ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVG
Computer Vision
and Geometry Lab

MonoDy-GS:
Online Monocular Dynamic Gaussian Splatting

Semester Project

Shi Chen
Department of Information Technology and Electrical Engineering

Advisors: Prof. Dr. Martin R. Oswald
Dr. Sandro Lombardi
Supervisor: Prof. Dr. Marc Pollefeys
Computer Vision and Geometry Group
Department of Computer Science D-INFK

August 20, 2024

Abstract

In this project, we attempt to address the challenge of online modelling of dynamic scenes from monocular RGB-D inputs. Traditional methods for dynamic scene reconstruction often rely on extensive multi-view coverage or operate in an offline manner, which limits their accessibility and interactability in potential application scenarios such as Mixed Reality. To overcome these limitations, we propose MonoDy-GS, an online system that uses 3D Gaussians as the underlying representation to model dynamic scenes. Our approach incorporates several priors based on real-world scene dynamics to compensate for the lack of multi-view information inherent in monocular setups. This enables the system to track and model scene deformations incrementally as new data becomes available. We validate the effectiveness of MonoDy-GS in modelling scene dynamics through a series of qualitative and quantitative experiments. However, the current implementation still faces two significant challenges: Maintaining spatial and temporal motion consistency in fast-moving scenes and ensuring geometric accuracy across different viewpoints. We suggest potential solutions to these challenges for future work. Overall, the development of MonoDy-GS represents a step forward in the field of dynamic scene reconstruction, providing insights that could guide future improvements and applications.

Acknowledgements

I would like to express my deepest gratitude to my advisors, Prof. Dr. Martin R. Oswald and Dr. Sandro Lombardi, for their unwavering support and guidance throughout this project. Their expertise and encouragement were invaluable, and this work would not have been possible without their continuous help. I am also sincerely grateful to Prof. Marc Pollefeys for facilitating our collaboration and for kindly approving my request for a deadline extension during the final stages of the project. Lastly, I extend my heartfelt thanks to Hidenobu Matsuki from Imperial College London for generously open-sourcing their codebase, MonoGS, which laid the foundation for this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mixed Reality (MR) is an evolving field that merges real and virtual environments, enabling users to interact with digital content in ways that closely mimic real-world experiences. The development of MR technologies relies heavily on both photorealistic and fast rendering capabilities to create immersive environments that respond in real-time to user input. Achieving this level of realism and responsiveness is crucial for applications ranging from gaming to training simulations and beyond.

Gaussian Splatting [9], a technique that excels in photorealistic rendering while maintaining real-time processing, has emerged as a promising solution for such demanding MR applications. By representing scenes with 3D Gaussians, this approach can produce high-quality renderings at speeds conducive to interactive applications. This capability makes it a strong candidate for creating immersive and interactive 3D environments that can respond to user input in real-time.

Building on this foundation, recent research [4, 16, 27, 35, 20, 18, 38, 13, 32, 1, 14, 17, 11, 6] has introduced deformable Gaussian representations that extend the capabilities of Gaussian Splatting to dynamic scenes. These advancements allow virtual objects to not only appear photorealistic but also move and deform in ways that resemble real-world physics. Such developments are crucial for enhancing the interactivity of MR environments, enabling users to engage with digital content that behaves as naturally as real objects.

However, these methods often rely on extensive multi-view coverage during the scene capture process. In fact, high-quality multi-view data collection can be prohibitively expensive and technically demanding, creating a barrier to the widespread adoption of these techniques. This limitation particularly affects casual users and content creators who may not have access to the sophisticated equipment required for capturing multi-view data, thereby limiting the availability of interactable 3D assets.

To address these challenges, researchers have begun exploring methods for 4D reconstruction using monocular video inputs [26, 24, 25, 31, 15, 12, 19, 34, 28, 33, 36]. These approaches could significantly lower the barrier to entry by leveraging the vast number of monocular videos available on platforms such as YouTube and TikTok, as well as those that can be casually captured using smartphones. The ability to reconstruct dynamic 3D assets from such accessible data sources could broaden the creation of MR content, making it easier for users to generate and interact with high-quality digital environments.

Nevertheless, existing methods for monocular 4D reconstruction are typically offline methods. Users must provide an entire video and then wait for the processing to complete, without knowing how the reconstrction will turn out. This offline nature is undesirable, as it prevents users from seeing intermediate results and making adjustments on the fly. An online approach to 4D reconstruction would not only improve the user experience by providing immediate feedback but also open up new possibilities for interactive content creation in MR.

In this project, we explore the feasibility of online modelling of dynamic scenes in a monocular setting using 3D Gaussians as the underlying representation. To this end, we propose MonoDy-GS. MonoDy-GS is built upon MonoGS [22], which is a Gaussian-based SLAM system showing excellent tracking and mapping performance in various static scenes. For modelling scene deformation using Gaussians, our approach is inspired by the “Dynamic 3D Gaussians” method, while incorporating several priors to compensate for the lack of multi-view information inherent in a monocular setting.

The contributions of this project are as follows:

- We propose MonoDy-GS, an online pipeline capable of modeling dynamic scenes from an RGB-D sequence, using 3D Gaussians as the foundational representation.
- We improve the performance of Gaussian tracking in a monocular setting by integrating various priors based on real-world scene dynamics, addressing the challenges posed by the lack of multi-view data.
- We validate the effectiveness of MonoDy-GS in modeling scene deformations through a series of qualitative and quantitative experiments.
- We further discuss the limitations of MonoDy-GS, including poor spatial and temporal motion consistency and poor geometry, and attempt to propose potential solutions that can be explored in the future.

Chapter 2

Related Work

2.1 Deformable Gaussian Representations

Deformable Gaussian representations have been actively explored since the introduction of the seminal work 3D Gaussian Splatting [9], and these methods can generally be categorized based on their approach to modelling deformations.

Canonical Space and Deformation Field In this approach [4, 16, 27, 35, 20, 18, 38, 13, 32, 19, 36], Gaussians are anchored in a shared canonical space, and the deformation field at a given timestamp, with respect to the canonical frame, is inferred using simple neural networks such as MLPs. While these methods are relatively straightforward to implement and have been successful in modeling object-level short sequences, they tend to struggle with complex deformations in longer sequences, particularly when topological changes are involved. Moreover, MLPs are prone to catastrophic forgetting, and can be time-consuming to optimize, which hinder its online or interactive application.

4D Gaussians Rather than actually moving the Gaussians, several methods [1, 14] extend the mechanics of 3D Gaussian Splatting into a fourth dimension, allowing Gaussians to appear transiently. However, this approach does not represent the dynamic parts of the scene with a consistent set of Gaussians, making it challenging to maintain temporal consistency in appearance and geometry.

Gaussian Trajectory Decomposition This category of methods leverages the low-rank nature of Gaussian dynamics by decomposing each Gaussian's trajectory into a combination of several fixed or learnable trajectory bases. Lin et al. [17] uses spline-fitting to decompose Gaussian motions, while Kratimenos et al. [11] applies learnable neural motion factorization. A few methods [33, 12] utilizes motion scaffolds for compact trajectory representation, and Katsumata et al. [6] employs Fourier series to approximate Gaussian positions and orientations. Although these methods are often efficient, they are essentially compressed representations of Gaussian dynamics, which can lead to compromised accuracy.

Direct Gaussian Tracking Luiten et al. [21] proposed a method to directly track the motion of each Gaussian over time. Our approach draws inspiration from this technique. However, this method assumes sufficient multi-view coverage at every timestamp, which does not hold in our monocular setup. Therefore, we introduce various priors to guide the optimization of Gaussian parameters.

2.2 4D Reconstruction from Monocular Videos

Several methods have been proposed for 4D reconstruction from monocular videos, often using Neural Radiance Field (NeRF) or 3D Gaussians as underlying representations.

NeRF-Based Methods NeRF and its variants have been used extensively for 4D reconstruction, offering outstanding results in modeling dynamic scenes from monocular video inputs [26, 24, 25, 31, 15]. However, these methods typically suffer from slow rendering, making them unsuitable for interactive or real-time applications.

Gaussian-Based Methods The emergence of 3D Gaussian Splatting [10] revolutionized the field of 3D reconstruction with its real-time-capable fast rendering. Gaussian-based approaches have also been employed for 4D reconstruction from monocular videos. MoSca [12] models scene dynamics using a sparse graph of motion trajectories, which can be interpolated into individual Gaussian trajectories. MoDGS [19] and GFlow [34] both leverage 2D priors, such as monocular depth and optical flow estimation, to constrain Gaussian motion under the monocular setting. Dynamic Gaussian Marbles [28] simplifies the motion model by using isotropic Gaussians, eliminating the degrees of freedom for rotation, and effectively incorporates 2D point tracking priors. Shape of Motion [33] employs trajectory decomposition along with 2D priors to reconstruct 4D geometry and appearance. Gaussian Splatting LK [36] introduces an analytical scene flow representation, enhancing structural fidelity in dynamic scenes, particularly those with rapid motions.

