

Masterarbeit

Compositional Gaussian Splatting for Multi-Object and Dynamic Scene Synthesis

Luca Hebeda

Abschlussarbeit
zur Erlangung des akademischen Grades
Master of Science

Vorgelegt von Luca Hebeda
am 31. Oktober 2025
Hochschullehrer/in Prof. Dr. Marcus Vetter
Zweikorrektor/in M. Sc. Yannick Bukschat

Schriftliche Versicherung laut Studien- und Prüfungsordnung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mannheim, 31. Oktober 2025

Luca Hebeda

Abstract

We present a modular and methodologically transparent pipeline for multi-view and temporally consistent synthetic dataset generation based on Gaussian Splatting. The framework integrates static 3D Gaussian Splatting (3DGS) and persistent Dynamic 3D Gaussian Splatting (D3DGS) into a unified system that enables reconstruction, export, and recomposition of dynamic multi-object scenes. Using a hardware-synchronized 56-camera rig, we capture spatially dense and temporally aligned sequences that serve as the basis for static background reconstruction and dynamic object modeling. A dedicated compositing stage allows the exported Gaussian models to be reinserted, duplicated, or combined into novel virtual environments, producing multi-view renderings with synchronized RGB, depth, segmentation, and occlusion annotations. Additional modules provide pose estimation, keypoint propagation, and temporal supersampling for motion-consistent supervision signals. Qualitative evaluations demonstrate that the pipeline maintains spatial coherence, robust occlusion reasoning, and temporal stability across views and environments. The system establishes a practical foundation for controllable 4D dataset generation and for benchmarking methods in dynamic scene understanding.

Contents

1	Introduction	3
2	Fundamentals	5
2.1	Neural Radiance Fields	5
2.2	Gaussian Splatting	6
2.2.1	Scene Representation with 3D Gaussians	6
2.2.2	Rendering with 3D Gaussian Splatting	7
2.2.3	Optimization	9
2.2.4	Differentiable Rasterizer	9
3	State of the Art	12
3.1	Gaussian Splatting for Scene Representation	12
3.2	Dynamic Extensions of Gaussian Splatting	12
3.3	Synthetic Compositing and Dataset Pipelines	13
3.4	Gaussian Splatting for Synthetic Dataset Generation	14
3.5	Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis	15
4	Methodology	18
4.1	Pipeline A: Reconstruction of 3D and Dynamic 3D Models	18
4.1.1	Capture setup and calibration	18
4.1.2	Data Preparation: Mask Generation and Prompt Propagation .	19
4.1.3	3D / Dynamic-3D Gaussian Model Training	20
4.2	Pipeline B: Composition and Synthetic Dataset Generation	20

4.2.1	Scene configuration and class indexing	21
4.2.2	Camera definition and interpolation	21
4.2.3	Placement duplication and motion handling	21
4.2.4	Mask rendering and mask types	21
4.2.5	6D pose estimation	22
4.2.6	Keypoint detection and propagation	22
4.2.7	Rendered outputs and metadata	23
5	Results	24
5.1	Compositional Scene Generation	24
5.2	Multi-view Occlusion Handling	25
5.3	Scene Generalization	25
5.4	Temporal Consistency in Dynamic Scenes	27
5.5	Qualitative Pose Estimation	27
5.6	Keypoint Propagation	28
6	Summary and Outlook	29
6.1	Summary of Key Findings	29
6.2	Limitations	30
6.3	Future Developments and Research Perspectives	30
	Literaturverzeichnis	32

Chapter 1

Introduction

Artificial intelligence has made remarkable progress in understanding and generating visual content [21, 20]. Modern computer vision systems can recognize objects, estimate depth, and reconstruct three-dimensional scenes from ordinary images [17, 50]. Behind these capabilities lies an essential component: the data used to train and evaluate such models [36]. Most deep learning approaches rely on large collections of labeled data, yet creating high-quality datasets with accurate geometry and temporal consistency remains one of the most expensive and time-consuming parts of research and development [11, 25].

In recent years, the representation of visual scenes has shifted from traditional image-based techniques toward neural and hybrid three-dimensional representations. Neural scene representation aims to describe a scene not as a set of disconnected pixels but as a continuous spatial structure that can be viewed from any angle [28]. Methods such as Neural Radiance Fields (NeRFs) have shown that it is possible to reconstruct continuous three-dimensional models from ordinary photographs, although they often require long optimization times and complex rendering processes [1].

Gaussian Splatting has recently emerged as a faster and more transparent alternative [18]. Instead of relying on deep implicit functions, it models a scene as a collection of Gaussian primitives that represent color, position, and opacity in space. This structure allows real-time rendering while maintaining a high level of detail and realism [12]. Because Gaussian primitives can be easily manipulated and recombined, this approach opens new possibilities for modular scene generation and synthetic data creation [39].

The ability to generate synthetic data is becoming increasingly important for artificial intelligence [38, 7]. Real-world datasets are often limited by annotation errors, missing

labels, and a lack of variety [10]. Synthetic datasets can overcome these problems by providing unlimited, precisely labeled, and perfectly synchronized examples [48]. They are especially valuable in areas where ground-truth information such as depth, motion, or segmentation is difficult to obtain through manual annotation [13, 4]. For training robust AI models, it is not only the amount of data that matters but also the quality, consistency, and diversity of the examples. A system that can automatically generate realistic and correctly annotated images from structured three-dimensional representations therefore offers a significant advantage for both academic research and industrial applications.

Despite this potential, most existing methods focus on reconstructing individual static or dynamic scenes [27, 44]. They do not provide tools for composing multiple objects or actors into coherent multi-view environments with synchronized annotations. In many practical scenarios such as robotics, autonomous systems, or simulation, it is essential to create dynamic scenes that include several interacting elements while maintaining accurate geometry and temporal alignment [39].

This thesis addresses these challenges by developing a modular pipeline based on static and dynamic Gaussian Splatting. The proposed system allows the reconstruction, composition, and rendering of complex multi-object scenes in calibrated camera setups. It produces synchronized outputs that include color images, depth maps, and instance-level segmentation. The framework also supports additional annotations such as keypoints, occlusion scores, and estimated poses, which makes it a complete solution for generating structured and reproducible synthetic datasets.

The approach emphasizes compositional reliability rather than algorithmic novelty. By integrating existing Gaussian Splatting methods into a unified workflow, it demonstrates how realistic and temporally consistent synthetic data can be created with minimal manual effort. The system thus connects the recent progress in neural scene representation with the practical need for scalable dataset generation.

The remainder of this thesis presents the theoretical background, the design of the proposed pipeline, and a qualitative evaluation of its results. The findings show that Gaussian-based representations can serve not only as tools for rendering but also as a foundation for controllable four-dimensional data generation and future research in dynamic scene understanding.

Chapter 2

Fundamentals

The following chapter explains the fundamental theoretical concepts required to understand this thesis.

2.1 Neural Radiance Fields

Neural Radiance Fields (NeRF), introduced by Mildenhall et al. [28], represent a decisive step in the development of novel 3D scene representations. The approach models a scene implicitly through a neural network that predicts volumetric density and color for a given position and viewing direction. By employing differentiable rendering techniques and optimizing on multi-view image data, it becomes possible to reconstruct novel viewpoints with high quality. The underlying idea is illustrated in Figure 2.1.

