}] + (\theta^{t_s} - \sin \theta^{t_s}) [\hat{\omega}^{t_s}]^2. \quad (9)$$

Through the whole process, we obtain N continuous rigid body transformation matrices $\dot{\mathbf{T}}_r = e^{[\mathcal{S}]\theta}$, and by multiplying these with the input pose as in Eq. (4), we obtain the transformed poses.

4.3 Continuous Deformable Body Motion

In real-world scenarios, camera motion blur during exposure time does not always preserve the shape and size of objects. Especially when the camera moves along a nonlinear path, its complex trajectory may not be sufficiently explained by rigid body transformations alone [44]. Based on this assumption, we propose a deformable body transformation to provide additional corrections to the rigid body transformation. This transformation has more degrees of freedom from a learning perspective and is relatively simple to implement. The deformable body transformation optimizes the transformation matrix in $SE(3)$ directly without converting $\hat{\omega} \in \mathbb{R}^3$ into a 3×3 skew-symmetric matrix as described in Eq. (7). Thus, instead of modeling $\hat{\omega}$ and v , this transformation designs the rotation matrix $\dot{\mathbf{R}}$ and the translation vector $\dot{\mathbf{t}}$. This process begins by encoding the image index into the latent state $\mathbf{z}_d(t_0)$ of the deformable $\dot{\mathbf{R}}$ and $\dot{\mathbf{t}}$ using the encoder \mathcal{E}_d , similar to Sec. 4.2. The latent state $\mathbf{z}_d(t_s)$ at any arbitrary time t_s is obtained using the neural derivative g parameterized by ψ and a solver, as shown in Eq. (5):

$$\mathbf{z}_d(t_s) = \mathbf{z}_d(t_0) + \int_{t_0}^{t_s} g(\mathbf{z}_d(t), t; \psi) dt. \quad (10)$$

These latent features are then transformed into the rotation matrix $\dot{\mathbf{R}}_d$ and translation vector $\dot{\mathbf{t}}^{t_s}$ through a simple MLP decoder \mathcal{D}_d :

$$\mathcal{D}_d(\mathbf{z}_d(t_s)) = (\tilde{\mathbf{R}}_d^{t_s}, \mathbf{t}^{t_s}) \rightarrow \dot{\mathbf{T}}_d^{t_s} = \begin{bmatrix} \tilde{\mathbf{R}}_d^{t_s} + \mathbf{I} & \mathbf{t}^{t_s} \\ 0 & 1 \end{bmatrix} \in SE(3), \quad (11)$$

where $\tilde{\mathbf{R}}_d^{t_s} + \mathbf{I} = \dot{\mathbf{R}}_d^{t_s}$ and \mathbf{I} is the identity matrix. The transformation matrix $\dot{\mathbf{T}}_d^{t_s}$ is modeled to satisfy the conditions of the $SE(3)$ field. Additionally, since the deformable body transformation is intended to correct the rigid body transformation, it should not significantly affect the rigid body transformation during the initial stages of training. Therefore, we initialize the weights of the decoder to be close to the identity matrix using a uniform distribution $\mathcal{U}(-10^{-5}, 10^{-5})$. Furthermore, to ensure that the rotation matrix $\dot{\mathbf{R}}_d$ satisfies the main conditions of the $SE(3)$ matrix during training, we introduce two regularization losses. The first, $\mathcal{L}_{\text{ortho}}$, ensures the orthogonality condition: $\mathcal{L}_{\text{ortho}} = \|\dot{\mathbf{R}}_d \dot{\mathbf{R}}_d^T - \mathbf{I}\|_2$. The second, \mathcal{L}_{det} , ensures the determinant condition: $\mathcal{L}_{\text{det}} = \|\det(\dot{\mathbf{R}}_d^{t_s}) - 1\|_2$. These regularizations ensure that the transformation matrix $\dot{\mathbf{T}}_d^{t_s}$ lies within the $SE(3)$ field. Finally, we apply Eq. (5) using $\dot{\mathbf{T}}_d^{t_s}$ and $\dot{\mathbf{T}}_r^{t_s}$ obtained from Sec. 4.2 to get the refined transformed camera poses.

4.4 Optimization

Pixel-wise Weight. Once the N images along the camera motion trajectory are obtained from the N camera poses, we apply a pixel-wise weighted sum to create the blurred image $\mathcal{I}_{\text{blur}}$, following previous research [25, 18, 19, 36]. We use a shallow CNN \mathcal{F} and a softmax function to compute the pixel-wise weights $\mathcal{P} \in \mathbb{R}^{N \times H \times W \times 3}$ for the resulting images, as follows:

$$\mathcal{I}_{\text{blur}} = \sum_{i=1}^N \mathcal{I}_i \cdot \mathcal{P}_i, \quad \text{where } \mathcal{P}_i = \text{softmax}(\mathcal{F}(\mathcal{I}_i)), \quad (12)$$

where \mathcal{I}_i is the i -th image along the camera motion, and \mathcal{P}_i is the pixel-wise weight of that image. This method allows us to obtain the final blurred image, which is then optimized against the ground truth blurred image using a loss function.