Both the NeRF-based and the Gaussian-based methods are typically offline, which means that they require access to all frames simultaneously. This contrasts with our requirement for online processing, which adds more difficulty to the reconstruction task.

2.3 Gaussian-Based SLAM

Gaussian-based SLAM systems have been developed to address the challenge of dense scene reconstruction in real time. Notable examples include SplaTAM [8], Gaussian-SLAM [39], and MonoGS [22], among others [3, 37]. These systems leverage the efficiency of 3D Gaussian representations to achieve photo-realistic reconstructions while navigating through the environment.

However, these methods are inherently designed for static scenes, and thus lack the capability to model temporally inconsistent Gaussian attributes. This limitation restricts their applicability to dynamic environments, where scene elements might deform or move over time. The challenge we address in this project can thus be seen as extending such Gaussian-based SLAM systems to become “deformation-aware”, enabling them to handle dynamic scenes by simultanously modeling scene motion and the reconstructed geometry.

Chapter 3

Method

3.1 Problem Statement

This project aims to model dynamic scenes from monocular videos using 3D Gaussians as the underlying representation. The method is designed to operate in an online fashion, where the Gaussian representation is incrementally updated with each incoming frame. In order to limit the scope and complexity of this project, we assume that all camera intrinsic parameters and camera poses are known throughout the video sequence. Additionally, our method operates under the assumption that most parts of the scene remain static, which is based on the observation that a casual monocular video usually contains a large chunk of pixels capturing static backgrounds. This assumption helps regularize the underdetermined Gaussian motions.

3.2 Preliminary: MonoGS

MonoGS [22] leverages 3D Gaussians as its foundational representation to achieve precise and efficient tracking and mapping in static scenes. The system is structured into a multi-process architecture, which separates the tasks of tracking and mapping to optimize performance.

The system's frontend is responsible for the online optimization of camera poses. It continually adjusts the camera poses as new data comes in, ensuring that the system remains aligned with the incoming data. Once a keyframe is detected based on a covisibility criterion, the frontend sends the optimized camera poses to the backend.

The backend of MonoGS then refines the Gaussian representation using multi-view constraints derived from these keyframes. This approach allows the backend to update the Gaussians used by the frontend for continuous tracking, ensuring that the system maintains a high level of accuracy in the tracking process.

We build MonoDy-GS system upon the MonoGS codebase for its superior accuracy in both tracking and mapping. In MonoDy-GS, in accordance with our previous assumption that camera poses are known, we omit the camera tracking feature by directly feeding ground truth camera poses into the system. This adjustment allows the system to focus on modeling the dynamics in the scene without the need for joint camera pose optimization.

3.3 Gaussian Tracking

3.3.1 Modeling Dynamics with 3D Gaussians

Our approach to tracking Gaussian motions draws inspiration from the *Dynamic 3D Gaussians* method proposed by Luiten et al. [21]. In MonoDy-GS, we optimize the positions and orientations of Gaussians in

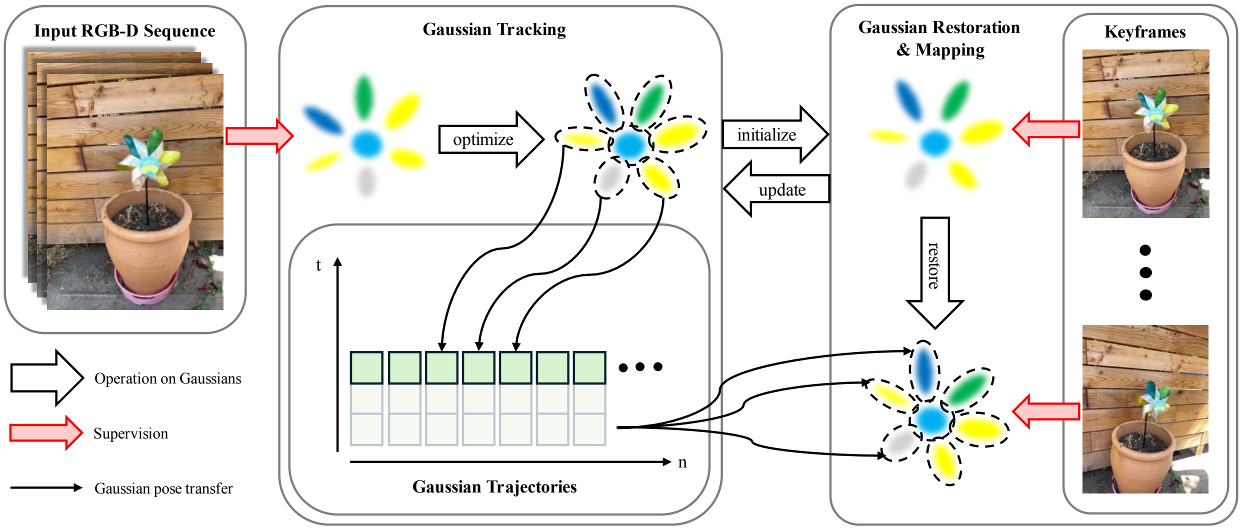


Figure 3.1: **MonoDy-GS Overview.** The frontend performs Gaussian tracking, guided by each frame of the input RGB-D sequence and additional priors. Once the tracking converges, the Gaussian trajectories at the current frame are updated using the optimized Gaussian poses. We use a square block to represent a stored Gaussian pose (*i.e.* center position and orientation). When a keyframe update is requested, the frontend sends the latest Gaussian attributes and Gaussian trajectories sliced at selected keyframes to the backend. The backend restores the Gaussian poses at past keyframes using the provided Gaussian trajectories and performs deformation-aware multi-view mapping on both the latest and restored Gaussians. Finally, the backend updates the frontend with the refined Gaussian attributes, allowing the frontend to begin the next round of Gaussian tracking.

the frontend in an online manner, updating the Gaussian poses incrementally with each new frame. Note that we remove the constant speed assumption employed in *Dynamic 3D Gaussians*. Instead, we initialize the Gaussians at the poses they held after optimizing for the previous frame. This adjustment enhances optimization stability, particularly in the challenging monocular setting where multi-view constraints are lacking.

To guide the Gaussian motion optimization, we employ the basic photometric and depth rerendering loss, which is also used in MonoGS [22]. Additionally, to further address the absence of multi-view constraints inherent in a monocular setup, we introduce several additional priors to achieve more reliable and consistent tracking of dynamic elements in the scene. The loss functions are explained in detail in 3.3.2.

3.3.2 Loss Functions

Photometric and Depth Rerendering Loss Following MonoGS [22], we utilize the photometric and depth rerendering loss to supervise the optimization of Gaussian poses during Gaussian tracking. These loss functions ensure that the rendered image and depth map from the current Gaussians closely match the observed RGB-D data.

Let $\mathcal{G}$ represent the set of Gaussians, and $\boldsymbol{T} \in \boldsymbol{SE}(3)$ denote the ground truth camera pose at the current time step. The photometric rerendering loss $\mathcal{L}_{\text{pho}}$ is defined as the L_1 norm between the observed image I_{obs} and the rendered image $I(\mathcal{G}, \boldsymbol{T})$

$$\mathcal{L}_{\text{pho}} = \|I(\mathcal{G}, \boldsymbol{T}) - I_{\text{obs}}\|_1, \tag{3.1}$$

where $I(\mathcal{G}, \boldsymbol{T})$ is the image rendered by rasterizing the Gaussians $\mathcal{G}$ onto the image plane using the ground truth camera pose $\boldsymbol{T}$.

Similarly, the depth rerendering loss $\mathcal{L}_{\text{depth}}$ is defined as the L_1 error between the observed depth map D_{obs} and the rendered depth map $\hat{D}$

$$\mathcal{L}_{\text{depth}} = \|\hat{D} - D_{\text{obs}}\|_1. \tag{3.2}$$

Here, per-pixel depth $\hat{D}(\boldsymbol{u})$ is rasterized by alpha-blending as

$$\hat{D}(\boldsymbol{u}) = \sum_{i=1}^{N} z^i \alpha^i(\boldsymbol{u}) \prod_{j=1}^{i-1}(1 - \alpha^j(\boldsymbol{u})), \tag{3.3}$$

where $\alpha^i(\boldsymbol{u})$ denotes the opacity value of Gaussian i, evaluated at the pixel with 2D image coordinates $\boldsymbol{u} = (u, v)^T$ from the 2D Gaussian formed by splatting the 3D Gaussian onto the image plane of the ground truth camera pose $\boldsymbol{T}$, and z^i is the distance to the center mean of Gaussian i along the camera ray.