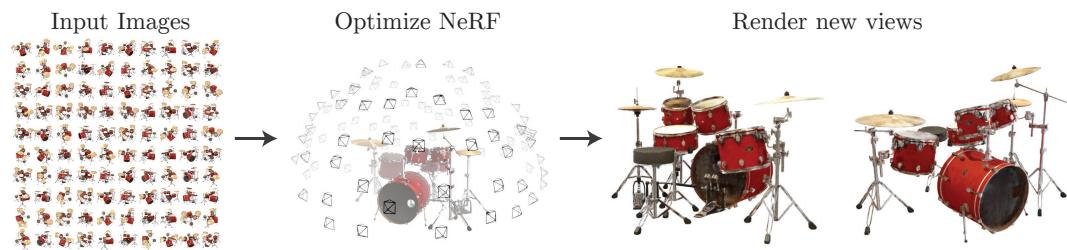


Figure 2.1: Core principle of NeRF: A model is optimized from input images and can subsequently generate novel views of the scene (after [28]).

The model receives a 5D vector (x, y, z, θ, ϕ) as input, consisting of the 3D position and the ray direction. For each combination of position and direction, the MLP outputs a value for density σ and the associated color (r, g, b) . Rendering is performed by

integrating the color and density values estimated by NeRF along the rays projected from the camera into the scene. The scene representation is optimized by minimizing the photometric error between the rendered and the actual image. The pipeline is illustrated in Figure 2.2.

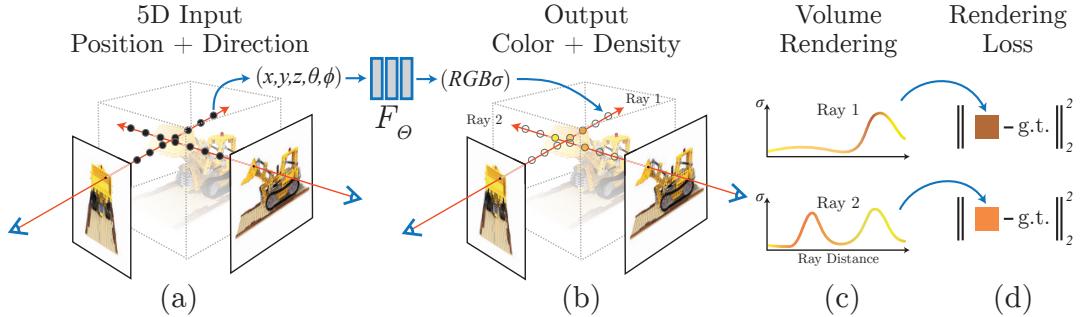


Figure 2.2: NeRF pipeline: The MLP maps 5D inputs to color and density, which are composed into an image via volumetric rendering (after [28]).

NeRF enables detailed reconstruction and accurate depiction of complex lighting effects such as reflections and transparencies, making it particularly suitable for photorealistic applications. Furthermore, the implicit representation ensures consistent multi-view renderings that allow robust synthesis of novel perspectives.

2.2 Gaussian Splatting

While NeRF enables high rendering quality, it suffers from long training times and slow rendering, which is particularly problematic for dynamic scenes and interactive applications. This motivated the development of alternative representations that are more explicit and efficient. One of the most significant works in this area is 3D Gaussian Splatting (3DGS) by Kerbl et al. [18]. The approach replaces NeRF’s implicit network representation with a set of explicit 3D Gaussians that can be rendered in real time while maintaining high image quality. An overview of this approach is shown in Figure 2.3, which illustrates the optimization and rendering pipeline of Gaussian Splatting. The following sections discuss the detailed components of this model to provide a comprehensive understanding of its operation.

2.2.1 Scene Representation with 3D Gaussians

In contrast to NeRF, which requires images from multiple viewpoints as well as camera poses from multi-view stereo (MVS) [33], Gaussian Splatting is based on

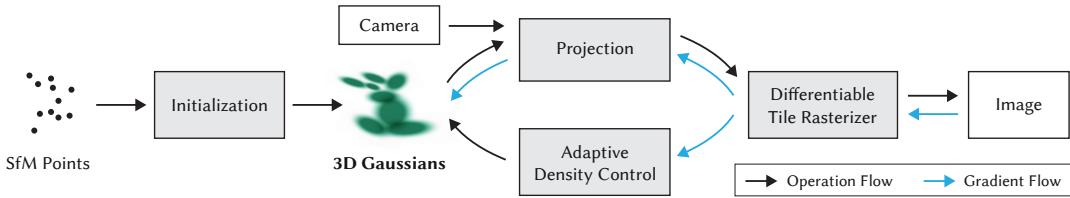


Figure 2.3: Optimization and rendering pipeline of Gaussian Splatting: The process begins with a sparse structure-from-motion (SfM) point cloud, which serves as the basis for creating an initial set of 3D Gaussian functions. These Gaussians are refined through iterative optimization, with their density adaptively controlled to ensure accurate scene representation. A fast tile-based rasterizer enables competitive rendering times compared to modern radiance field methods (figure from [18]).

a sparse point cloud and camera poses obtained via structure-from-motion (SfM) techniques such as COLMAP [32]. These points form the basis for creating a set of 3D Gaussian functions, each defined by its position (μ), an anisotropic covariance matrix (Σ), an opacity (α), and spherical harmonic coefficients for color representation. The 3D Gaussian functions, referred to as *3D Gaussians* throughout this work, are differentiable and can be efficiently projected into 2D splats, allowing fast α -blending for rendering. The 3D Gaussians are described by a covariance matrix Σ in world coordinates [18], which defines their shape and orientation in 3D space:

$$G(x) = e^{-\frac{1}{2}x^T \Sigma^{-1} x} \quad (2.1)$$

The matrix Σ describes the extent and correlation of the Gaussian along different axes, providing a compact and flexible representation of the scene. During rendering, each Gaussian is weighted by its opacity α to contribute to the final image.

2.2.2 Rendering with 3D Gaussian Splatting

Rendering 3D Gaussians onto the 2D image plane requires several transformations and optimizations to ensure both accuracy and efficiency. The approach proposed by Zwicker et al. [51] provides a robust framework that is central to the 3D Gaussian Splatting technique.

Projection of 3D Gaussians

The projection of 3D Gaussians onto the 2D image plane is based on affine transformations and the computation of the covariance matrix in camera coordinates. The original covariance matrix Σ describes the spatial distribution of the Gaussians in 3D space. To project it onto the image plane, a view transformation W is applied, converting the covariance matrix into camera coordinates. The resulting transformed covariance matrix Σ' is computed as follows:

$$\Sigma' = JW\Sigma W^T J^T \quad (2.2)$$

Where:

- Σ' : transformed covariance matrix in camera coordinates,
- W : view transformation from world to camera coordinates,
- Σ : original covariance matrix of the 3D Gaussian functions,
- J : Jacobian matrix of the affine projection.

The Jacobian J describes the effect of the affine projection on small variations in 3D coordinates. This transformation maps the spatial distribution of the Gaussians onto the 2D image plane, accounting for the uncertainty of the data points during rendering.

Simplified Covariance Computation

Since the transformation described above can be computationally expensive, a simplified approach is often used, based on scaling and rotation matrices. The covariance matrix Σ can be expressed using a scaling matrix S and a rotation matrix R :

$$\Sigma = RSS^T R^T \quad (2.3)$$

For optimization, scaling and rotation are stored separately: a 3D vector s for scaling and a quaternion q for rotation. These can easily be converted into matrices and combined, with the quaternion q being normalized to ensure a valid unit rotation.