Objective. We optimize the learning process using the \mathcal{L}_1 loss and D-SSIM between the generated blurred image and the ground truth blurred image, similar to 3D-GS [14]. The \mathcal{L}_1 loss ensures pixel-wise accuracy, while D-SSIM captures perceptual differences. Additionally, we apply the regularization losses \mathcal{L}_{det} and $\mathcal{L}_{\text{ortho}}$ mentioned in Sec. 4.3 to regularize the deformable transformation. The final objective \mathcal{L} is defined as follows:

$$\mathcal{L} = (1 - \lambda_c)\mathcal{L}_1 + \lambda_c \mathcal{L}_{\text{D-SSIM}} + \lambda_{\text{det}} \mathcal{L}_{\text{det}} + \lambda_{\text{ortho}} \mathcal{L}_{\text{ortho}}, \quad (13)$$

where λ_c is a factor for balancing \mathcal{L}_1 and $\mathcal{L}_{\text{D-SSIM}}$, and λ_{det} and λ_{ortho} are relatively small factors for the regularization losses.

5 Experiments

Datasets. CRiM-GS is evaluated using the camera motion blur dataset provided by Deblur-NeRF [25], which includes 5 synthetic scenes and 10 real-world scenes. The synthetic scenes are generated using Blender [5], and each single image that constitutes a motion-blurred image is obtained through linear interpolation between the first and last camera poses along the camera path. The obtained images are combined in linear RGB space to create the final blurred images. The real-world scenes consist of images captured with a CANON EOS RP camera, where the exposure time is manually set to produce blurry images. For each scene, all images exhibit different types of blur, including non-uniform blur. Additionally, for both synthetic and real-world scenes, we use COLMAP [43, 45] to obtain the camera poses for each image and the initial point cloud for training the Gaussian primitives.

Implementation Details. CRiM-GS is trained for 40k iterations, with all settings for learning the Gaussian primitives, including the learning rate, being identical to those of 3D-GS. We set the number of poses N that constitute the continuous camera trajectory to 9. The size of the hidden state is fixed at 64 for all processes involving rigid body transformation and deformable body transformation. Both neural derivatives f and g are composed very simply, with single hidden layer and a ReLU activation function. The CNN \mathcal{F} consists of three layers with 64 channels each. \mathcal{F} starts training after 3k iterations, allowing the initial camera path to be sufficiently optimized. Additionally, we set λ_c , λ_{det} , and λ_{ortho} to 0.3, 10^{-3} , and 10^{-3} respectively for the objective function. All experiments are conducted on a single NVIDIA RTX 3090 GPU.

Table 1: **Comparisons on synthetic and real-world scene dataset.** We evaluate the performance on three metrics (PSNR, SSIM, LPIPS). “*” denotes the results obtained by reproducing the released code. The orange and yellow cells respectively indicate the highest and second-highest value.

Methods	Synthetic Scene Dataset			Real-World Scene Dataset		
	PSNR(\uparrow)	SSIM(\uparrow)	LPIPS(\downarrow)	PSNR(\uparrow)	SSIM(\uparrow)	LPIPS(\downarrow)
Naive NeRF [28]	23.78	0.6807	0.3362	22.69	0.6347	0.3687
NeRF+MPR [62]	25.11	0.7476	0.2148	23.38	0.6655	0.3140
NeRF+PVD [47]	24.58	0.7190	0.2475	23.10	0.6389	0.3425
Deblur-NeRF [25]	28.77	0.8593	0.1400	25.63	0.7645	0.1820
PDRF-10* [36]	28.86	0.8795	0.1139	25.90	0.7734	0.1825
BAD-NeRF* [54]	27.32	0.8178	0.1127	22.82	0.6315	0.2887
DP-NeRF [18]	29.23	0.8674	0.1184	25.91	0.7751	0.1602
DeblurGS* [31]	20.22	0.5454	0.1042	20.48	0.5813	0.1186
BAD-GS* [64]	21.12	0.5852	0.1007	20.82	0.5917	0.1000
BAGS [37]	27.34	0.8353	0.1116	26.70	0.8237	0.0956
CRIM-GS	30.19	0.9004	0.0485	27.33	0.8310	0.0634

