Department of Information Technology
Course of Studies: Information Technology

Technische Hochschule
mannheim

Compositional Gaussian Splatting for Multi-Object and Dynamic Scene Synthesis

Luca Hebeda

Master Thesis

for the acquisition of the academic degree

Master of Science

Submitted by	Luca Hebeda
on	31. Oktober 2025
Supervisor	Prof. Dr. Marcus Vetter
Second examiner	M. Sc. Yannick Bukschat

Written Declaration in Accordance with the Study and Examination Regulations

I hereby declare that I have written this thesis independently and have not used any sources or aids other than those indicated. All passages that are taken literally or in substance from other works are identified as such.

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

0.1 Introduction 1

0.1.1 Motivation 1

0.1.2 Structure of the Work 2

1 Fundamentals **3**

1.1 Neural Radiance Fields 3

1.2 Gaussian Splatting 4

1.2.1 Scene Representation with 3D Gaussians 4

1.2.2 Rendering with 3D Gaussian Splatting 5

1.2.3 Optimization 7

1.2.4 Differentiable Rasterizer 7

2 State of the Art **10**

2.1 Gaussian Splatting for Scene Representation 10

2.2 Dynamic Extensions of Gaussian Splatting 10

2.3 Synthetic Compositing and Dataset Pipelines 12

2.4 Gaussian Splatting for Synthetic Dataset Generation 12

2.5 Detailed Review of Prominent Dynamic GS Methods 13

2.5.1 Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting 13

2.6 Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis 15

3 Methodology **18**

3.1 Model Training 18

3.1.1 Data Preparation . . . 18
3.1.2 3D Gaussian Splatting . . . 19
3.1.3 4D Gaussian Splatting . . . 20
3.1.4 Dynamic 3D Gaussian Splatting . . . 21
3.2 Composition Pipeline . . . 22
3.3 Pipeline A: Reconstruction of 3D and Dynamic 3D Models . . . 23
3.3.1 Capture setup and calibration . . . 24
3.3.2 Data Preparation: Mask Generation and Prompt Propagation . 24
3.4 Pipeline B: Composition and Synthetic Dataset Generation . . . 25
3.4.1 Scene configuration and class indexing . . . 25
3.4.2 Camera definition and interpolation . . . 25
3.4.3 Placement duplication and motion handling . . . 26
3.4.4 Mask rendering and mask types . . . 26
3.4.5 6D pose estimation . . . 27
3.4.6 Keypoint detection and propagation . . . 28
3.4.7 Rendered outputs and metadata . . . 28

4 Results **29**
4.1 Model Evaluation . . . 29
4.1.1 Comparison of Results in NSTL with COLMAP and OpenCV . 29
4.2 4D Gaussian Splatting . . . 31
4.2.1 Yang et al. . . . 31
4.2.2 Volley Dataset . . . 39
4.2.3 Training with Fixed Background . . . 39
4.3 Composition Pipeline Evaluation . . . 41
4.4 Compositional Scene Generation . . . 41
4.5 Multi-view Occlusion Handling . . . 42
4.6 Scene Generalization . . . 43
4.7 Temporal Consistency in Dynamic Scenes . . . 43
4.8 Qualitative Pose Estimation . . . 44

4.9 Keypoint Propagation 45

5 Summary and Outlook **46**

5.1 Summary of Key Findings 46

5.2 Limitations 47

5.3 Future Developments and Research Perspectives 47

References **49**

0.1 Introduction

0.1.1 Motivation

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, 52]. At the core of these capabilities lies the data used to train and evaluate such models [38]. 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 aspects of research and development [11, 26].

In recent years, visual scene representation 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 observed from any viewpoint [30]. Methods such as Neural Radiance Fields (NeRFs) have shown that continuous three-dimensional models can be reconstructed from ordinary photographs, although these methods often involve 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 representing color, position, and opacity in space. This approach allows real-time rendering while maintaining a high level of detail and realism [12]. Because Gaussian primitives can be easily manipulated and recombined, they offer new possibilities for modular scene generation and synthetic data creation [41].

The generation of synthetic data is increasingly important for artificial intelligence [40, 7]. Real-world datasets are often limited by annotation errors, missing labels, and insufficient diversity [10]. Synthetic datasets can overcome these limitations by providing unlimited, precisely labeled, and perfectly synchronized examples [50]. They are particularly valuable in scenarios where ground-truth information such as depth, motion, or segmentation is difficult to obtain manually [13, 4].