We use the notation $\mathcal{L}_{\text{RGB-D}}$ to represent a weighted sum of the photometric and depth rerendering loss terms:

$$\mathcal{L}_{\text{RGB-D}} = \lambda_{\text{pho}} \mathcal{L}_{\text{pho}} + (1 - \lambda_{\text{pho}}) \mathcal{L}_{\text{depth}}, \tag{3.4}$$

where λ_{pho} is a hyperparameter controlling the balance between the photometric and depth losses. We follow the hyperparameter configuration from [22] and set $\lambda_{\text{pho}} = 0.9$.

Optical Flow Prior We incorporate a 2D optical flow prior to guide the motion of Gaussians. In the absence of ground truth optical flow, we use pseudo-ground truth estimated by RAFT [30], an off-the-shelf optical flow estimator. The optical flow is rendered from the Gaussians using a differentiable rasterizer, enabling the alignment of Gaussian motions with the observed flow patterns in the scene through error backpropagation. Both forward and backward flows are utilized in this process.

Our method for rendering optical flow from Gaussians is inspired by Katsumata et al. [7], with the key distinction that we account for camera motion. Let the position of a Gaussian center at the current frame be $\boldsymbol{\mu}_t^i$ and at the previous frame be $\boldsymbol{\mu}_{t-1}^i$. The Gaussian's contribution to the forward and backward optical flow is given by

$$\hat{f}_{\text{fwd}}^i = \mathbf{J}_t^i \boldsymbol{\mu}_t^i - \mathbf{J}_{t-1}^i \boldsymbol{\mu}_{t-1}^i, \tag{3.5}$$

$$\hat{f}_{\text{bwd}}^i = \mathbf{J}_{t-1}^i \boldsymbol{\mu}_{t-1}^i - \mathbf{J}_t^i \boldsymbol{\mu}_t^i, \tag{3.6}$$

respectively, where $\mathbf{J}_t^i$ represents the Jacobian of the affine approximation of the projective transformation at $\boldsymbol{\mu}_t^i$. With N Gaussians sorted in ascending depth order (e.g., $i = 0$ being the closest), the pixel-wise forward and backward optical flow can be rendered via alpha-blending as

$$\hat{f}_{\text{fwd}}(\boldsymbol{u}) = \sum_{i=1}^{N} \hat{f}_{\text{fwd}}^i \alpha_{t-1}^i(\boldsymbol{u}) \prod_{j=1}^{i-1}(1 - \alpha_{t-1}^j(\boldsymbol{u})), \tag{3.7}$$

$$\hat{f}_{\text{bwd}}(\boldsymbol{u}) = \sum_{i=1}^{N} \hat{f}_{\text{bwd}}^i \alpha_t^i(\boldsymbol{u}) \prod_{j=1}^{i-1}(1 - \alpha_t^j(\boldsymbol{u})), \tag{3.8}$$

where $\alpha_t^i(\boldsymbol{u})$ is the 2D opacity value, defined similarly to those in Eq. 3.3, with t denoting the frame index. For implementation, we reuse the CUDA-optimized rasterizer [9] originally designed for rendering color images, enabling efficient optical flow generation with comparable rendering speed.

Finally, with ground truth or pseudo-ground truth optical flow f_{fwd} and f_{bwd}, we compute the optical flow loss $\mathcal{L}_{\text{flow}}$ as the L_1 error

$$\mathcal{L}_{\text{flow}} = \|\hat{f}_{\text{fwd}} - f_{\text{fwd}}\|_1 + \|\hat{f}_{\text{bwd}} - f_{\text{bwd}}\|_1. \tag{3.9}$$

Physically-Based Priors In MonoDy-GS, we incorporate several physically-based priors to maintain realistic and consistent Gaussian motions, following *Dynamic 3D Gaussians* [21]. These priors include local-rigidity, local-rotation similarity, and local-isometry losses, which softly constrain any Gaussian motion that violates the local rigidity prior.

The local-rigidity loss $\mathcal{L}_{\text{rigid}}$ ensures that nearby Gaussians move in a manner that follows a rigid-body transformation. For each Gaussian i, its movement relative to nearby Gaussians j is expected to follow this transformation. The local-rigidity loss is defined as

$$\mathcal{L}_{\text{rigid}}^{i,j} = w^{i,j} \left\|(\boldsymbol{\mu}_{t-1}^j - \boldsymbol{\mu}_{t-1}^i) - \mathbf{R}_{t-1}^i \mathbf{R}_t^{i^{-1}} (\boldsymbol{\mu}_t^j - \boldsymbol{\mu}_t^i)\right\|_2, \tag{3.10}$$

$$\mathcal{L}_{\text{rigid}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} \mathcal{L}_{\text{rigid}}^{i,j}, \tag{3.11}$$

$$\mathcal{L}_{\text{rigid}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w^{i,j} \left\|(\boldsymbol{\mu}_{t-1}^j - \boldsymbol{\mu}_{t-1}^i) - \mathbf{R}_{t-1}^i \mathbf{R}_t^{i^{-1}} (\boldsymbol{\mu}_t^j - \boldsymbol{\mu}_t^i)\right\|_2, \tag{3.12}$$

where $\mathbf{R}_t^i$ denotes the rotation matrix of Gaussian i at frame t, and k is the number of nearest neighbors considered. We follow Luiten et al. [21] and set $k = 20$ for all k-nearest-neighbors searches in our method. The term $w^{i,j}$ is a Gaussian weighting factor defined as

$$w^{i,j} = \exp\left(-\lambda_w \left\|\boldsymbol{\mu}_{t-1}^j - \boldsymbol{\mu}_{t-1}^i\right\|_2^2\right), \tag{3.13}$$

where λ_w is a constant that controls the spread of the weighting, effectively restricting the rigidity loss to local neighborhoods. We set $\lambda_w = 2000$ following Luiten et al. [21].

The local-rotation similarity loss $\mathcal{L}_{\text{rot}}$ enforces that neighboring Gaussians maintain similar rotations over time. This loss is defined as

$$\mathcal{L}_{\text{rot}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w^{i,j} \left\|\boldsymbol{q}_t^j \boldsymbol{q}_{t-1}^{j^{-1}} - \boldsymbol{q}_t^i \boldsymbol{q}_{t-1}^{i^{-1}}\right\|_2, \tag{3.14}$$

where $\boldsymbol{q}_t^i$ represents the normalized quaternion describing the rotation of Gaussian i at frame t.

Lastly, the local-isometry loss $\mathcal{L}_{\text{iso}}$ is applied to ensure that the distances between neighboring Gaussians remain consistent over time. This loss is given by

$$\mathcal{L}_{\text{iso}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w^{i,j} \left|\left\|\boldsymbol{\mu}_{t-1}^j - \boldsymbol{\mu}_{t-1}^i\right\|_2 - \left\|\boldsymbol{\mu}_t^j - \boldsymbol{\mu}_t^i\right\|_2\right|. \tag{3.15}$$

Finally, we compute a weighted sum of the three losses to form the total physically-based loss $\mathcal{L}_{\text{phys}}$, given by

$$\mathcal{L}_{\text{phys}} = \lambda_{\text{rigid}}\mathcal{L}_{\text{rigid}} + \lambda_{\text{rot}}\mathcal{L}_{\text{rot}} + \lambda_{\text{iso}}\mathcal{L}_{\text{iso}}, \tag{3.16}$$

where $\lambda_{\text{rigid}} = 4.0$, $\lambda_{\text{rot}} = 4.0$, and $\lambda_{\text{iso}} = 2.0$ are the respective weights for each loss term. Here, we follow the relative weights used in *Dynamic 3D Gaussians* [21].

Motion Sparsity Regularization We apply an L_1 regularization on all Gaussian motions to promote sparsity in the motion field, consistent with our assumption that the majority of the scene remains static.