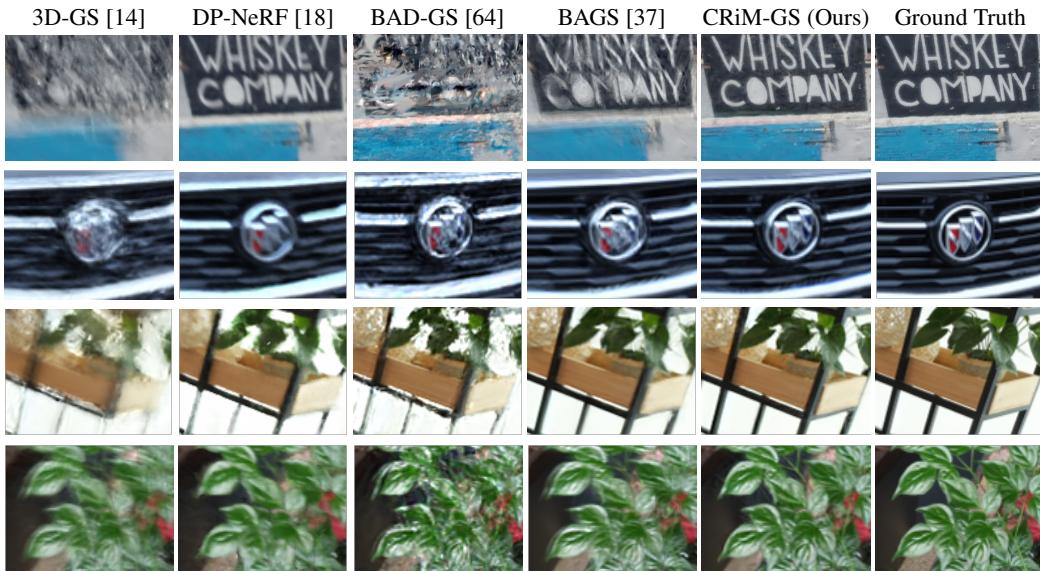


Figure 3: **Qualitative comparison on the synthetic and real-world scenes.** The scenes, from top to bottom, are FACTORY, BUICK, GIRL, and PARTERRE. The FACTORY is a synthetic scene, while the other three are all real-world scenes.

5.1 Novel View Synthesis Results

We evaluate CRIM-GS using three metrics: peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), and learned perceptual image patch similarity (LPIPS). We compare our method with both ray-based and rasterization methods. Note that the results for DeblurGS [31] and BAD-GS [64] were obtained by re-running the released code ourselves. The overall results for all scenes are shown in Tab. 1, demonstrating that our method achieves superior performance compared to all other methods. Specifically for the LPIPS, CRIM-GS shows an improvement of approximately 52% for synthetic scenes and 33% for real-world scenes than the state-of-the-art. Tab. 2 presents the performance for individual scenes in synthetic dataset. Our CRIM-GS consistently achieves high PSNR and SSIM scores on average and demonstrates the best LPIPS performance across all scenes except for the “POOL”. Although DeblurGS and BAD-GS achieve relatively good LPIPS scores, their PSNR and SSIM scores are lower, indicating that their camera poses are not properly optimized. For more details about pose optimization, please refer to our appendix.

To qualitatively evaluate the results, we present rendering outcomes for several scenes in Fig. 3. These include one synthetic scene and three real-world scenes. Our method shows superior qualitative results compared to the state-of-the-art ray-based method DP-NeRF [18], as well as rasterization-based methods such as BAD-GS [64] and BAGS [37]. Benefiting from using 3D-GS as the backbone,

Table 2: **Comparison of performance on the synthetic scenes.** CRiM-GS demonstrates the best performance across all scenes except for the “POOL”, where it still shows notably good performance.

Synthetic Scene	FACTORY			COZYROOM			POOL			TANABATA			TROLLEY		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
Deblur-NeRF [25]	25.60	0.7750	0.2687	32.08	0.9261	0.0477	31.61	0.8682	0.1246	27.11	0.8640	0.1228	27.45	0.8632	0.1363
PDRF-10* [36]	25.87	0.8316	0.1915	31.13	0.9225	0.0439	31.00	0.8583	0.1408	28.01	0.8931	0.1004	28.29	0.8921	0.0931
BAD-NeRF* [54]	24.43	0.7274	0.2134	29.77	0.8864	0.0616	31.51	0.8620	0.0802	25.32	0.8081	0.1077	25.58	0.8049	0.1008
DP-NeRF [18]	25.91	0.7787	0.2494	32.65	0.9317	0.0355	31.96	0.8768	0.0908	27.61	0.8748	0.1033	28.03	0.8752	0.1129
DeblurGS* [31]	17.82	0.3980	0.1453	22.88	0.6758	0.0514	24.53	0.6095	0.1171	17.54	0.4800	0.1132	18.32	0.5638	0.0942
BAD-GS* [64]	16.84	0.3220	0.2425	22.31	0.6954	0.0519	26.70	0.7051	0.0642	20.13	0.6095	0.0729	19.60	0.5941	0.0722
BAGS [37]	22.35	0.6639	0.2277	32.21	0.9359	0.0245	28.72	0.8404	0.0804	26.79	0.8735	0.1099	26.61	0.8627	0.1156
CRiM-GS	27.93	0.8521	0.0693	33.01	0.9409	0.0209	31.61	0.8778	0.0657	29.04	0.9180	0.0376	29.36	0.9131	0.0492

Table 3: **Comparisons on synthetic and real-world scene dataset.** We evaluate the performance on three metrics (PSNR, SSIM, LPIPS).