This approach allows efficient computation and optimization of the 3D Gaussian Splatting parameters.

2.2.3 Optimization

Optimization in 3D Gaussian Splatting is based on iteratively rendering the scene and comparing it with training images to minimize projection errors of 3D structures on the 2D image plane. The parameters of the 3D Gaussians, position (μ), covariance matrix (Σ), opacity (α), and color coefficients, are adjusted using stochastic gradient descent. Opacity α is constrained to the range [0, 1] using a sigmoid function, while covariance scaling is controlled with an exponential activation function. The covariance matrix is initially estimated as an isotropic Gaussian model based on the distances to the three nearest points.

The optimization employs a combined loss function:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}} \quad (2.4)$$

Here, \mathcal{L}_1 measures absolute pixel differences between rendered and training images, while $\mathcal{L}_{\text{D-SSIM}}$ evaluates structural similarity to preserve fine details. The weighting factor λ balances these two components.

Adaptive Control of 3D Gaussians

Adaptive density control, illustrated in Figure 2.4, optimizes the number and distribution of 3D Gaussians. Every 100 iterations, transparent Gaussians with $\alpha < \epsilon_\alpha$ are removed, and in regions with large position-dependent gradients, small Gaussians are cloned or large ones are split into smaller ones to accurately represent geometry. Every 3000 iterations, density is regularized by resetting α values. Large Gaussians with excessive influence are removed to maintain efficiency. This approach allows for a compact and precise scene representation without additional spatial compression.

2.2.4 Differentiable Rasterizer

The differentiable rasterizer is central to efficient rendering and optimization of scenes in 3D Gaussian Splatting, as it enables real-time performance and high-quality image synthesis [18].

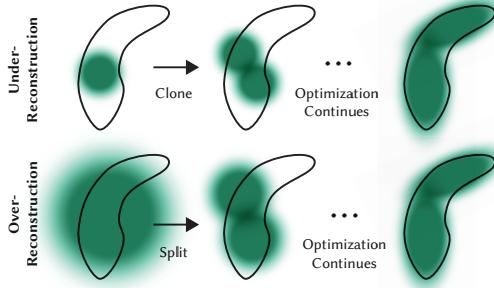


Figure 2.4: Adaptive density control: The top row shows under-reconstruction, where small geometries (black outlines) are supplemented by cloning a Gaussian. The bottom row shows over-reconstruction, where a large Gaussian is split into two smaller ones (after [18]).

Tile-based Rasterization Process

The rasterization projects 3D Gaussians onto the 2D image plane using a view transformation. The influence of each Gaussian on the pixel grid is computed based on its position (μ), covariance (Σ), and opacity (α). A tile-based approach divides the image plane into small tiles processed in parallel on the GPU. The Gaussians are sorted by depth, and their IDs are stored in a list. Sorting ensures correct color and transparency values during the blending process.

Rendering and Blending

During the rendering stage, the individual contributions of all projected Gaussians are accumulated to synthesize the final image. Each Gaussian acts as a semi-transparent, view-dependent surface element whose influence on the pixel color is weighted by its opacity and spatial extent in image space. This process is realized through α -blending, a differentiable compositing operation that models light accumulation along the viewing ray.

Formally, the color of a pixel is obtained by sequentially blending all N Gaussians along the ray in depth order:

$$C = \sum_{i=1}^N T_i \alpha_i c_i, \quad \text{with} \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \quad (2.5)$$

Here, each term in the summation corresponds to a single Gaussian contribution:

- C : final accumulated pixel color,

- T_i : transmittance describing how much light passes through all previous Gaussians,
- α_i : opacity of the i -th Gaussian, determining its visibility,
- c_i : emitted color of the i -th Gaussian.

The transmittance term T_i ensures physically correct ordering along the ray, such that closer Gaussians partially occlude those behind them. This yields a continuous and differentiable approximation of volumetric compositing, analogous to classical volume rendering but without explicit ray integration.

For each pixel (u, v) , the color can be expressed more precisely as:

$$I(u, v) = \sum_{i=1}^N p_i(u, v; \mu_i^{2d}, \Sigma_i^{2d}) \alpha_i c_i(d_i) \prod_{j=1}^{i-1} (1 - p_j(u, v; \mu_j^{2d}, \Sigma_j^{2d}) \alpha_j) \quad (2.6)$$

Here, $p_i(u, v; \mu_i^{2d}, \Sigma_i^{2d})$ denotes the projected 2D Gaussian footprint on the image plane, which determines how strongly each Gaussian contributes to a given pixel. The term $c_i(d_i)$ models view-dependent color variations based on the viewing direction d_i , typically parameterized through spherical harmonics. Finally, the product term $\prod_{j=1}^{i-1} (1 - p_j \alpha_j)$ ensures correct visibility by progressively attenuating the contributions of Gaussians located behind others along the ray.

Backward Pass and Gradient Computation

Since the entire rendering pipeline is differentiable, gradients of the image reconstruction loss can be propagated back through the rasterization process. During the forward pass, the rasterizer stores intermediate quantities such as accumulated transmittance and per-pixel α -values, which are reused in the backward pass to efficiently compute derivatives. This allows for the calculation of gradients with respect to Gaussian parameters including position μ , covariance Σ , opacity α , and color coefficients.

By leveraging these stored buffers, the system avoids redundant recomputation of visibility and blending terms, substantially accelerating optimization. This differentiable rasterization design enables end-to-end training through standard gradient descent, directly optimizing the spatial distribution and appearance of Gaussians based on image-space supervision.

Chapter 3

State of the Art

3.1 Gaussian Splatting for Scene Representation

Gaussian Splatting has recently emerged as a compact and efficient method for high-quality novel-view synthesis. It provides a practical alternative to volumetric or implicit representations such as Neural Radiance Fields. [28, 1, 2]. In GS, a scene is modeled as a set of anisotropic 3D Gaussian primitives, each parameterized by position, covariance, color, and opacity, which are projected and rasterized into image space.

This representation provides both high rendering fidelity and real-time performance while avoiding the computationally expensive optimization and ray marching inherent to NeRF-based methods [18]. Because Gaussian primitives can be directly exported, edited, and recombined, 3DGS serves as a convenient intermediate representation for object-level modeling and downstream applications. Recent studies have extended the approach with semantic and structural grouping of Gaussian primitives. For instance, Gaussian Grouping [45] demonstrated that semantically coherent clusters of Gaussians can be learned jointly with scene geometry, enabling object-level reasoning and segmentation directly in Gaussian space. Such methods illustrate that Gaussian-based representations can serve as a unified bridge between geometric reconstruction and semantic scene understanding.

3.2 Dynamic Extensions of Gaussian Splatting

Following the success of static 3D Gaussian Splatting (3DGS), a substantial body of work has extended the approach to dynamic scenes. Two main paradigms have

emerged.

The first models temporal evolution explicitly by tracking Gaussian primitives over time. Each Gaussian is associated with a trajectory describing its spatial and appearance changes, enabling temporal coherence without retraining. Early works such as *Dynamic 3D Gaussians* [27] and *Dynamic Gaussian Marbles* [34] follow this paradigm, deriving dense motion fields from the per-Gaussian trajectories. More recent variants such as *Deformable 3DGS* [44] incorporate deformation fields inspired by D-NeRF [30], improving temporal smoothness and geometric consistency through explicit regularization. These methods preserve modularity and interpretability, allowing reconstructed objects to be reused and exported, but they require careful trajectory regularization to avoid drift.