The motion sparsity regularization term is defined as

$$\mathcal{L}_{\text{sparse}} = \|\Delta\tilde{\boldsymbol{p}}_t\|_1, \ \Delta\tilde{\boldsymbol{p}}_t = [\Delta\tilde{\boldsymbol{p}}_t^1, \Delta\tilde{\boldsymbol{p}}_t^2, \dots, \Delta\tilde{\boldsymbol{p}}_t^{|\mathcal{G}|}], \tag{3.17}$$

where $\Delta\tilde{\boldsymbol{p}}_t^i$ represents a weighted concatenation of positional and rotational changes of Gaussian i between consecutive frames:

$$\Delta\tilde{\boldsymbol{p}}_t^i = \begin{bmatrix} \Delta\boldsymbol{\mu}_t^i \\ \lambda_{\boldsymbol{q}}\Delta\boldsymbol{q}_t^i \end{bmatrix}, \ \Delta\boldsymbol{\mu}_t^i = \boldsymbol{\mu}_t^i - \boldsymbol{\mu}_{t-1}^i, \ \Delta\boldsymbol{q}_t^i = \boldsymbol{q}_t^i - \boldsymbol{q}_{t-1}^i. \tag{3.18}$$

By concatenating $\Delta\boldsymbol{\mu}_t^i$ and $\Delta\boldsymbol{q}_t^i$, we enforce a unified sparsity pattern for both positional and rotational changes through joint L_1 regularization. This design reflects the prior that if a Gaussian corresponds to a static region of the scene, both its position and orientation should remain constant. The weighting coefficient $\lambda_{\boldsymbol{q}}$ balances the relative influence between the positional and rotational components, with $\lambda_{\boldsymbol{q}} = 0.2$ determined empirically.

Motion TV Regularization A 2D total variation (TV) regularization is applied to the rasterized Gaussian translational motions, encouraging piecewise constant motion fields that are consistent with typical rigid-body dynamics in the real world.

We first rasterize Gaussian motions into 2D via alpha-rendering:

$$\Delta\boldsymbol{\mu}_{\text{2D}}(\boldsymbol{u}) = \sum_{i=1}^{N} \Delta\boldsymbol{\mu}_t^i \alpha_{t-1}^i(\boldsymbol{u}) \prod_{j=1}^{i-1} (1 - \alpha_{t-1}^j(\boldsymbol{u})). \tag{3.19}$$

The resulting 2D translational motion field $\Delta\boldsymbol{\mu}_{\text{2D}}$ captures the collective motion of all Gaussians splatted onto the image plane of frame t. Again, we reuse the CUDA-optimized rasterizer [9] here. The TV regularization term is then defined as

$$\mathcal{L}_{\text{TV}} = \sum_{u,v} \left| \frac{\partial\Delta\boldsymbol{\mu}_{\text{2D}}(\boldsymbol{u})}{\partial u} \right| + \left| \frac{\partial\Delta\boldsymbol{\mu}_{\text{2D}}(\boldsymbol{u})}{\partial v} \right|, \tag{3.20}$$

where $\partial\Delta\boldsymbol{\mu}_{\text{2D}}/\partial u$ and $\partial\Delta\boldsymbol{\mu}_{\text{2D}}/\partial v$ represent the partial derivatives of the motion field $\Delta\boldsymbol{\mu}_{\text{2D}}$ with respect to the horizontal and vertical image coordinates u and v, respectively.

In summary, the Gaussian tracking problem is formulated as minimizing a weighted sum of all the aforementioned loss terms:

$$\min_{\boldsymbol{\mu}_t, \boldsymbol{q}_t} \mathcal{L}_{\text{RGB-D}} + \lambda_{\text{flow}}\mathcal{L}_{\text{flow}} + \lambda_{\text{phys}}\mathcal{L}_{\text{phys}} + \lambda_{\text{sparse}}\mathcal{L}_{\text{sparse}} + \lambda_{\text{TV}}\mathcal{L}_{\text{TV}}, \tag{3.21}$$

where $\boldsymbol{\mu}_t$ and $\boldsymbol{q}_t$ represent the Gaussian positions and rotations at time t, respectively, and each λ represents a weighting coefficient that balances the contribution of the corresponding loss term to the overall optimization objective.

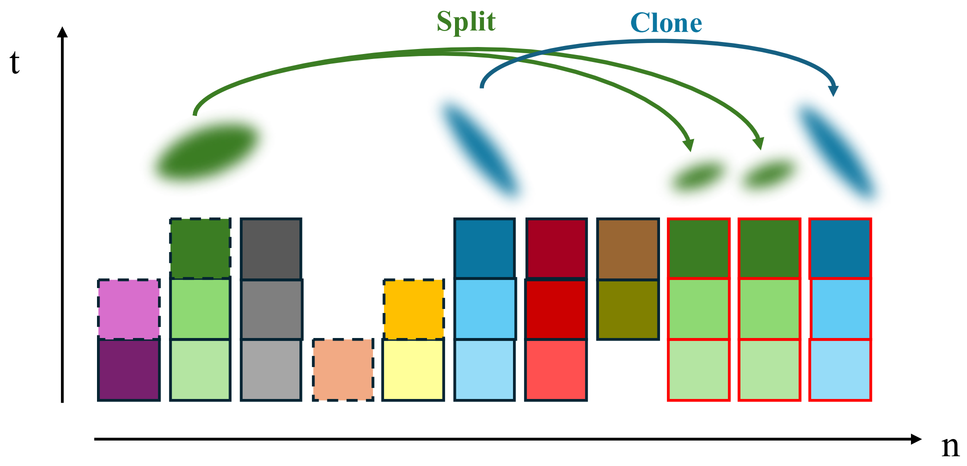


Figure 3.2: **Gaussian Trajectory Inheritance.** This figure illustrates how Gaussian trajectories are managed when Gaussians are split or cloned in the backend. A colored block represents the pose of a Gaussian (*i.e.* center position and orientation). Note that the colors are used solely to distinguish between different Gaussian poses and do not necessarily correspond to the colors of the Gaussians themselves. Red outlines indicate new Gaussians generated through splitting or cloning, while dotted outlines represent Gaussians that are pruned.

3.4 Mapping with Keyframes

3.4.1 Gaussian Trajectory Management

To enable the restoration of Gaussian poses at past frames, we maintain a record of all Gaussian poses (*i.e.* center positions and orientations) at every frame up to the current one in the frontend. This record is referred to as *Gaussian trajectories*.

As depicted in Figure 3.1, the Gaussian trajectories are continuously updated at each frame. Specifically, after the Gaussian tracking process is completed, the Gaussian trajectories corresponding to the current frame is updated with the newly optimized Gaussian poses. This approach allows us to track the scene dynamics over time.

When the frontend requests a keyframe update from the backend (as described in 3.4.2 and 3.4.3), the Gaussian trajectories at selected keyframes are transferred to the backend. This transfer allows the backend to restore the Gaussian poses at those keyframes.

As shown in Figure 3.2, in scenarios where Gaussians undergo splitting or cloning in the backend as part of the densification strategy, the "child" Gaussians (*i.e.* the new Gaussians inserted through splitting or cloning) inherit the trajectories of their "parent" Gaussians (*i.e.* the split or cloned Gaussians). This inheritance mechanism guarantees that we can theoretically restore the scene representation at any past frame based on the latest set of Gaussians, even after modifications including splitting and cloning have occurred.

3.4.2 Keyframe Selection

In MonoDy-GS, we replace the covisibility-based keyframe selection strategy used in MonoGS [22] with a constant-interval approach. Specifically, we select a new keyframe every 10 frames to ensure regular updates in dynamic scenes.

Once a keyframe is detected, the frontend initiates a keyframe update request to the backend. During this process, the frontend first sends the latest optimized Gaussians along with the Gaussian trajectories corresponding to the past keyframes to the backend. To manage the computational load and ensure efficiency, we do not transmit the Gaussian trajectories for all past keyframes when their number exceeds a certain threshold. Instead, following the approach in MonoGS [22], we maintain a small window of keyframes $\mathcal{W}_k$, which contains a carefully selected subset of keyframes based on inter-frame covisibility.

In addition to the normal keyframe updates, MonoDy-GS also executes *pseudo-keyframe updates* regularly. After Gaussian tracking for each non-keyframe, the frontend requests a pseudo-keyframe update, treating the latest frame as a temporary keyframe that is appended to the active keyframe window $\mathcal{W}_k$. This process ensures that the backend receives the most up-to-date Gaussian motions, which in turn enhances the accuracy of the Gaussian representation maintained in the frontend.

3.4.3 Mapping with Dynamic Gaussians

When the frontend requests a keyframe update, the backend is responsible for refining the Gaussian representation using multi-view information from the selected keyframes. The transition from a completely static setting assumed by MonoGS [22] to a partly dynamic one introduces the challenge of finding temporal correspondence among moving Gaussians, which complicates the imposition of multi-view constraints.

To address this, we restore the Gaussian poses at past keyframes using Gaussian trajectories, which allows us to apply multi-view constraints across keyframes, even when the Gaussian poses vary temporally. During mapping optimization, the restored Gaussians share all attributes except for position and orientation with the latest Gaussians, which ensures temporal consistency of Gaussian appearance. Specifically, the Gaussian poses at each keyframe in $\mathcal{W}_k$ is restored using the corresponding slice of the Gaussian trajectories. Additionally, Gaussian poses are restored for two randomly chosen inactive keyframes $\mathcal{W}_r$. Unlike in MonoGS, where $\mathcal{W}_r$ is updated every iteration, we execute the random sampling once per keyframe update to reduce GPU memory consumption.