Methods	Rigid $\dot{\mathbf{T}}_r$	Deformable $\dot{\mathbf{T}}_d$	Pixel-wise Weight	Synthetic Scene Dataset			Real-World Scene Dataset		
				PSNR(↑)	SSIM(↑)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
(A)				21.09	0.5974	0.2774	21.80	0.6135	0.2698
(B)	✓			28.75	0.8644	0.0732	25.95	0.7724	0.0851
(C)	✓		✓	29.87	0.8876	0.0524	27.01	0.8211	0.0759
(D)		✓		27.34	0.8358	0.0789	25.68	0.7588	0.0922
(E)		✓	✓	28.81	0.8706	0.0591	26.93	0.8159	0.0813
(F)	✓	✓		29.33	0.8753	0.0654	26.48	0.8054	0.0715
(G)	✓	✓	✓	30.19	0.9004	0.0485	27.33	0.8310	0.0634

our method ensures fast training and fast rendering speeds (approximately 500 fps). As mentioned earlier, CRiM-GS achieves the best quantitative results and the most perceptually pleasing qualitative results.

5.2 Ablation Study

Ablation on Continuous Transformation. To thoroughly demonstrate the validity and effectiveness of our contributions, we conduct ablation experiments on all scenes, not just a single scene. As shown in Tab. 3, the model (A) with the main component of CRiM-GS, the continuous rigid body transformation, shows a significant performance increase, indicating that the camera motion path is well-modeled. The model (D) with only the continuous deformable body transformation performs worse than the model with only the rigid body transformation, suggesting that the deformable transformation leads to suboptimal results due to its high degrees of freedom. Therefore, the model (F) combining both transformations, which refines camera motion distortion, achieves the best performance without pixel-wise weights, surpassing existing state-of-the-art methods.

Ablation on Pixel-wise Weight. The pixel-wise weight, inspired by traditional ray-based 3D scene deblurring methods [18, 25] and blind image deblurring methods [47, 62], differs from the approach of averaging images that constitute a blurry image. As shown in Tab. 3 for models (C), (E), and (G), applying this method consistently results in higher performance than not applying it. This indicates that by assigning corresponding weights to each pixel location, a more precise blurred image is reconstructed, and the images that constitute the blurry image are modeled more accurately.

6 Conclusion

We propose CRiM-GS, a novel method for reconstructing sharp 3D scenes from blurry images caused by camera motion. We apply a rigid body transformation method under the assumption that the object remains unchanged during camera movement and propose a learnable deformable body transformation method to correct potential distortions of actual camera path. These transformation methods are continuously modeled through neural ODEs in 3D physical space, representing continuous 3D camera motion trajectories. Inspired by traditional image deblurring methods and existing 3D scene deblurring research, we introduce a simple CNN-based pixel-wise weight. Our CRiM-GS outperforms the state-of-the-art in 3D scene deblurring, and we demonstrate the effectiveness of our contributions through extensive experiments that rigorously evaluate the validity of the proposed components.

References

- [1] Bian, W., Wang, Z., Li, K., Bian, J.W., Prisacariu, V.A.: Nope-nerf: Optimising neural radiance field with no pose prior. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4160–4169 (2023)
- [2] Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: European Conference on Computer Vision. pp. 333–350. Springer (2022)
- [3] Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. Advances in neural information processing systems **31** (2018)
- [4] Chen, W., Liu, L.: Deblur-gs: 3d gaussian splatting from camera motion blurred images. Proceedings of the ACM on Computer Graphics and Interactive Techniques **7**(1), 1–15 (2024)
- [5] Community, B.O.: Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam (2018), <http://www.blender.org>
- [6] Dormand, J.R., Prince, P.J.: A family of embedded runge-kutta formulae. Journal of computational and applied mathematics **6**(1), 19–26 (1980)
- [7] Euler, L.: Institutionum calculi integralis, vol. 4. impensis Academiae imperialis scientiarum (1845)
- [8] Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5501–5510 (2022)
- [9] Gao, H., Tao, X., Shen, X., Jia, J.: Dynamic scene deblurring with parameter selective sharing and nested skip connections. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 3848–3856 (2019)
- [10] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)
- [11] Ince, E.L.: Ordinary differential equations. Courier Corporation (1956)
- [12] Jeong, Y., Ahn, S., Choy, C., Anandkumar, A., Cho, M., Park, J.: Self-calibrating neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5846–5854 (2021)
- [13] Jiang, W., Yi, K.M., Samei, G., Tuzel, O., Ranjan, A.: Neuman: Neural human radiance field from a single video. In: European Conference on Computer Vision. pp. 402–418. Springer (2022)
- [14] Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4), 1–14 (2023)
- [15] Kupyn, O., Budzan, V., Mykhailych, M., Mishkin, D., Matas, J.: Deblurgan: Blind motion deblurring using conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8183–8192 (2018)
- [16] Kupyn, O., Martyniuk, T., Wu, J., Wang, Z.: Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8878–8887 (2019)
- [17] Kutta, W.: Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. Teubner (1901)
- [18] Lee, D., Lee, M., Shin, C., Lee, S.: Dp-nerf: Deblurred neural radiance field with physical scene priors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12386–12396 (2023)