For training robust AI models, both the quality and diversity of data are critical. A system that can automatically generate realistic, correctly annotated images from structured three-dimensional representations thus offers significant advantages for both academic research and industrial applications.

Despite this potential, most existing methods focus on reconstructing individual static

or dynamic scenes [28, 45], without providing tools for composing multiple objects or actors into coherent multi-view environments with synchronized annotations. Many practical applications—such as robotics, autonomous systems, and simulation—require dynamic scenes that include interacting elements while maintaining accurate geometry and temporal alignment [41].

0.1.2 Structure of the Work

This thesis addresses the aforementioned challenges by developing a modular pipeline based on static and dynamic Gaussian Splatting. The proposed system enables the reconstruction, composition, and rendering of complex multi-object scenes in calibrated camera setups. It produces synchronized outputs including color images, depth maps, and instance-level segmentation. Additionally, the framework supports annotations such as keypoints, occlusion scores, and estimated poses, providing a comprehensive 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 generated with minimal manual effort. In this way, the system bridges the gap between recent advances in neural scene representation and the practical need for scalable dataset generation.

The remainder of this thesis is organized as follows. Chapter 1 introduces the theoretical foundations of Gaussian Splatting. Chapter **??** reviews existing methods and positions the proposed system within the broader landscape of Gaussian-based representations and related work in dynamic scene modeling, including a detailed discussion of the two most prominent approaches. Chapter **??** describes the modifications and training improvements applied, followed by a step-by-step presentation of the completed pipeline. Chapter **??** presents reconstruction results for both static and dynamic Gaussian Splatting and provides a qualitative evaluation of the newly constructed pipeline. Finally, Chapter **??** summarizes the key findings, discusses limitations, and outlines potential directions for future research.

Chapter 1

Fundamentals

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

1.1 Neural Radiance Fields

Neural Radiance Fields (NeRF), introduced by Mildenhall et al. [30], 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 1.1.

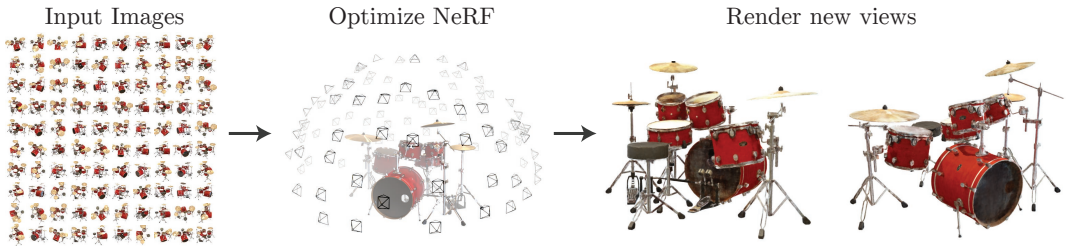


Figure 1.1: Core principle of NeRF: A model is optimized from input images and can subsequently generate novel views of the scene (after [30]).

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 1.2.

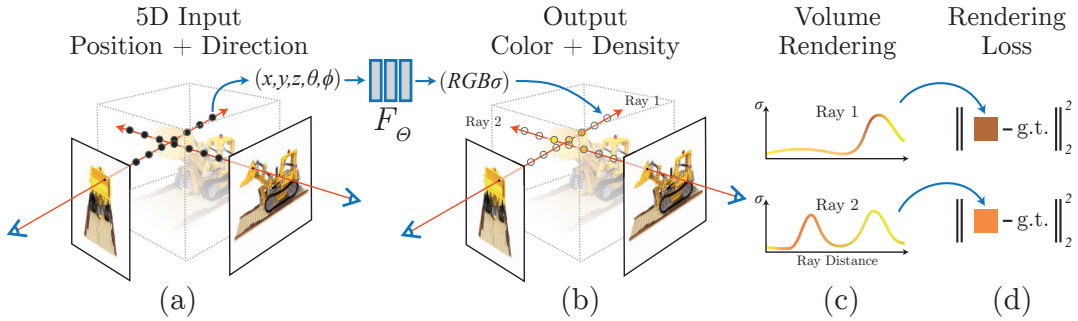


Figure 1.2: NeRF pipeline: The MLP maps 5D inputs to color and density, which are composed into an image via volumetric rendering (after [30]).