The second paradigm embeds time directly into the Gaussian representation, treating the primitives as 4D entities with a temporal axis. Representative approaches include *4DGS* [43], *SpaceTime Gaussians* [24], and *4D Gaussian Splatting* [40], which employ high-dimensional parameterizations or low-rank factorizations (e.g., K-Planes [12]) to model spatio-temporal changes in position, scale, and orientation. These methods achieve globally consistent reconstructions across time but at the cost of increased computational demand and reduced flexibility. Once trained, 4DGS models are monolithic and difficult to segment, edit, or recombine, making them less suited for object-centric workflows or dataset generation.

From the perspective of dataset generation, the trajectory-based approach offers better modularity and reusability. It allows reconstructed objects to be relocated, duplicated, or combined across scenes, which is essential for large-scale synthetic data pipelines. Although four-dimensional representations achieve elegant temporal consistency, their high computational cost and limited flexibility make them less practical for compositional workflows. For this reason, the pipeline developed in this work adopts a trajectory-based dynamic formulation that balances temporal coherence with scalability and per-object control.

3.3 Synthetic Compositing and Dataset Pipelines

Parallel to developments in neural scene representations, a growing body of work addresses synthetic data generation for computer vision. Approaches vary from simple image-level composition to physically grounded 3D rendering pipelines. Some works create datasets by cutting out segmented objects and pasting them into real backgrounds, optionally guided by depth or matting networks [9, 37, 26, 23]. Others

synthesize entire virtual scenes using graphics engines such as BlenderProc or Unreal Engine, enabling precise control over scene layout, lighting, and annotations [6, 22]. More recent efforts combine learned 3D object models with image-based rendering to generate annotated imagery (RGB, depth, segmentation) for downstream tasks [19, 5, 15, 29].

However, many compositing pipelines still rely on 2D-level heuristics such as monocular depth estimation or per-frame segmentation, which introduce inconsistencies across viewpoints and over time [46, 14]. Promptable segmentation frameworks like SAM [19] have improved mask quality, but their integration into geometry-aware, multi-view pipelines remains limited [3]. Using explicit 3D representations—such as object-level Gaussian models—can mitigate these issues by preserving intrinsic spatial consistency and enabling deterministic occlusion reasoning across views.

Beyond masks and depth, many dataset pipelines incorporate mid-level cues such as 2D keypoints to provide additional semantic structure. Modern detectors based on Mask R-CNN and Detectron2 architectures [16, 41] produce accurate joint locations but remain frame-dependent, often suffering from temporal jitter or occlusion failures. To improve stability, lightweight propagation and filtering techniques are applied to transfer reliable detections across time [3].

3.4 Gaussian Splatting for Synthetic Dataset Generation

Recently, Gaussian Splatting has been explored directly for dataset synthesis. *Gaussian Splatting is an Effective Data Generator for 3D Object Detection* [47] employs geometric transformations to place 3D Gaussian assets in realistic scenes to improve object detection training. Other works have used GS to generate domain-specific datasets, including aerial imagery [35], surgical data [49], and robotic perception scenes [8]. *Cut-and-Splat* [39] further explores cut-and-paste strategies via Gaussian composition.

These developments highlight the potential of Gaussian Splatting as a foundation for spatially and temporally consistent data generation. However, existing pipelines often focus on specific applications and lack a general-purpose system that unifies object extraction, segmentation refinement, temporal propagation, and multi-view composition. The work presented in this thesis addresses this gap by providing a modular and reproducible framework that integrates static and dynamic Gaussian models into coherent multi-view environments with synchronized multimodal annotations.

3.5 Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis

One of the foundational works for dynamic Gaussian Splatting is the method proposed by Luiten et al. [27], which introduces *Dynamic 3D Gaussians* (D3DGS) for representing and tracking dynamic scenes. This method forms the basis for the dynamic object reconstruction component of the pipeline developed in this work. The following section provides a detailed overview of its representation, physical regularization, and optimization strategy.

Scene Representation

In contrast to approaches that model space and time within a shared continuum, D3DGS explicitly tracks Gaussian primitives over time. While 3DGS treats Gaussians as static entities in three-dimensional space, D3DGS extends this framework by allowing their positions and orientations to vary across frames. Intrinsic properties such as color, opacity, and scale remain fixed, resulting in a representation where Gaussians behave like physical particles undergoing rigid-body motion. The method maintains a fixed number of Gaussians per scene rather than dynamically creating or deactivating them over time. This consistency facilitates compositional workflows, as objects can be represented and manipulated uniformly across scenes. To ensure physically plausible and temporally coherent motion, Luiten et al. augment the model with several regularization terms that encode physical priors.

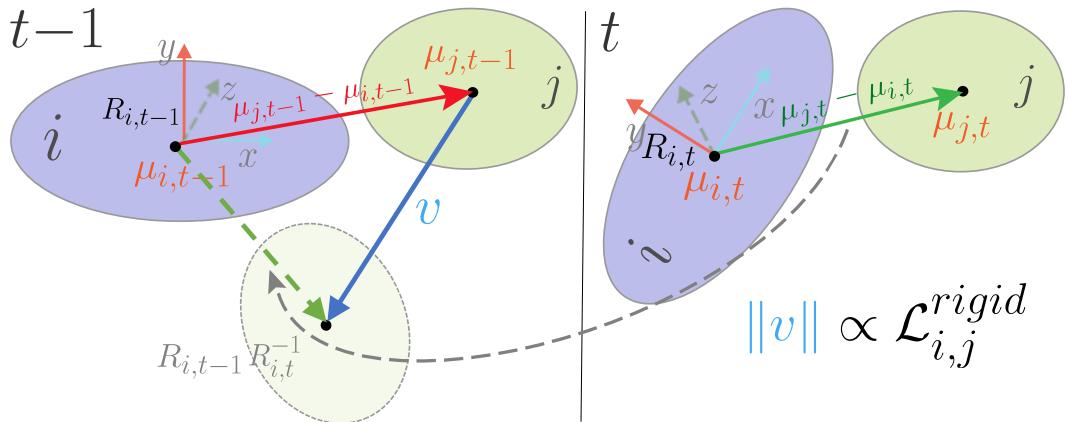


Figure 3.1: Schematic illustration of the local rotation similarity loss in D3DGS [27].

Physical Regularization

The physical regularization ensures that local structures move consistently and undergo realistic motion patterns. Three principal loss terms are introduced to enforce these constraints.

Local Rigidity Loss The rigidity loss enforces that neighboring Gaussians exhibit motion consistent with a local rigid-body transformation. For each Gaussian i , its neighbors j across two consecutive time steps should follow a similar transformation (see Fig. 3.1). The loss is defined as:

$$\mathcal{L}_{\text{rigid}} = \frac{1}{k|S|} \sum_{i \in S} \sum_{j \in k\text{-nn}} w_{i,j} \|(\mu_{j,t-1} - \mu_{i,t-1}) - R_{i,t-1} R_{i,t}^{-1} (\mu_{j,t} - \mu_{i,t})\|^2, \quad (3.1)$$

where the weighting factor $w_{i,j}$ depends on the spatial proximity of two Gaussians in the initial configuration:

$$w_{i,j} = \exp(-\lambda_w \|\mu_{j,0} - \mu_{i,0}\|^2). \quad (3.2)$$

This loss preserves the relative structure of local neighborhoods over time.