- [19] Lee, J., Lee, D., Lee, M., Kim, D., Lee, S.: Smurf: Continuous dynamics for motion-deblurring radiance fields. arXiv preprint arXiv:2403.07547 (2024)
- [20] Li, T., Slavcheva, M., Zollhoefer, M., Green, S., Lassner, C., Kim, C., Schmidt, T., Lovegrove, S., Goesele, M., Newcombe, R., et al.: Neural 3d video synthesis from multi-view video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5521–5531 (2022)
- [21] Li, Z., Müller, T., Evans, A., Taylor, R.H., Unberath, M., Liu, M.Y., Lin, C.H.: Neuralangelo: High-fidelity neural surface reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8456–8465 (2023)
- [22] Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6498–6508 (2021)
- [23] Lindelöf, E.: Sur l’application de la méthode des approximations successives aux équations différentielles ordinaires du premier ordre. Comptes rendus hebdomadaires des séances de l’Académie des sciences **116**(3), 454–457 (1894)
- [24] Lynch, K.M., Park, F.C.: Modern robotics. Cambridge University Press (2017)
- [25] Ma, L., Li, X., Liao, J., Zhang, Q., Wang, X., Wang, J., Sander, P.V.: Deblur-nerf: Neural radiance fields from blurry images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12861–12870 (2022)
- [26] Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7210–7219 (2021)
- [27] Mildenhall, B., Hedman, P., Martin-Brualla, R., Srinivasan, P.P., Barron, J.T.: Nerf in the dark: High dynamic range view synthesis from noisy raw images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16190–16199 (2022)
- [28] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I. pp. 405–421 (2020)
- [29] Niemeyer, M., Barron, J.T., Mildenhall, B., Sajjadi, M.S., Geiger, A., Radwan, N.: Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5480–5490 (2022)
- [30] Nimisha, T.M., Kumar Singh, A., Rajagopalan, A.N.: Blur-invariant deep learning for blind-deblurring. In: Proceedings of the IEEE international conference on computer vision. pp. 4752–4760 (2017)
- [31] Oh, J., Chung, J., Lee, D., Lee, K.M.: Deblurgs: Gaussian splatting for camera motion blur. arXiv preprint arXiv:2404.11358 (2024)
- [32] Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5865–5874 (2021)
- [33] Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. arXiv preprint arXiv:2106.13228 (2021)
- [34] Park, S., Kim, K., Lee, J., Choo, J., Lee, J., Kim, S., Choi, E.: Vid-ode: Continuous-time video generation with neural ordinary differential equation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 2412–2422 (2021)
- [35] Pearl, N., Treibitz, T., Korman, S.: Nan: Noise-aware nerfs for burst-denoising. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12672–12681 (2022)

- [36] Peng, C., Chellappa, R.: Pdrf: progressively deblurring radiance field for fast scene reconstruction from blurry images. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 2029–2037 (2023)
- [37] Peng, C., Tang, Y., Zhou, Y., Wang, N., Liu, X., Li, D., Chellappa, R.: Bags: Blur agnostic gaussian splatting through multi-scale kernel modeling. arXiv preprint arXiv:2403.04926 (2024)
- [38] Peng, S., Dong, J., Wang, Q., Zhang, S., Shuai, Q., Zhou, X., Bao, H.: Animatable neural radiance fields for modeling dynamic human bodies. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14314–14323 (2021)
- [39] Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-nerf: Neural radiance fields for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10318–10327 (2021)
- [40] Rodrigues, O.: De l’attraction des sphéroïdes, Correspondence sur l’École Impériale Polytechnique. Ph.D. thesis, PhD thesis, Thesis for the Faculty of Science of the University of Paris (1816)
- [41] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III 18. pp. 234–241. Springer (2015)
- [42] Rubanova, Y., Chen, R.T., Duvenaud, D.K.: Latent ordinary differential equations for irregularly-sampled time series. Advances in neural information processing systems **32** (2019)
- [43] Schönberger, J.L., Zheng, E., Frahm, J.M., Pollefeys, M.: Pixelwise view selection for unstructured multi-view stereo. In: European conference on computer vision. pp. 501–518. Springer (2016)
- [44] Servoing, V.: Real time control of robot manipulators based on visual sensory feedback. World Scientific Series in Robotics and Automated Systems **7** (1993)
- [45] Shan, Q., Jia, J., Agarwala, A.: High-quality motion deblurring from a single image. ACM transactions on graphics (tog) **27**(3), 1–10 (2008)
- [46] Sim, H., Kim, M.: A deep motion deblurring network based on per-pixel adaptive kernels with residual down-up and up-down modules. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)
- [47] Son, H., Lee, J., Lee, J., Cho, S., Lee, S.: Recurrent video deblurring with blur-invariant motion estimation and pixel volumes. ACM Transactions on Graphics (TOG) **40**(5), 1–18 (2021)
- [48] Sun, J., Xie, Y., Chen, L., Zhou, X., Bao, H.: Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15598–15607 (2021)
- [49] Tao, X., Gao, H., Shen, X., Wang, J., Jia, J.: Scale-recurrent network for deep image deblurring. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8174–8182 (2018)
- [50] Tretschk, E., Tewari, A., Golyanik, V., Zollhöfer, M., Lassner, C., Theobalt, C.: Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12959–12970 (2021)
- [51] Wang, G., Chen, Z., Loy, C.C., Liu, Z.: Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9065–9076 (2023)
- [52] Wang, J., Wang, P., Long, X., Theobalt, C., Komura, T., Liu, L., Wang, W.: Neuris: Neural reconstruction of indoor scenes using normal priors. In: European Conference on Computer Vision. pp. 139–155. Springer (2022)