Following the approach employed in MonoGS [22], we minimize the sum of the photometric and depth rerendering losses (Eq. 3.4) computed using the latest and restored Gaussians and an isotropic regularization term that constrains overly elongated Gaussian scales. The isotropic regularization loss $\mathcal{L}_{\text{iso}}$ is defined as

$$\mathcal{L}_{\text{iso}} = \sum_{i=1}^{|\mathcal{G}|} \left\| \boldsymbol{s}^i - \bar{\boldsymbol{s}}^i \cdot \mathbf{1} \right\|_1 , \tag{3.22}$$

where $\boldsymbol{s}_i$ represents the scaling parameters of Gaussian i, and $\bar{\boldsymbol{s}}_i$ is the mean scale across the axes. The overall mapping process can be formulated as the following optimization problem:

$$\min_{\substack{\boldsymbol{T}_k, \boldsymbol{p}_k, \mathcal{G}, \\ \forall k \in \mathcal{W}}} \sum_{k \in \mathcal{W}} \mathcal{L}^k_{\text{RGB-D}} + \lambda_{\text{iso}} \mathcal{L}_{\text{iso}}, \tag{3.23}$$

where $\mathcal{W} = \mathcal{W}_k \cup \mathcal{W}_r$. The optimization is performed over the camera poses $\boldsymbol{T}_k$, Gaussian poses $\boldsymbol{p}_k$ for all $k \in \mathcal{W}$, and other appearance-related attributes of the latest Gaussian $\mathcal{G}$. λ_{iso} is a weight coefficient balancing the relative influence between the two loss terms, and we set $\lambda_{\text{iso}} = 10$ following MonoGS [22].

Chapter 4

Experiments

Our experiments are designed to validate the effectiveness of our method in modeling dynamics in monocular videos. We compare our method against the baseline MonoGS [22], with a focus on assessing photometric rendering quality across several datasets. The datasets include Dynamic Replica [5], TUM RGB-D [29] (dynamic scenes), and Dycheck iPhone [2], each offering unique challenges to our dynamic scene modeling approach.

4.1 Experimental Setup

4.1.1 Datasets

Dynamic Replica We utilize the Dynamic Replica dataset [5], a synthetic dataset designed to simulate dynamic scenes by incorporating moving people and animals into static indoor environments originally captured in the Replica dataset. For our experiments, we selected five scenes from the Dynamic Replica dataset that demonstrate significant object motions. We use the RGB-D images and the ground truth optical flow provided by the dataset. Although Dynamic Replica offers stereo data, we only use the left images to create a monocular setting.

TUM RGB-D (Dynamic Scenes) The TUM RGB-D [29] is a real-world dataset collected for evaluation of RGB-D SLAM systems. Undesirable factors such as motion blur make it a relatively challenging dataset. We specifically select scenes from TUM RGB-D that involve dynamic objects, such as moving humans. For our evaluation, we focus on two scenes, *sitting_halfsphere* and *walking_halfsphere*, chosen based on the extent of camera motions and sequence lengths. The RGB-D input from these dynamic scenes is used in our experiments.

Dycheck iPhone Dataset The Dycheck iPhone dataset [2] is a real-world dataset designed to emulate casual captures using an iPhone. It is characterized by quick object motions relative to camera motions, which result in minimal multi-view cues for the dynamic parts of the scene. The dataset provides RGB-D input across several scenes, with seven of these scenes including both training and validation videos. The validation videos, shot from a fixed viewpoint, are used to evaluate the quality of novel view synthesis. We found that some of the scenes of this dataset demonstrate non-negligible drifts in the ground truth camera poses. For this reason, we exclude two scenes, *teddy* and *wheel*, resulting in five scenes in total.

4.1.2 Baselines

We compare our method with MonoGS [22], which is designed for static scenes. Our goal is to demonstrate superior reconstruction quality at dynamic objects. For a fair comparison, we also provide ground truth camera poses to the baseline MonoGS in the following experiments. Although there are recent offline methods that excel in modeling dynamics from monocular videos, we do not include them in our comparisons due to the additional challenges introduced by the online nature of our method.

4.1.3 Metrics

We evaluate the photometric rendering quality using PSNR, SSIM, and LPIPS for the Dynamic Replica and TUM RGB-D datasets. These metrics are computed online every fifth frame of the training images and averaged across the sequence to assess overall performance.

For the Dycheck iPhone dataset, we use mPSNR and mLPIPS, which are standard metrics introduced by Gao et al. [2], to evaluate novel view synthesis quality. These metrics are computed only on areas indicated by a covisibility mask and are evaluated once per scene after processing the entire sequence to maximize viewpoint coverage.

4.1.4 Implementation Details

All experiments are conducted using a single NVIDIA GeForce RTX 3090 GPU. The relative weights of each loss term in Eq. 3.21 are configured as follows: $\lambda_{\text{flow}} = 1.0$, $\lambda_{\text{phys}} = 2.0$, $\lambda_{\text{sparse}} = 10.0$, and $\lambda_{\text{TV}} = 1.0 \times 10^{-5}$. These hyperparameters are kept consistent across all sequence.

4.2 Results

4.2.1 Dynamic Replica

In Figure 4.1, we present an example of qualitative visual comparison between the ground truth image, the MonoGS rendering, and the MonoDy-GS rendering, all from one of the training views of Dynamic Replica [5]. We can see that the MonoDy-GS (Fig. 4.1c) tracks the object motion better (*e.g.* hand, head, shadow). In comparison, the corresponding parts rendered by the MonoGS (Fig. 4.1b) are more blurred and in less accurate positions.

In Table 4.1 we provide a quantitative evaluation of rendering quality of both MonoGS and MonoDy-GS. As can be seen from the table, MonoDy-GS gets constantly superior results over the baseline across all scenes evaluated on, which backs up our observation from the qualitative result. This demonstrates the effectiveness of our Gaussian Tracking method and the revised mapping strategy tailored to modelling dynamic scenes.

4.2.2 TUM RGB-D (Dynamic Scenes)

In Figure 4.2, we present an example qualitative visual comparison between the ground truth image, the MonoGS rendering, and the MonoDy-GS rendering on TUM RGB-D [29]. Through the comparison of the ground truth image (Fig. 4.2a) and the corresponding image rendered by MonoGS (Fig. 4.2b), we can see that the baseline method fails to capture the movement of the person's right hand, and suffers from blurry artefacts around the person's head, which is also slightly moving at the moment. In contrast, MonoDy-GS (Fig. 4.2c) is able to reflect the latest position of the hand, and also to show finer details of the person's body.

Method	Metric	3c30ce	7bf8c1	216ba3	75643c	b907d5	Avg.
MonoGS [5]	PSNR [dB] ↑	19.85	22.95	21.78	21.97	17.06	20.72
	SSIM ↑	0.822	0.858	0.876	0.901	0.728	0.837
	LPIPS ↓	0.268	0.222	0.207	0.181	0.345	0.245
Ours	PSNR [dB] ↑	**22.43**	**27.96**	**23.55**	**24.50**	**19.83**	**23.66**
	SSIM ↑	**0.850**	**0.927**	**0.900**	**0.916**	**0.804**	**0.879**
	LPIPS ↓	**0.222**	**0.114**	**0.148**	**0.155**	**0.269**	**0.182**

Table 4.1: **Comparison of training-view rendering quality on Dynamic Replica.** The results are evaluated online on every fifth frame of the training sequences. An upward arrow (↑) indicates that higher values are better, while a downward arrow (↓) indicates that lower values are better.

Method	Metric	sitting_halfsphere	walking_halfsphere	Avg.
MonoGS [5]	PSNR [dB] ↑	14.10	12.72	13.41
	SSIM ↑	0.481	0.417	0.449
	LPIPS ↓	0.528	0.558	0.543
Ours	PSNR [dB] ↑	**19.70**	**14.92**	**17.31**
	SSIM ↑	**0.680**	**0.496**	**0.588**
	LPIPS ↓	**0.350**	**0.497**	**0.423**

Table 4.2: **Comparison of training-view rendering quality on TUM RGB-D (Dynamic Scenes).** The results are evaluated on every fifth frame of the dynamic sequences. An upward arrow (↑) indicates that higher values are better, while a downward arrow (↓) indicates that lower values are better.

In Table 4.2, we provide quantitative comparisons evaluated on every fifth frame of the training sequences. Again, MonoDy-GS outperforms the baseline in both scenes, which validates the effectiveness of our method in real scenes. However, it is worth noting that MonoDy-GS has much less advantage over the baseline when evaluated on the scene *walking_halfsphere* than on *sitting_halfsphere*. We anticipate that this is because *walking_halfsphere* involves faster motions, which MonoDy-GS can sometimes struggle to catch up with.