Local Rotation Similarity Loss In addition to translation, the rotation of neighboring Gaussians should remain consistent. To achieve this, a rotation similarity loss measures the difference between the quaternions representing the local motion:

$$\mathcal{L}_{\text{rot}} = \frac{1}{k|S|} \sum_{i \in S} \sum_{j \in k\text{-nn}} w_{i,j} \|q_{j,t} q_{j,t-1}^{-1} - q_{i,t} q_{i,t-1}^{-1}\|^2. \quad (3.3)$$

This encourages spatially coherent rotational motion across neighboring Gaussians.

Isometry Loss The isometry loss stabilizes the distances between Gaussians over time, preventing local neighborhoods from deforming excessively. It is defined as:

$$\mathcal{L}_{\text{iso}} = \frac{1}{k|S|} \sum_{i \in S} \sum_{j \in k\text{-nn}} w_{i,j} \left| \|\mu_{j,0} - \mu_{i,0}\| - \|\mu_{j,t} - \mu_{i,t}\| \right|. \quad (3.4)$$

Compared to the more restrictive rigidity loss, this term acts as a softer constraint, allowing moderate flexibility while maintaining approximately constant inter-Gaussian distances.

Optimization

Optimization follows the same principle as static 3DGS and relies on the image reconstruction loss defined in Equation 2.4. Training proceeds sequentially over time steps, leading to a linear increase in computation time with sequence length. A major limitation of this formulation is its reliance on dense multi-view supervision: if certain viewpoints are missing, Gaussians that become occluded or leave the field of view can no longer be reliably tracked, reducing robustness in scenes with partial visibility.

Chapter 4

Methodology

This chapter presents the complete methodology developed for constructing, combining, and rendering Gaussian Splatting models for both static and dynamic 3D scenes. The proposed system is designed to transform synchronized multi-view recordings into modular Gaussian representations and to recompose them into complex, synthetic environments. It consists of two main processing stages: **Pipeline A**, which reconstructs 3D and 4D Gaussian models from calibrated camera data, and **Pipeline B**, which composes these models into structured multi-object scenes for dataset generation. Together, these stages establish a unified framework that bridges real-world capture and synthetic data creation through differentiable Gaussian rendering.

4.1 Pipeline A: Reconstruction of 3D and Dynamic 3D Models

The first stage of the system, referred to as *Pipeline A*, transforms synchronized multi-view image sequences into individual Gaussian Splatting models. It uses a high-density capture rig and follows four main stages: capture and calibration, preprocessing, mask generation, and model training.

4.1.1 Capture setup and calibration

The acquisition takes place in a custom-built multi-camera rig equipped with 56 globally synchronized cameras operating at a resolution of 1920×1200 pixels and a frame rate of 30 frames per second. All cameras are connected to a shared hardware trigger that provides a global clock signal, guaranteeing microsecond-level temporal synchronization across all views. The cameras are arranged in a hemispherical

configuration around a capture volume of approximately $1.5 \times 1.5 \times 2.5$ meters. The geometric center of this volume serves as the global origin for all reconstructions. This setup ensures dense overlapping views and uniform angular coverage, which are essential for complete and consistent 3D and 4D reconstructions.

Calibration is performed in two stages. Intrinsic parameters are estimated for each camera individually following the standard OpenCV pinhole model with five distortion coefficients (k_1, k_2, p_1, p_2, k_3). The intrinsic model comprises focal length, principal point, and both radial and tangential distortion terms. Because the optical center (c_x, c_y) of each lens is not necessarily aligned with the image center, an *optimal new camera matrix* is computed using OpenCV’s rectification routines to balance field of view and minimal distortion. This step yields undistorted, rectified images that serve as input for all subsequent stages. Extrinsic calibration is performed once for the entire rig via a global optimization of inter-camera correspondences obtained from a moving checkerboard sequence. The resulting extrinsic parameters define all cameras within a unified world coordinate frame.

4.1.2 Data Preparation: Mask Generation and Prompt Propagation

Dynamic 3D Gaussian Splatting requires per-frame object masks to disentangle dynamic objects from the static background and to maintain temporal consistency during training [27]. While the reference implementation primarily relies on binary foreground/background masks, this pipeline extends that step by integrating **SAM2** [31], enabling interactive prompting and high-quality instance segmentation across frames.

Segmentation is initialized by selecting a seed frame and providing manual point or box prompts for the objects of interest. These seed prompts are propagated across all cameras in the calibrated multi-view setup. Concretely, for each camera in the sequence the corresponding seed image is used to generate an initial segmentation prompt that is added to the predictor state, ensuring consistent initial object masks across views. Subsequently, masks are propagated temporally across the sequence using the SAM2 video predictor. This procedure yields temporally consistent, per-camera instance masks for all frames of the dataset. Compared to classical foreground/background segmentation, this approach permits fine-grained object delineation and is more resilient to occlusions and appearance changes across time and viewpoints. The resulting masks are fed directly into the D3DGS training pipeline to supervise dynamic object reconstruction.

4.1.3 3D / Dynamic-3D Gaussian Model Training

Per-object Gaussian Splatting models are trained using the rectified images, calibrated poses, and instance masks. Static instances are reconstructed with 3DGS [18]; dynamic instances use persistent D3DGS [27].

The training procedure follows the original methods with two notable deviations. First, Gaussian positions are not initialized from sparse COLMAP point clouds. Because the multi-camera rig is pre-calibrated with known intrinsics and extrinsics, no additional SfM reconstruction is required for each scene. Instead, random point clouds are used to initialize each scene, significantly reducing overhead when training new sequences. Second, all reconstructions share a common global coordinate system, guaranteeing consistent alignment across different captures. Consequently, models trained in separate sequences can be merged directly without additional markers, manual registration, or extra calibration steps.

External 3D assets. In addition to models reconstructed with 3DGS and D3DGS, Pipeline A can incorporate externally generated 3D assets. To demonstrate this, small object models produced with **Trellis** [42] are used; Trellis reconstructs compact GLB meshes from a few input images and can export models as Gaussian representations. Exported Gaussian assets are imported without modification into the composition framework, demonstrating compatibility with heterogeneous 3D sources such as small household objects (e.g., boxes, plants, or tools).

4.2 Pipeline B: Composition and Synthetic Dataset Generation

While Pipeline A focuses on reconstructing individual static and dynamic Gaussian models, Pipeline B composes these models into coherent multi-object scenes. Pipeline B combines exported models, places them in calibrated virtual environments, and produces synchronized outputs such as RGB images, depth maps, and segmentation masks. The pipeline thus connects reconstruction and dataset generation by providing all necessary modalities for synthetic training and evaluation.

4.2.1 Scene configuration and class indexing

A synthetic scene is defined by a structured list of object instances, each associated with placement parameters and camera trajectories used for rendering. Every object model is assigned a fixed class index that remains consistent throughout the dataset. These indices, together with the scene configuration, are recorded in a JSON manifest that serves as the central reference for all annotations. Each instance also receives a unique identifier and a stable display color from a predefined palette, ensuring consistent instance-level segmentation across frames and scenes. Once configuration is complete, the scene can be rendered either from the physical rig cameras or from virtual viewpoints.