- [53] Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 (2021)
- [54] Wang, P., Zhao, L., Ma, R., Liu, P.: Bad-nerf: Bundle adjusted deblur neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4170–4179 (2023)
- [55] Wang, Z., Wu, S., Xie, W., Chen, M., Prisacariu, V.A.: Nerf–: Neural radiance fields without known camera parameters. arXiv preprint arXiv:2102.07064 (2021)
- [56] Weng, C.Y., Curless, B., Srinivasan, P.P., Barron, J.T., Kemelmacher-Shlizerman, I.: Humannerf: Free-viewpoint rendering of moving people from monocular video. In: Proceedings of the IEEE/CVF conference on computer vision and pattern Recognition. pp. 16210–16220 (2022)
- [57] Wynn, J., Turmukhambetov, D.: Diffusionerf: Regularizing neural radiance fields with denoising diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4180–4189 (2023)
- [58] Yang, J., Pavone, M., Wang, Y.: Freenerf: Improving few-shot neural rendering with free frequency regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8254–8263 (2023)
- [59] Yifan, W., Serena, F., Wu, S., Öztireli, C., Sorkine-Hornung, O.: Differentiable surface splatting for point-based geometry processing. ACM Transactions on Graphics (TOG) **38**(6), 1–14 (2019)
- [60] Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5752–5761 (2021)
- [61] Zamir, S.W., Arora, A., Khan, S., Hayat, M., Khan, F.S., Yang, M.H., Shao, L.: Multi-stage progressive image restoration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 14821–14831 (2021)
- [62] Zamir, S.W., Arora, A., Khan, S., Hayat, M., Khan, F.S., Yang, M.H., Shao, L.: Multi-stage progressive image restoration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 14821–14831 (2021)
- [63] Zhang, K., Ren, W., Luo, W., Lai, W.S., Stenger, B., Yang, M.H., Li, H.: Deep image deblurring: A survey. International Journal of Computer Vision **130**(9), 2103–2130 (2022)
- [64] Zhao, L., Wang, P., Liu, P.: Bad-gaussians: Bundle adjusted deblur gaussian splatting. arXiv preprint arXiv:2403.11831 (2024)

Appendix

A Performance on Real-World Scenes

As shown in Tab. 4, CRiM-GS demonstrates superior quantitative performance across most real-world scenes. In particular, LPIPS shows the best performance not only for synthetic scenes but also for all real-world scenes, highlighting the excellent quality achieved by our CRiM-GS.

Table 4: **Comparison of performance on the real-world scenes.** CRiM-GS demonstrates high performance in most real-world scenes and shows particularly superior performance in LPIPS.