4.2.3 Dycheck iPhone

For the Dycheck iPhone dataset [2], we focus on novel view synthesis results from a fixed validation view. We show in Figure 4.3 a qualitative visual comparison. We can see that both MonoGS and our MonoDy-GS fail to update the latest physical pose of the person.

In Table 4.3, We present quantitative results evaluated on the fixed novel views. For the Dycheck iPhone dataset, our method is only almost on par with, or even slightly worse overall, than the baseline. According to Gao et al. [2], the *effective multi-view factor* of the entire dataset is 0.2, which means the camera motion is on average only 0.2 times as fast as the scene motion. This shows that our method is still incapable of capturing faster scene motion relative to the camera motion because of the lack of effective multi-view cues.

4.3 Limitations

4.3.1 Poor Spatial and Temporal Motion Consistency

One of the primary limitations of MonoDy-GS is the poor spatial and temporal consistency in Gaussian motions, particularly in scenes with fast-moving dynamic parts. Despite implementing regularizations to

Method	Metric	apple	block	paper-windmill	space-out	spin	Avg.
MonoGS [5]	mPSNR [dB] ↑	22.61	**14.91**	**14.63**	14.90	**16.81**	**16.77**
	mLPIPS ↓	0.378	**0.435**	0.350	0.402	**0.277**	**0.368**
Ours	mPSNR [dB] ↑	**23.09**	13.83	14.51	**15.42**	13.81	16.13
	mLPIPS ↓	**0.370**	0.463	**0.345**	**0.385**	0.384	0.389

Table 4.3: **Comparison of NVS performance on Dycheck iPhone [2].** The results are evaluated on fixed novel views. An upward arrow (↑) indicates that higher values are better, while a downward arrow (↓) indicates that lower values are better.

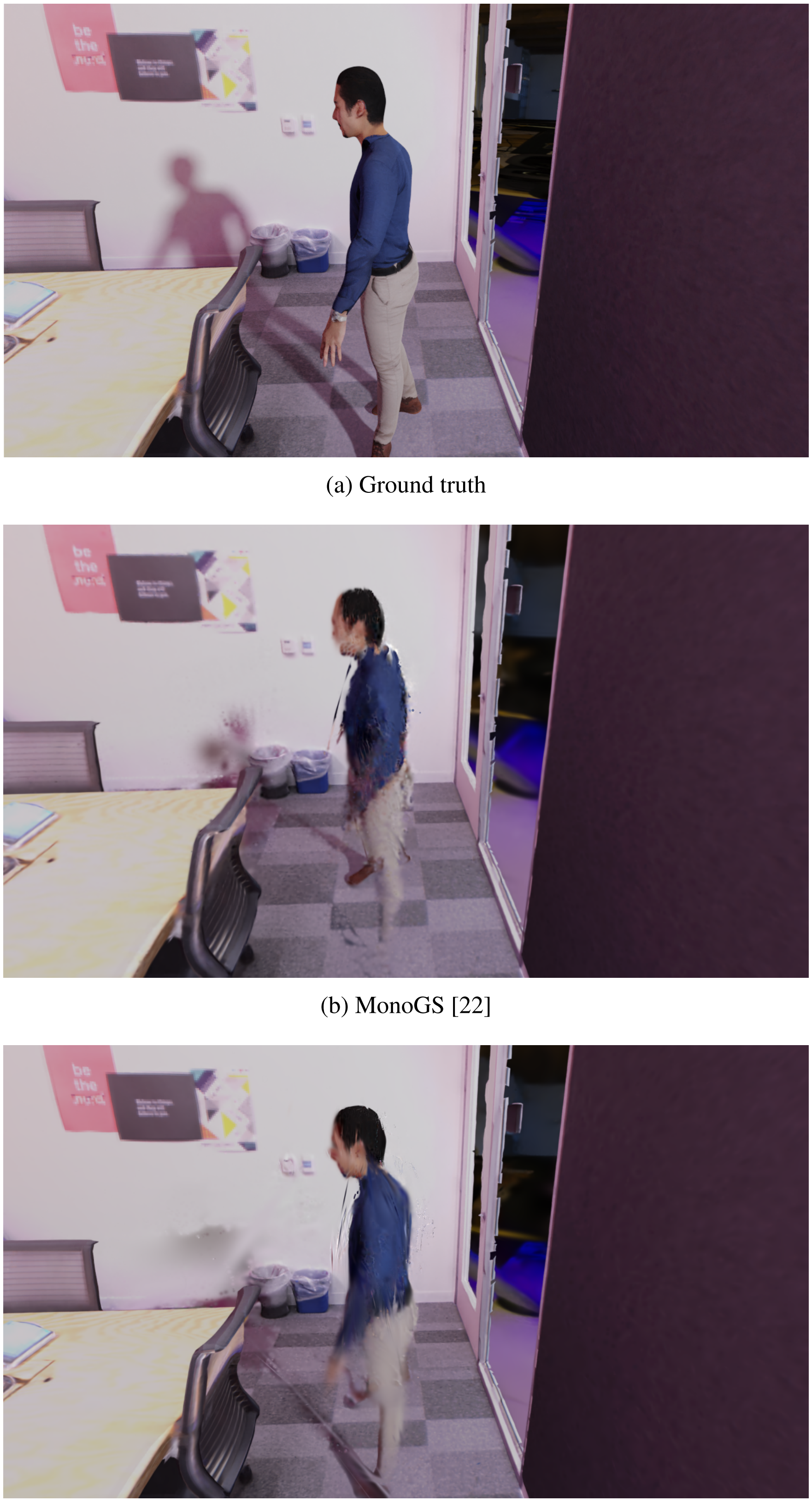
encourage local rigidity and natural object motion, certain dynamic elements deteriorate quickly over a few frames of Gaussian tracking, often appearing “torn up” into separate Gaussians (see Figure 4.4).

This issue stems from the excessive degree of freedom in Gaussian motions. The Gaussian tracking problem is inherently ill-posed due to the monocular nature of the input. Many possible solutions can render similar 2D images and optical flow as the ground truths, while lacking meaningful 3D consistency. To address this, future work could focus on reducing the degrees of freedom in Gaussian motions by exploring low-rank representations. Such representations could internally constrain Gaussian motions, leading to more coherent and consistent tracking results.

4.3.2 Poor Geometry

Another significant limitation observed in MonoDy-GS is the poor geometric consistency due to the monocular nature of the task. During experiments, we noticed that Gaussians often overfit to the most recent views, leading to substantial geometric inconsistencies even with slight viewpoint changes (see Figure 4.5). This limitation indicates that even with perfect motion consistency, rendering quality can still suffer from viewpoint sensitivity, which in turn can mislead the Gaussian tracking process.

A potential solution to this issue could involve synthesizing pseudo multi-view priors. For instance, using image warping techniques, RGB images and depth maps could be generated from slightly different viewpoints. Enforcing the Gaussians to fit not only the original training image but also these synthesized nearby views could potentially help improve geometric robustness. Additionally, more sophisticated depth and surface normal rasterization methods, such as RaDe-GS [40], could be utilized to better constrain the surface geometry.