4.2.2 Camera definition and interpolation

The pipeline supports both real and virtual camera configurations. Rendering may be performed using the original rig calibration or via interpolated viewpoints that generate smooth camera paths. Physical camera poses are interpolated while averaging intrinsic parameters to obtain continuous motion through the virtual environment. Temporal supersampling can optionally create intermediate frames through sub-frame interpolation of the Gaussian parameters. This yields smoother apparent motion and higher effective frame rates, which are useful for generating realistic video sequences and temporally consistent annotations.

4.2.3 Placement duplication and motion handling

Objects are positioned in the global coordinate system using rigid transformations composed of translation, rotation, and uniform scaling. Dynamic models carry time-dependent Gaussian attributes and are rendered according to their internal trajectories. The system allows objects to be duplicated and repositioned while maintaining unique instance identifiers. This flexibility enables the creation of diverse scene configurations with varying spatial arrangements and interactions between static and dynamic elements. The same set of reconstructed assets can therefore be reused across many compositions without retraining or manual adjustment.

4.2.4 Mask rendering and mask types

During scene composition, each object is rasterized independently into an object identifier buffer while performing depth testing to ensure correct occlusion ordering.

The system exports two complementary mask variants that support different training scenarios. The *visible mask* contains only pixels that remain visible after depth compositing, while the *complete mask* is obtained by rendering the object in isolation and thresholding its depth and alpha values. The complete mask therefore contains the full object silhouette, including regions that may be occluded in the final composition. Providing both mask types ensures compatibility with various downstream pipelines: some models require visible masks for instance segmentation, while others benefit from complete masks for occlusion-aware augmentation.

Occlusion scoring For each frame t and instance i , an occlusion score is computed as

$$s_{\text{occ}}(t, i) = 1 - \frac{|V_{t,i}|}{|C_{t,i}|},$$

where $V_{t,i}$ and $C_{t,i}$ denote the pixel counts of the visible and complete masks, respectively. The score ranges from 0 (fully visible) to 1 (completely occluded). Occlusion scores are stored in the JSON manifest and can be used for filtering, sampling, or weighting during downstream training.

4.2.5 6D pose estimation

Each object is associated with a candidate six-degree-of-freedom transform mapping its canonical coordinates into the global scene frame. Static poses derive from the original reconstruction alignment, while dynamic poses are obtained from per-Gaussian trajectories via centroid and rotation estimation. Although these poses do not constitute absolute ground truth, they provide consistent internal motion estimates that are valuable for pose refinement, tracking, and activity recognition.

4.2.6 Keypoint detection and propagation

The system generates 3D keypoints for each captured dynamic human model. Keypoints for each dynamic human model are initially detected in a reference frame using a Detectron2-based detector [41]. The rasterizer was extended to return per-pixel Gaussian indices, enabling tracking of keypoints over time by associating them with the Gaussians that influence the corresponding 2D pixel locations. To ensure temporal consistency across frames, 2D keypoints are reprojected into 3D using rendered depth maps together with camera intrinsics and extrinsics. These 3D points are subsequently updated in each frame according to the motion of the associated

3D Gaussians, ensuring that keypoints follow the dynamic objects throughout the sequence.

4.2.7 Rendered outputs and metadata

For each rendered frame and camera, the pipeline exports RGB images, depth maps, per-instance visible and complete masks, bounding boxes, and object metadata. All outputs are indexed in a JSON manifest that stores camera parameters, object transformations, occlusion scores, and keypoints, ensuring consistent synchronization across modalities. For validation and manual inspection the pipeline can additionally output diagnostic overlays such as mask overlays, rendered bounding boxes, and projected 6D poses and keypoints.

Together, these components form a unified, reproducible system for scene composition and dataset generation from modular Gaussian-based models.

Chapter 5

Results

This chapter presents a qualitative evaluation of the proposed multi-camera compositing and dynamic scene generation pipeline. In the absence of complete ground-truth annotations, the analysis focuses on visual consistency, temporal coherence, and scene-level generalization across multiple reconstruction and rendering configurations. All results were produced using the same calibrated multi-camera rig under identical rendering parameters, without manual post-processing unless explicitly stated.

5.1 Compositional Scene Generation

Figure 5.1 illustrates how the proposed compositing pipeline integrates multiple independently reconstructed Gaussian Splatting models into a single, globally aligned scene. Each row in the figure shows one configuration step, starting from a single-object placement and progressing toward increasingly populated scenes that combine both static and dynamic objects. For each configuration, the corresponding RGB renderings, depth maps, bounding boxes, and segmentation overlays are presented to demonstrate multi-modal coherence.

A central advantage of the system is that all semantic annotations, including instance masks, class indices, and bounding boxes, are generated automatically and remain geometrically consistent across all rendered views. Since every 3D model is represented as Gaussian primitives within a shared coordinate system, mask generation requires no additional training or inference using segmentation networks. All semantic labels are deterministically derived from the compositing process itself, ensuring reproducible and noise-free annotations.

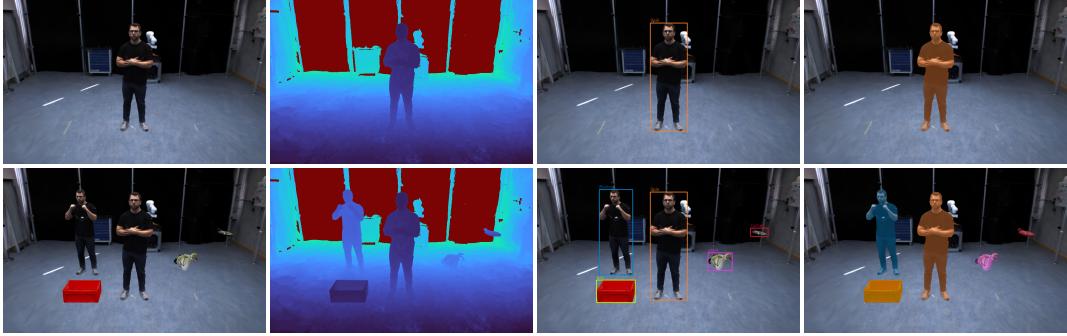


Figure 5.1: Compositional scene generation. Example of multi-object scenes composed from independently reconstructed Gaussian Splatting models. Each row shows one scene configuration with RGB, depth, bounding boxes, and segmentation overlays. All semantic annotations are generated automatically and remain spatially consistent across views.

The modular design of the pipeline further enables high scene variability. Objects can be translated, rotated, scaled, or duplicated directly in 3D space without retraining or manual re-annotation. This flexibility allows large-scale dataset creation with controlled variation in object arrangement and spatial relationships. In practice, the same set of Gaussian models can be reused across numerous novel compositions while preserving consistent geometry, color, and semantic labeling—making the approach well suited for scalable and reproducible synthetic dataset generation.

5.2 Multi-view Occlusion Handling

To assess the robustness of the compositing mechanism under occlusions, Figure 5.2 shows segmentation overlays from two viewpoints of a scene containing interacting dynamic subjects. Even when large parts of one subject are occluded, the instance masks remain spatially consistent and tightly aligned with visible contours. This demonstrates that the rendering process preserves correct inter-object depth ordering and instance integrity across multiple perspectives. Such behavior is crucial for generating reliable training data for tasks such as instance segmentation or 3D scene understanding, where stable occlusion boundaries are essential.