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.

1.2 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 1.3, 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.

1.2.1 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) [35], Gaussian Splatting is based on

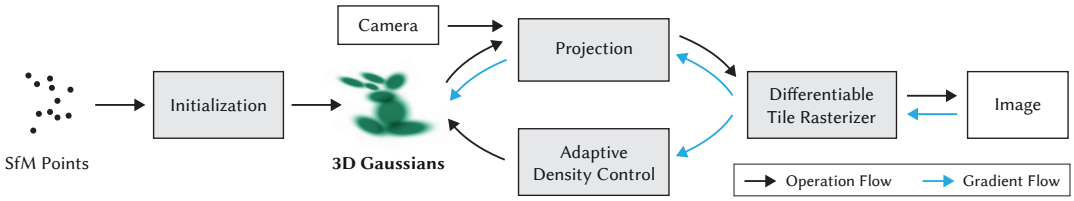


Figure 1.3: Optimization and rendering pipeline of Gaussian Splatting: The process begins with a sparse structure-from-motion (SfM) point cloud, which serves as the basis for creating an initial set of 3D Gaussian functions. These Gaussians are refined through iterative optimization, with their density 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 [34]. 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:

$$G(x) = e^{-\frac{1}{2}x^T\Sigma^{-1}x} \tag{1.1}$$

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.

1.2.2 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. [53] 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:

$$\Sigma' = JW\Sigma W^T J^T \tag{1.2}$$

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:

$$\Sigma = RSS^T R^T \tag{1.3}$$

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.

1.2.3 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:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}} \tag{1.4}$$

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 1.4, 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.

1.2.4 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].

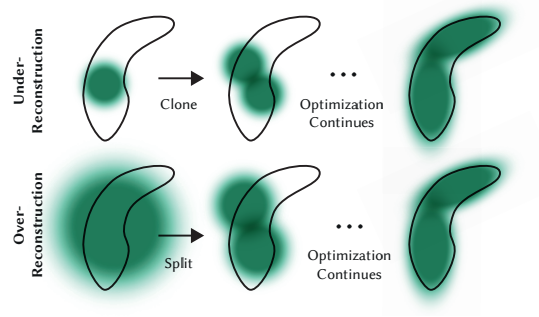


Figure 1.4: Adaptive density control: The top row shows under-reconstruction, where small geometries (black outlines) are supplemented by cloning a Gaussian. The bottom row shows over-reconstruction, where a large Gaussian is split into two smaller ones (after [18]).

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:

$$C = \sum_{i=1}^{N} T_i \alpha_i c_i, \quad \text{with} \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{1.5}$$

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:

$$I(u,v) = \sum_{i=1}^{N} p_i(u,v;\mu_i^{2d},\Sigma_i^{2d})\,\alpha_i\,c_i(d_i)\prod_{j=1}^{i-1}\left(1 - p_j(u,v;\mu_j^{2d},\Sigma_j^{2d})\,\alpha_j\right) \tag{1.6}$$

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 2

State of the Art

2.1 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. [30, 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 [46] 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.

2.2 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 for

dynamic extensions 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* [28] and *Dynamic Gaussian Marbles* [36] follow this paradigm, deriving dense motion fields from the per-Gaussian trajectories. More recent variants such as *Deformable 3DGS* [45] incorporate deformation fields inspired by D-NeRF [32], 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* [44], *SpaceTime Gaussians* [25], and *4D Gaussian Splatting* [42], 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.

The two dynamic paradigms, trajectory-based and 4D representations, offer complementary trade-offs. Trajectory-based methods, exemplified by D3DGS [28], prioritize modularity, interpretability, and flexibility, making them well-suited for compositional workflows and synthetic dataset generation. In contrast, 4DGS [44] provides globally consistent temporal reconstruction at higher computational cost but with reduced object-level control. 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. Methods using 4D Gaussian Splatting are also evaluated as a benchmark for comparison.

2.3 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, 39, 27, 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, 31].

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 [47, 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 spatial consistency and enabling deterministic reasoning about what is obscured in different 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, 43] 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].

2.4 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* [49] 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 [37], surgical data [51], and robotic perception scenes [8]. *Cut-and-Splat* [41] 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.