Real-World Scene	BALL			BASKET			BUICK			COFFEE			DECORATION		
	PSNR	SSIM	LPIPS												
Naive NeRF [28]	24.08	0.6237	0.3992	23.72	0.7086	0.3223	21.59	0.6325	0.3502	26.48	0.8064	0.2896	22.39	0.6609	0.3633
Deblur-NeRF [25]	27.36	0.7656	0.2230	27.67	0.8449	0.1481	24.77	0.7700	0.1752	30.93	0.8981	0.1244	24.19	0.7707	0.1862
PDRF-10* [36]	27.37	0.7642	0.2093	28.36	0.8736	0.1179	25.73	0.7916	0.1582	31.79	0.9002	0.1133	23.55	0.7508	0.2145
BAD-NeRF* [54]	21.33	0.5096	0.4692	26.44	0.8080	0.1325	21.63	0.6429	0.2593	28.98	0.8369	0.1956	22.13	0.6316	0.2894
DP-NeRF [18]	27.20	0.7652	0.2088	27.74	0.8455	0.1294	25.70	0.7922	0.1405	31.19	0.9049	0.1002	24.31	0.7811	0.1639
DeblurGS [31]	21.44	0.5760	0.1480	21.00	0.6706	0.0908	18.91	0.5539	0.1442	25.97	0.8299	0.0614	19.60	0.5713	0.1035
BAD-GS* [64]	22.14	0.5791	0.1095	21.48	0.6787	0.0603	18.81	0.5271	0.0952	24.54	0.7627	0.1031	20.30	0.6162	0.0881
BAGS [37]	27.68	0.7990	0.1500	29.54	0.9000	0.0680	26.18	0.8440	0.0880	31.59	0.9080	0.0960	26.09	0.8580	0.0830
CRiM-GS	28.25	0.8064	0.1074	30.85	0.9048	0.0425	26.51	0.8379	0.0585	32.26	0.9230	0.0421	26.34	0.8610	0.0569
Real-World Scene	GIRL			HERON			PARTERRE			PUPPET			STAIR		
	PSNR	SSIM	LPIPS												
Naive NeRF [28]	20.07	0.7075	0.3196	20.50	0.5217	0.4129	23.14	0.6201	0.4046	22.09	0.6093	0.3389	22.87	0.4561	0.4868
Deblur-NeRF [25]	22.27	0.7976	0.1687	22.63	0.6874	0.2099	25.82	0.7597	0.2161	25.24	0.7510	0.1577	25.39	0.6296	0.2102
PDRF-10* [36]	24.12	0.8328	0.1679	22.53	0.6880	0.2358	25.36	0.7601	0.2263	25.02	0.7496	0.1532	25.20	0.6235	0.2288
BAD-NeRF* [54]	18.10	0.5652	0.3933	22.18	0.6479	0.2226	23.44	0.6243	0.3151	22.48	0.6249	0.2762	21.52	0.4237	0.3341
DP-NeRF [18]	23.33	0.8139	0.1498	22.88	0.6930	0.1914	25.86	0.7665	0.1900	25.25	0.7536	0.1505	25.59	0.6349	0.1772
DeblurGS* [31]	18.28	0.6660	0.1044	18.70	0.4579	0.1488	19.78	0.4953	0.1744	19.50	0.5302	0.1297	21.63	0.4623	0.0808
BAD-GS* [64]	18.72	0.6765	0.0925	18.53	0.4498	0.1412	20.79	0.5589	0.1298	21.08	0.6036	0.1022	21.77	0.4640	0.0782
BAGS [37]	25.45	0.8690	0.0790	22.04	0.7150	0.1260	25.92	0.8190	0.0920	25.81	0.8040	0.0940	26.69	0.7210	0.0800
CRiM-GS	26.20	0.8780	0.0438	22.84	0.7234	0.1110	26.09	0.8126	0.0698	26.66	0.8236	0.0613	27.30	0.7390	0.0415

B Pose Optimization

We re-ran DeblurGS [31] because it performs test pose optimization after learning 3D Gaussian Splatting. This optimization process requires test images, making it difficult to consider a fair comparison. Nevertheless, as shown in Tab. 5, CRiM-GS achieves better performance even when subjected to the same pose optimization process as DeblurGS, and it even outperforms the optimized DeblurGS without undergoing the optimization process itself. This indicates that CRiM-GS is robust to inaccurate camera poses. As illustrated in the error maps in Fig. 4, CRiM-GS not only performs better than DeblurGS but also optimizes poses better than other models without additional training.

Table 5: Comparisons on synthetic and real-world scene dataset.

Methods	Pose Optimization		Synthetic Scene Dataset			Real-World Scene Dataset		
			PSNR(\uparrow)	SSIM(\uparrow)	LPIPS(\downarrow)	PSNR(\uparrow)	SSIM(\uparrow)	LPIPS(\downarrow)
DeblurGS* [31]			20.22	0.5454	0.1042	20.48	0.5813	0.1186
CRiM-GS			30.19	0.9004	0.0485	27.33	0.8310	0.0634
DeblurGS* [31]	✓		30.24	0.8922	0.0637	26.07	0.8024	0.0958
CRiM-GS	✓		30.51	0.9051	0.0478	28.34	0.8478	0.0606