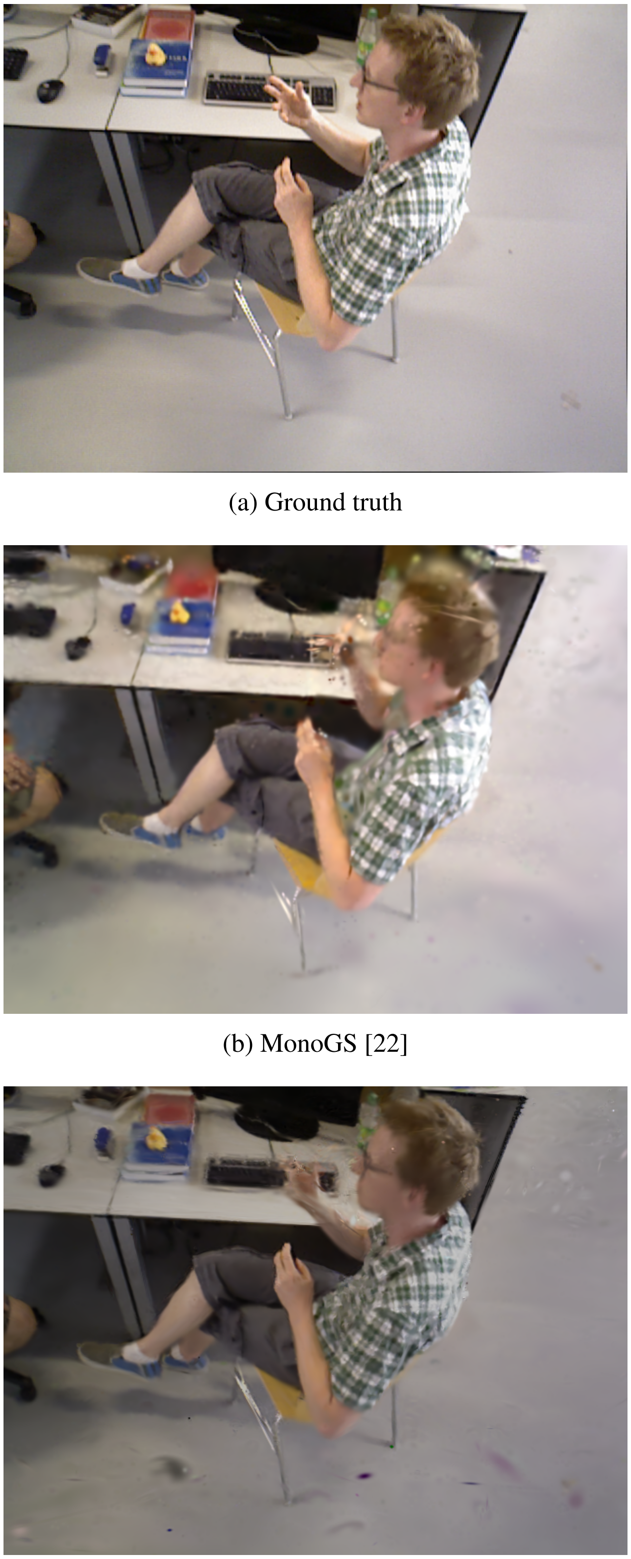


(a) Ground truth

(b) MonoGS [22]

(c) Ours

Figure 4.1: Visual comparison between the ground truth image, the MonoGS rendering, and the MonoDy-GS rendering on Dynamic Replica [5] (frame 100 of scene *75643c*).



(a) Ground truth

(b) MonoGS [22]

(c) Ours

Figure 4.2: Visual comparison between the ground truth image, the MonoGS rendering, and the MonoDy-GS rendering on TUM RGB-D (*sitting_halfsphere*, frame 110).

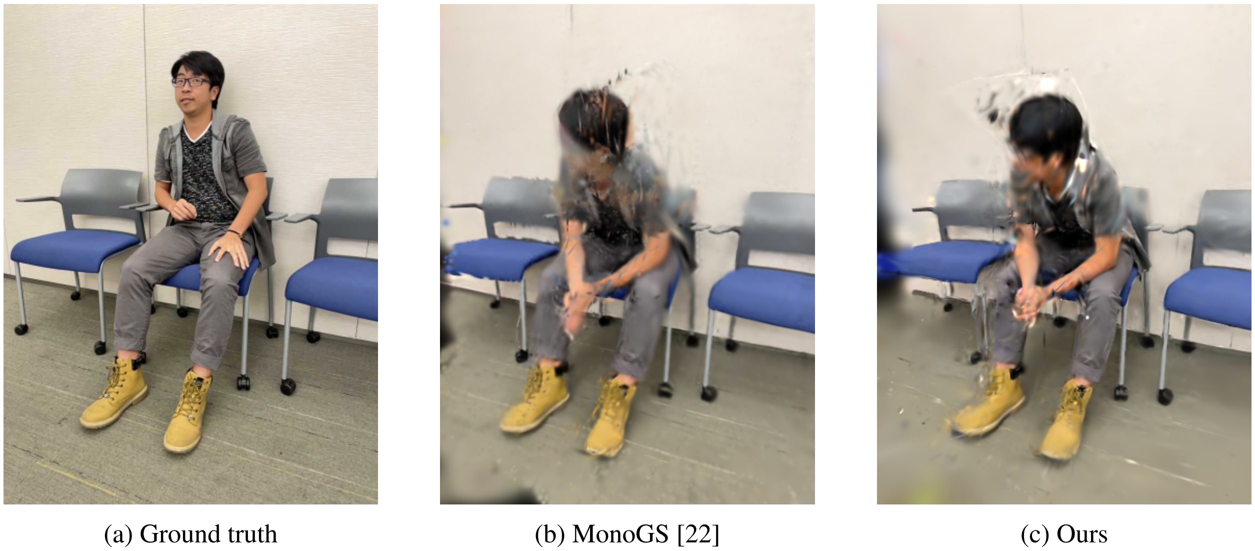

(a) Ground truth
(b) MonoGS [22]
(c) Ours

Figure 4.3: Visual comparison between the ground truth validation image, the MonoGS novel-view rendering, and the MonoDy-GS novel-view rendering on Dycheck iPhone dataset [2] (final frame of scene *space-out*).

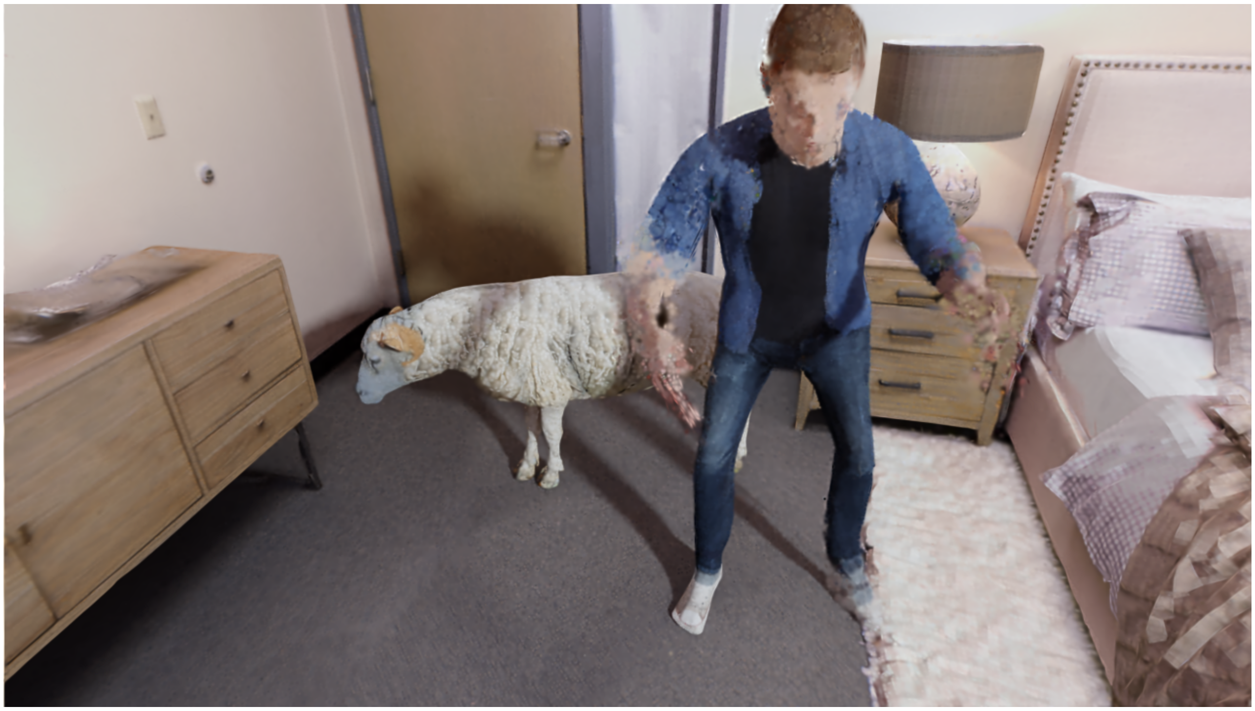
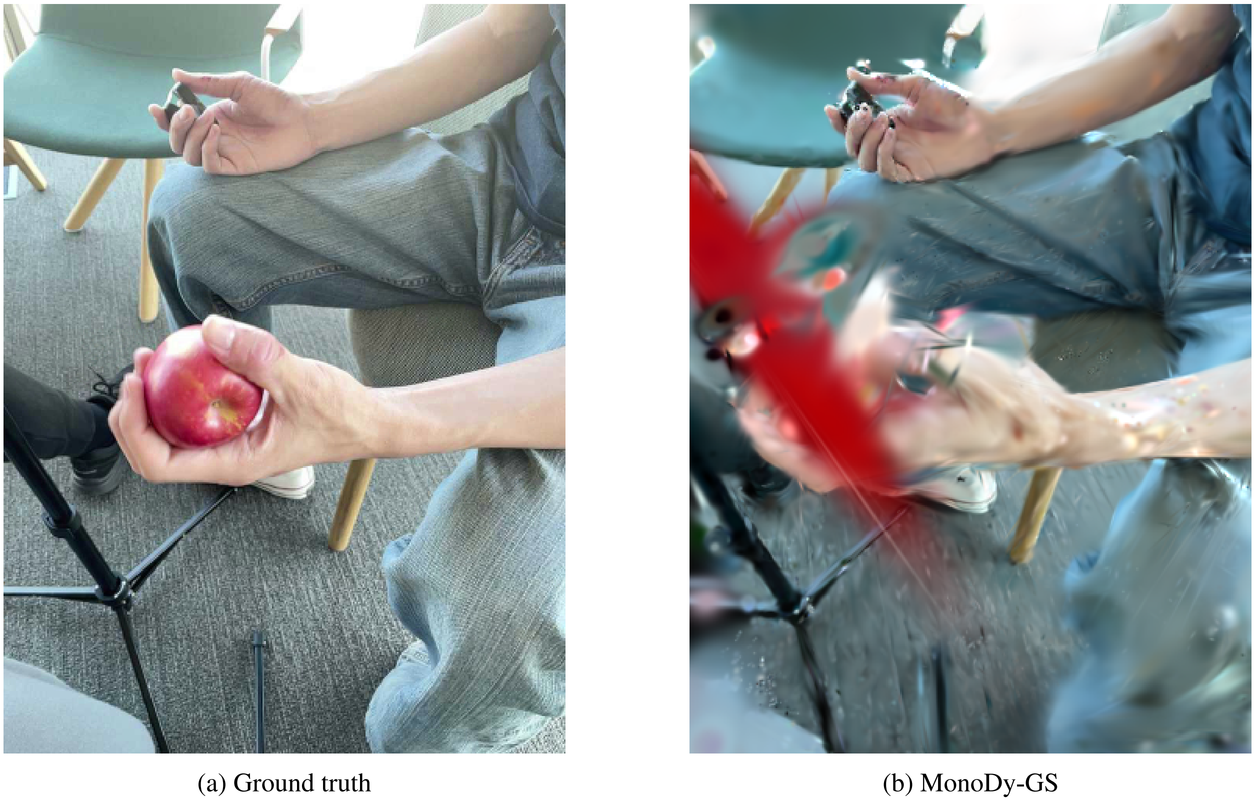