2.5 Detailed Review of Prominent Dynamic GS Methods

As previously mentioned, the two main paradigms of dynamic Gaussian splatting offer complementary benefits and trade-offs. The two most prominent methods, Dynamic 3D Gaussian Splatting (D3DGS) [28] and 4D Gaussian Splatting (4DGS) [44], are reviewed in detail below to illustrate their underlying principles and differences.

2.5.1 Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting

One of the pioneering extensions of 3D Gaussian Splatting to dynamic scenes is the approach by Yang et al. [44]. This method aims to model the underlying 4D space using 4D Gaussians, treating space and time as a unified construct. The complete rendering pipeline is illustrated in Figure 2.1 and explained step by step in the following sections.

Extension of the Rendering Equation

For dynamic scenes, indexing pixels solely by their spatial coordinates (u, v) is insufficient. Yang et al. extend Equation 1.6 by an additional timestamp t to capture the scene dynamics. The color $I(u, v, t)$ of a pixel at time t is computed by alpha-blending the visible 4D Gaussians:

$$I(u,v,t) = \sum_{i=1}^{N} p_i(u,v,t)\alpha_i c_i(d) \prod_{j=1}^{i-1}(1 - p_j(u,v,t)\alpha_j) \tag{2.1}$$

Here, $p_i(u, v, t)$ can be further factorized into a conditional probability $p_i(u, v|t)$ and a marginal probability $p_i(t)$:

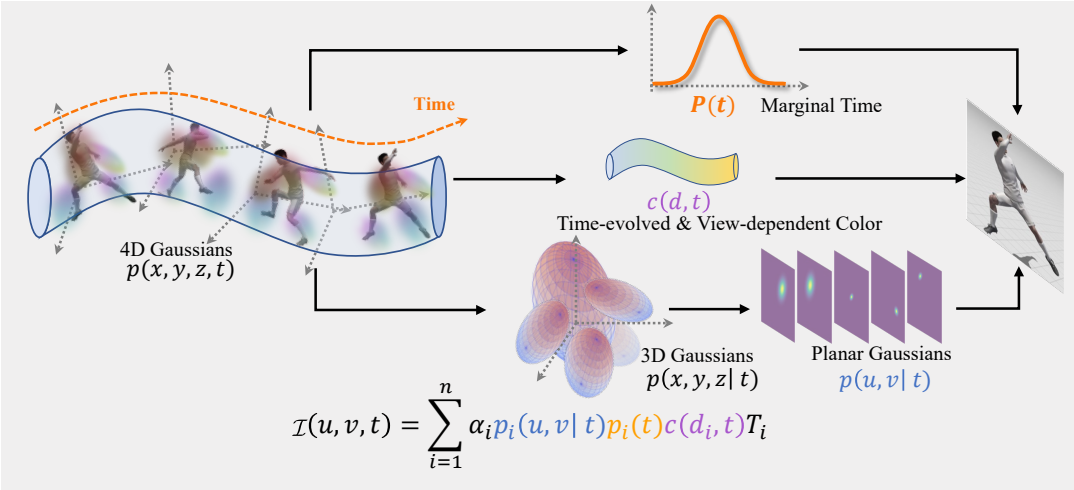


Figure 2.1: The rendering pipeline of the proposed 4DGS method. For a given time t and view I, each 4D Gaussian is decomposed into a conditional 3D Gaussian and a marginal 1D Gaussian. The conditional 3D Gaussian is then projected onto a 2D splat. Finally, the planar conditional Gaussian, the 1D marginal Gaussian, and the temporally varying, view-dependent color are combined to render the view I (figure adapted from [44]).

$$I(u,v,t) = \sum_{i=1}^{N} p_i(t) p_i(u,v|t) \alpha_i c_i(d) \prod_{j=1}^{i-1} (1 - p_j(t) p_j(u,v|t) \alpha_j) \tag{2.2}$$

Scene Representation with 4D Gaussians

A 4D Gaussian is described by a mean vector $\mu = (\mu_x, \mu_y, \mu_z, \mu_t)$. The definition of the covariance matrix Σ remains identical to the 3DGS formulation [18], factorized via a scaling matrix S and a rotation matrix R (cf. Equation 1.3).