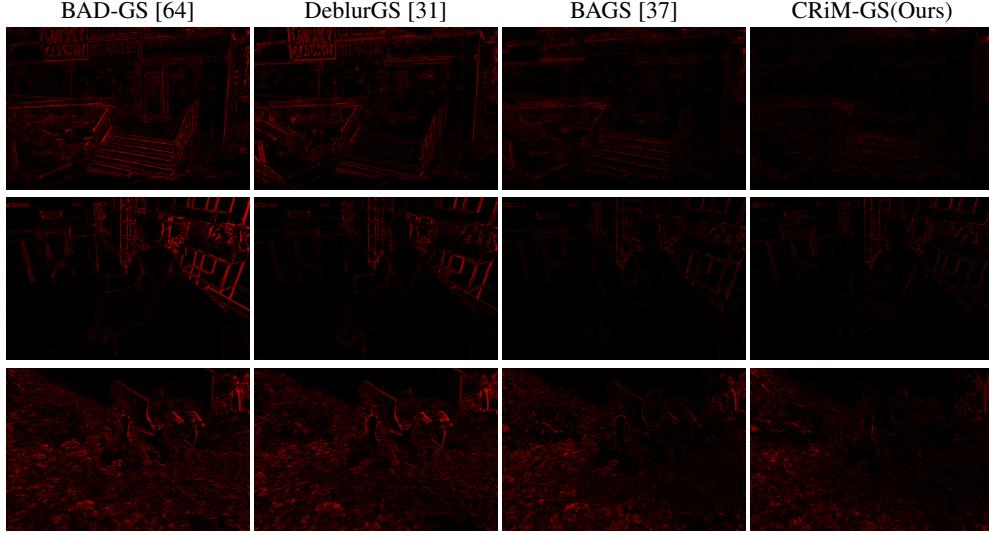


Figure 4: **Qualitative comparison on the synthetic and real-world scenes.**

C Additional Visualization

Additional visual results are shown in Fig. 5, demonstrating that our CRiM-GS not only achieves superior quantitative performance but also delivers excellent qualitative results.

D Limitations

CRiM-GS effectively models the continuous camera trajectory occurring during the exposure time. However, rendering N images to create a blurry image impacts the training time. Additionally, our model is optimized specifically for camera motion blur and does not account for other degradation phenomena such as defocus blur. Therefore, our next goal is to propose a method that involves post-processing through training, requiring only a single rendering pass. Furthermore, we aim to apply depth of field (DoF) to 3D Gaussian Splatting to reconstruct 3D scenes solely from blurry images caused by defocus.

E Broader Impacts

Our research presents a novel method for reconstructing sharp 3D scenes from images blurred by camera motion. By integrating neural ODEs, commonly used in computer vision applications, into 3D vision, our approach demonstrates robustness theoretically, mathematically, and experimentally. Additionally, by modeling the most common type of blur that can occur in real-world image acquisition environments, our method has potential applications in user-interactive VR, AR, and other various applications.

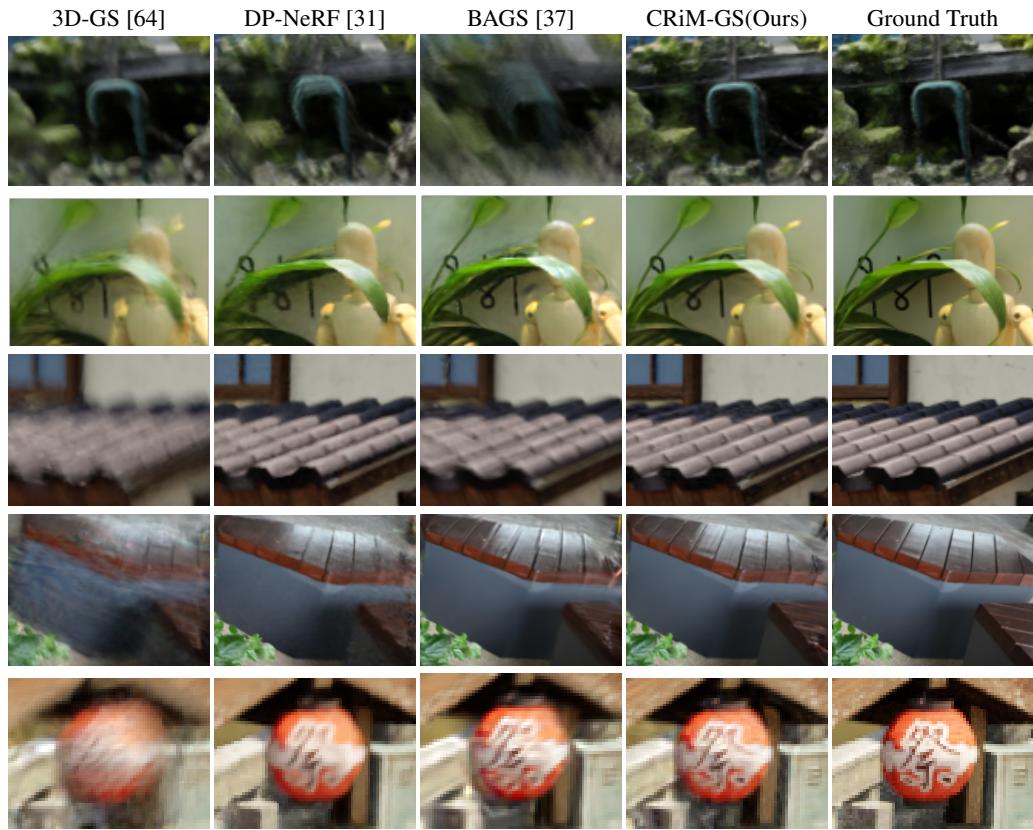


Figure 5: Qualitative comparison on the synthetic and real-world scenes.