5.3 Scene Generalization

Figure 5.3 demonstrates the adaptability of the proposed system by recontextualizing dynamic subjects within a novel virtual environment reconstructed from real imagery



Figure 5.2: **Occlusion handling across views.** Segmentation overlays from front and side viewpoints of two interacting subjects. Even under strong mutual occlusion, instance masks remain consistent and well aligned with visible contours.

using COLMAP [32]. The reconstructed environment provides realistic geometry and camera poses that serve as a spatial reference for compositing. Dynamic subjects are seamlessly integrated into the new setting, maintaining correct spatial alignment, depth consistency, and segmentation coherence. By decoupling object-level Gaussian models from a specific environment, the system enables arbitrary recombination of dynamic and static assets across scenes without retraining. This generalization capability supports large-scale variation in scene composition and facilitates applications such as domain adaptation, robustness evaluation, and synthetic-to-real transfer.



Figure 5.3: **Scene generalization and recontextualization.** Dynamic subjects composited into a novel virtual environment reconstructed with COLMAP. The system maintains consistent geometry and segmentation alignment across domains, demonstrating flexible recontextualization of pre-trained Gaussian models.

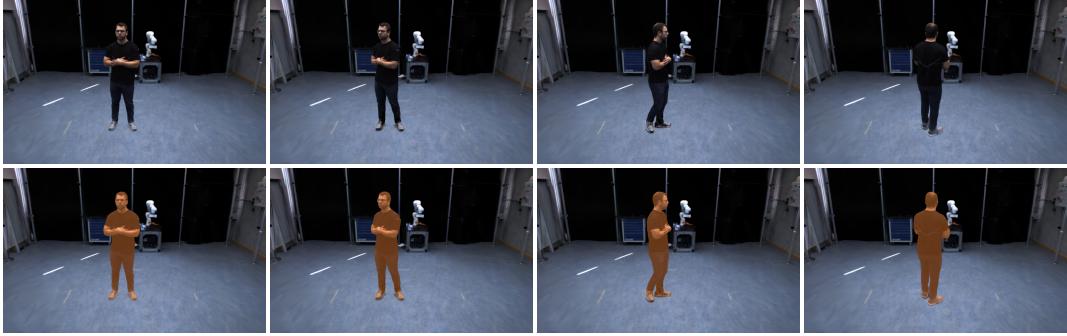


Figure 5.4: **Dynamic scene generation.** Representative time steps of a moving subject with RGB and segmentation overlays. The consistent motion and geometry across frames demonstrate temporally stable 4D rendering.

5.4 Temporal Consistency in Dynamic Scenes

Dynamic scene rendering enables the generation of temporally coherent 4D datasets suitable for applications such as activity recognition, pose estimation, or motion segmentation. Figure 5.4 shows representative frames from a dynamic sequence of a moving subject. The RGB and segmentation overlays reveal smooth and consistent motion over time, with stable geometry and accurate alignment across frames. Each time step preserves object structure and surface continuity, indicating that the Gaussian-based transformation updates produce interpretable and stable temporal behavior. This capability allows scalable synthesis of realistic dynamic sequences without requiring explicit motion capture or manual supervision.

5.5 Qualitative Pose Estimation

Figure 5.5 (top row) shows qualitative 6D pose estimates for a dynamic human sequence across four time steps. Each frame visualizes the canonical object coordinate system derived from the aggregated Gaussian transformations. Although no ground-truth pose data are available, the estimated coordinate axes evolve consistently and reflect plausible orientation changes relative to the subject’s motion. This indicates that Gaussian-based transformation tracking provides a reliable approximation of object-level motion, suitable for downstream tasks such as pose refinement or motion analysis.

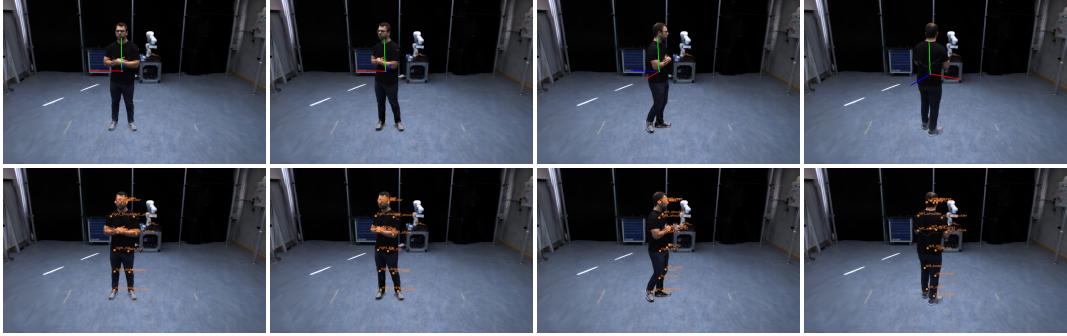


Figure 5.5: Qualitative analysis of pose and keypoint propagation. Four frames of a dynamic subject showing estimated object poses (top) and propagated keypoints (bottom). The coordinate systems evolve consistently over time, while keypoint trajectories remain coherent despite local misalignments.

5.6 Keypoint Propagation

The lower row of Figure 5.5 visualizes the temporal keypoint propagation results obtained with a Detectron2-based detector [41]. Initial 2D keypoints are detected in a reference frame and projected into 3D by associating them with the nearest Gaussian centroids of the corresponding subject. As a result, keypoints are anchored on the object surface rather than the true anatomical joints, but they remain spatially coherent throughout the sequence. The keypoints are subsequently propagated in time according to the motion of the underlying Gaussians.

Temporal stability is observed across most body regions, particularly in the upper body where Gaussian density is high and motion is well captured. Minor deviations appear in areas with sparse or unstable Gaussian coverage, such as the knees or hips, leading to small temporal inconsistencies or spatial offsets. Despite these limitations, the propagation maintains overall semantic coherence and demonstrates that the Gaussian-based motion field can sustain meaningful correspondences over time. This behavior provides a promising foundation for future integration with articulated motion priors or learned joint-space constraints.

Chapter 6

Summary and Outlook

6.1 Summary of Key Findings

This work presented a modular and methodologically transparent pipeline for multi-view dataset generation based on static and dynamic 3D Gaussian Splatting representations. Leveraging a purpose-built, hardware-synchronized multi-camera system, temporally aligned image sequences were captured from which both static (3DGS) and dynamic (D3DGS) models were reconstructed. By integrating promptable segmentation (SAM2) and temporal mask propagation into the reconstruction workflow, all exported object models achieve geometric precision, temporal stability, and direct compatibility with downstream compositing.

A central contribution of the proposed system is the unified treatment of static and dynamic Gaussian models within a single compositional framework. Pipeline B enables deterministic scene assembly in calibrated camera rigs, producing spatially and temporally coherent outputs including RGB, depth, and instance-level masks. By explicitly modeling occlusion, object placement, and temporal interpolation, the pipeline allows full control over rendered datasets while maintaining photometric and geometric realism. This structure bridges the gap between reconstruction-focused Gaussian pipelines and practical dataset generation tools, supporting reproducible, multi-object scenes with synchronized multimodal annotations.

The framework further supports the seamless integration of externally generated 3D assets—such as those exported from *Trellis*—demonstrating that pre-trained or lightweight 3D models can function as first-class entities within Gaussian-based rendering and composition. As a result, the system generalizes across input modalities, accommodating both data-driven and generative 3D sources without modifications

to the rendering logic.