Figure 4.4: **Example of "torn up" effects.** The person's hands, which are the parts showing the quickest motion in the scene, are torn up into separate Gaussians after Gaussian tracking. Parts with less significant motion (*e.g.* the chest) are tracked more reliably and are thus showing less artefacts. This image is rendered from the scene *3c30ce* of Dynamic Replica.



(a) Ground truth

(b) MonoDy-GS

Figure 4.5: An example of poor geometry from a novel viewpoint. The geometry of the apple is overfitting the training views and thus showing an elongated artifact.

Chapter 5

Conclusion

In this project, we introduce MonoDy-GS, an online pipeline designed to model dynamic scenes from RGB-D input using 3D Gaussians as the foundational representation. Building upon the existing MonoGS system, we extend its capabilities to handle dynamic scenes by tracking Gaussian motions online.

To enhance the performance of online Gaussian tracking, we introduce several priors grounded in real-world scene dynamics. These priors are crucial in mitigating the challenges posed by the lack of multi-view information inherent in a monocular setup.

The effectiveness of MonoDy-GS is validated through a series of qualitative and quantitative experiments. The results demonstrate the advantage of our method over the original MonoGS in modeling scene deformations in various dynamic scenarios.

Despite these advancements, MonoDy-GS still faces two notable limitations:

- **Poor Spatial and Temporal Motion Consistency:** The current approach struggles to maintain spatially and temporally consistent Gaussian motions, particularly in fast-moving dynamic scenes. A potential solution could involve exploring low-rank representations that internally constrain Gaussian motions, thereby enhancing motion consistency.

- **Poor Geometry:** Due to the inherent challenges of working with monocular inputs, the reconstructed geometry can exhibit inconsistencies with even slight changes in viewpoint. This issue could potentially be addressed by synthesizing pseudo multi-view priors or by using more advanced depth and surface normal rasterization techniques.

Conquering these challenges would not only enhance the accuracy and realism of dynamic scene reconstruction but also open up new possibilities for applications in Mixed Reality environments. By enabling more immersive and interactive experiences, such improvements could significantly contribute to the future of MR technology.

[23]

Bibliography

[1] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d gaussian splatting: Towards efficient novel view synthesis for dynamic scenes. *arXiv preprint arXiv:2402.03307*, 2024.

[2] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In *NeurIPS*, 2022.

[3] Seongbo Ha, Jiung Yeon, and Hyeonwoo Yu. Rgbd gs-icp slam. *arXiv preprint arXiv:2403.12550*, 2024.

[4] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4220–4230, 2024.

[5] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Dynamicstereo: Consistent dynamic depth from stereo videos. *CVPR*, 2023.

[6] Kai Katsumata, Duc Minh Vo, and Hideki Nakayama. An efficient 3d gaussian representation for monocular/multi-view dynamic scenes. *arXiv preprint arXiv:2311.12897*, 2023.

[7] Kai Katsumata, Duc Minh Vo, and Hideki Nakayama. A compact dynamic 3d gaussian representation for real-time dynamic view synthesis, 2024.

[8] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat track & map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21357–21366, 2024.

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

[10] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[11] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. *arXiv preprint arXiv:2312.00112*, 2023.

[12] Jiahui Lei, Yijia Weng, Adam Harley, Leonidas Guibas, and Kostas Daniilidis. Mosca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. *arXiv preprint arXiv:2405.17421*, 2024.

[13] Fang Li, Hao Zhang, and Narendra Ahuja. Self-calibrating 4d novel view synthesis from monocular videos using gaussian splatting. *arXiv preprint arXiv:2406.01042*, 2024.

[14] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8508–8520, 2024.

[15] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[16] Yiqing Liang, Numair Khan, Zhengqin Li, Thu Nguyen-Phuoc, Douglas Lanman, James Tompkin, and Lei Xiao. Gaufre: Gaussian deformation fields for real-time dynamic novel view synthesis. *arXiv preprint arXiv:2312.11458*, 2023.

[17] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21136–21145, 2024.

[18] Isabella Liu, Hao Su, and Xiaolong Wang. Dynamic gaussians mesh: Consistent mesh reconstruction from monocular videos. *arXiv preprint arXiv:2404.12379*, 2024.

[19] Qingming Liu, Yuan Liu, Jiepeng Wang, Xianqiang Lv, Peng Wang, Wenping Wang, and Junhui Hou. Modgs: Dynamic gaussian splatting from causually-captured monocular videos. *arXiv preprint arXiv:2406.00434*, 2024.

[20] Zhicheng Lu, Xiang Guo, Le Hui, Tianrui Chen, Min Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 3d geometry-aware deformable gaussian splatting for dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8900–8910, 2024.

[21] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024.

[22] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM. 2024.

[23] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[24] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021.

[25] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021.

[26] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021.

[27] Richard Shaw, Jifei Song, Arthur Moreau, Michal Nazarczuk, Sibi Catley-Chandar, Helisa Dhamo, and Eduardo Perez-Pellitero. Swags: Sampling windows adaptively for dynamic 3d gaussian splatting. *arXiv preprint arXiv:2312.13308*, 2023.

[28] Colton Stearns, Adam Harley, Mikaela Uy, Florian Dubost, Federico Tombari, Gordon Wetzstein, and Leonidas Guibas. Dynamic gaussian marbles for novel view synthesis of casual monocular videos. *arXiv preprint arXiv:2406.18717*, 2024.

[29] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[30] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow, 2020.

[31] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. 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*, pages 12959–12970, 2021.

[32] Diwen Wan, Ruijie Lu, and Gang Zeng. Superpoint gaussian splatting for real-time high-fidelity dynamic scene reconstruction. *arXiv preprint arXiv:2406.03697*, 2024.

[33] Qianqian Wang, Vickie Ye, Hang Gao, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. 2024.

[34] Shizun Wang, Xingyi Yang, Qiuhong Shen, Zhenxiang Jiang, and Xinchao Wang. Gflow: Recovering 4d world from monocular video. *arXiv preprint arXiv:2405.18426*, 2024.

[35] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20310–20320, 2024.

[36] Liuyue Xie, Joel Julin, Koichiro Niinuma, and Laszlo A Jeni. Gaussian splatting lk. *arXiv preprint arXiv:2407.11309*, 2024.

[37] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19595–19604, 2024.

[38] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20331–20341, 2024.

[39] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting. *arXiv preprint arXiv:2312.10070*, 2023.

[40] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. *arXiv preprint arXiv:2406.01467*, 2024.