6.2 Limitations

Despite its flexibility, several limitations should be acknowledged. First, dynamic reconstructions rely on per-frame segmentation masks that, despite prompt propagation, may accumulate small temporal inconsistencies under strong occlusions or motion blur. This can result in slight spatial drift of reconstructed Gaussians and propagated keypoints, particularly in regions with sparse Gaussian coverage such as knees or hips.

Second, although the compositional framework supports both static and dynamic elements, all rendered scenes currently assume rigid camera calibration and fixed illumination. Consequently, phenomena such as global illumination, specular reflections, or cast shadows between independently rendered objects are not explicitly modeled, which can limit realism in certain composite configurations.

Third, pose estimation for dynamic objects is derived from Gaussian trajectories rather than explicit joint-based motion models. While this approach ensures coherent motion propagation, it lacks semantic interpretability and may diverge from true skeletal motion.

Finally, the current implementation is optimized for controlled laboratory captures using pre-calibrated multi-camera rigs. Generalizing to in-the-wild or handheld capture setups would require additional mechanisms to handle calibration drift, synchronization errors, and varying illumination conditions.

6.3 Future Developments and Research Perspectives

Several avenues remain open for further exploration. A natural extension involves integrating articulated motion priors and skeleton-aware Gaussian tracking, which could improve alignment of dynamic keypoints and enable physically interpretable pose propagation. Embedding such priors directly into the D3DGS training or composition stages may strengthen temporal coherence in regions with complex non-rigid motion, such as human limbs or deformable surfaces.

Another promising direction involves scaling the capture and synthesis process toward larger and more diverse datasets. Automated scene composition, dynamic lighting control, and procedural environment synthesis could further enhance real-

ism and diversity while maintaining annotation consistency. In parallel, coupling Gaussian-based scene representations with diffusion-based appearance refinement or NeRF-style volumetric augmentation may bridge the remaining visual gap between synthetic and real-world imagery.

Finally, establishing standardized benchmarks for evaluating compositional and temporal consistency in Gaussian-rendered datasets would be highly beneficial. Such benchmarks could promote reproducibility, enable quantitative analysis of temporal stability, and help define a common evaluation protocol for future research in Gaussian-based dataset generation.

Overall, this work demonstrates that temporally consistent Gaussian representations—when combined with geometry-aware compositing and transparent annotation pipelines—provide a practical and extensible foundation for the next generation of large-scale, multi-view training datasets.

Literaturverzeichnis

- [1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.
- [2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20766–20776, 2023.
- [3] Gedas Bertasius and Lorenzo Torresani. Maskprop: Classifying, segmenting, and tracking object instances in video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2019.
- [4] Gedas Bertasius and Lorenzo Torresani. Maskprop: Classifying, segmenting, and tracking object instances in video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 970–978, 2019.
- [5] Gedas Bertasius and Lorenzo Torresani. Classifying, segmenting, and tracking object instances in video with mask propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9739–9748, 2020.
- [6] Markus Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohy Elbadrawy, Ahsan Lodhi, Haritha Katam, Tomas Hodan, Frank Michel, Carsten Rother, and Slobodan Ilic. Blenderproc: A procedural pipeline for photorealistic rendering. *arXiv preprint arXiv:1911.01911*, 2019.
- [7] Markus Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohy Elbadrawy, Ahsan Lodhi, Haritha Katam, Tomas Hodan, Frank Michel, Carsten Rother, and Slobodan Ilic. Blenderproc: A procedural pipeline for photorealistic rendering. *arXiv preprint arXiv:1911.01911*, 2019.
- [8] Aneesh Deogan, Wout Bekx, Peter Teurlings, Koen De Vos, Mark Van Den Brand, and René Van De Molengraft. Synthetic dataset generation for autonomous mobile robots using 3d gaussian splatting for vision training. In *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*, pages 2802–2807. IEEE, 2025.

- [9] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1310–1319, 2017.
- [10] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1301–1310, 2017.
- [11] Mark Everingham, Luc Van Gool, Christopher K I Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.
- [12] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023.
- [13] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 270–279, 2017.
- [14] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 270–279, 2017.
- [15] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3827–3837, 2019.
- [16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.
- [17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.
- [18] 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.

- [19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Zhuang Mao, C. David Roland, Laura Gustafson, Andrea Vedaldi, Ori Shapira, Joseph Finkelstein, Trevor Darrell, Ross Girshick, and Piotr Dollár. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.
- [20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [22] Donghoon Lee, Sifei Liu, Jinwei Gu, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Context-aware synthesis and placement of object instances. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
- [23] Jiayi Li, Li Niu, and Liqing Zhang. Deep image matting: A comprehensive survey. *arXiv preprint arXiv:2304.04672*, 2023.
- [24] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *CVPR*, 2024.
- [25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.
- [26] Guilin Liu, Fitzsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.
- [27] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *International Conference on 3D Vision (3DV)*, 2024.
- [28] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.
- [29] Li Niu, Wenyang Cong, Liu Liu, Yan Hong, Bo Zhang, Jing Liang, and Liqing Zhang. Making images real again: A comprehensive survey on deep image composition. *arXiv preprint arXiv:2106.14490*, 2021.

- [30] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021.
- [31] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädl, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.
- [32] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [33] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.
- [34] 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. In *SIGGRAPH Asia*, 2024.
- [35] Stina Strandberg. Synthetic data generation for object detection with nerf and gaussian splatting: Enhancing synthetic drone detection with neural and point-based rendering techniques in unreal engine 5, 2025.
- [36] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 843–852, 2017.
- [37] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.
- [38] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.
- [39] Bram Vanhele, Brent Zoomers, Jeroen Put, Frank Van Reeth, and Nick Michiels. Cut-and-splat: Leveraging gaussian splatting for synthetic data generation. In *International Conference on Robotics, Computer Vision and Intelligent Systems*, pages 44–62. Springer, 2025.

- [40] 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 *CVPR*, 2024.
- [41] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019. Facebook AI Research.
- [42] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024.
- [43] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *International Conference on Learning Representations (ICLR)*, 2024.
- [44] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, 2024.
- [45] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *ECCV*, 2024.
- [46] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Free-form image inpainting with gated convolution. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4471–4480, 2019.
- [47] Farhad G. Zanjani, Davide Abati, Auke Wiggers, Dimitris Kalatzis, Jens Petersen, Hong Cai, and Amirhossein Habibian. Gaussian splatting is an effective data generator for 3d object detection, 2025.
- [48] Farhad G. Zanjani, Davide Abati, Auke Wiggers, Dimitris Kalatzis, Jens Petersen, Hong Cai, and Amirhossein Habibian. Gaussian splatting is an effective data generator for 3d object detection. *arXiv preprint arXiv:2501.00000*, 2025. To appear in CVPR 2025.
- [49] Tianle Zeng, Gerardo Loza Galindo, Junlei Hu, Pietro Valdastri, and Dominic Jones. Realistic surgical image dataset generation based on 3d gaussian splatting. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 510–519. Springer, 2024.
- [50] Michael Zollhöfer, Patrick Stotko, Alexander Görlitz, Christian Theobalt, Matthias Nießner, Marc Stamminger, Marcus Magnor, and Andreas Kolb. State

- of the art on 3d reconstruction with rgb-d cameras. *Computer Graphics Forum*, 37(2):625–652, 2018.
- [51] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001.