The scaling matrix is extended with a temporal scaling factor along the diagonal, resulting in $S = \text{diag}(s_x, s_y, s_z, s_t)$. A rotation in 4D Euclidean space can be constructed from two isotropic rotations, both representable as quaternions. This allows anisotropic ellipsoids to rotate freely in space and time.

The left isotropic rotation $L(q_l)$ and right isotropic rotation $R(q_r)$ are defined as:

$$R = L(q_l)R(q_r) = \begin{pmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{pmatrix} \begin{pmatrix} p & -q & -r & -s \\ q & p & s & -r \\ r & -s & p & q \\ s & r & -q & p \end{pmatrix}. \tag{2.3}$$

Projection onto the Image Plane

The conditional 3D Gaussian at a specific time $p_i(x, y, z|t)$ can now be derived from the properties of the 4D Gaussian:

$$\mu_{xyz|t} = \mu_{1:3} + \Sigma_{1:3,4}\Sigma_{4,4}^{-1}(t - \mu_t) \tag{2.4}$$

$$\Sigma_{xyz|t} = \Sigma_{1:3,1:3} - \Sigma_{1:3,4}\Sigma_{4,4}^{-1}\Sigma_{4,1:3} \tag{2.5}$$

After extracting the conditional 3D Gaussians from the 4D representation, the 2D projection can be computed in the same way as for 3DGS.

The marginal distribution $p_i(t)$ indicates at which times a Gaussian contributes to the scene. It can be described by a 1D Gaussian:

$$p_i(t) = \mathcal{N}(t; \mu_4, \Sigma_{4,4}) \tag{2.6}$$

Spherical Harmonics for Color Representation

To model temporal evolution and view-dependent color, spherical harmonics are employed. These extend conventional spherical harmonics to account for the temporal dimension. The color is represented as a combination of 4D spherical harmonics:

$$Z_{nlm}(t, \theta, \phi) = \cos\left(\frac{2\pi n}{T}t\right) Y_l^m(\theta, \phi),$$

where Y_l^m are the 3D spherical harmonics, l the degree, m the order, and n the Fourier order. This enables modeling of temporally varying, view-dependent colors.

2.6 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. [28], 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.

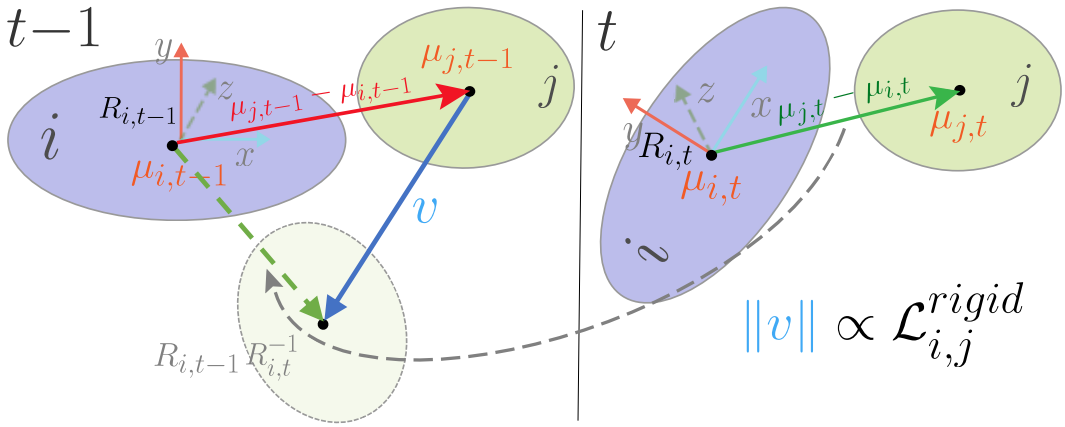


Figure 2.2: Schematic illustration of the local rotation similarity loss in D3DGS [28].

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. 2.2). The loss is defined as:

$$\mathcal{L}_{\text{rigid}} = \frac{1}{k|S|} \sum_{i \in S} \sum_{j \in \text{k-nn}} w_{i,j} \| (\mu_{j,t-1} - \mu_{i,t-1}) - R_{i,t-1} R_{i,t}^{-1} (\mu_{j,t} - \mu_{i,t}) \|^2, \tag{2.7}$$

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

$$w_{i,j} = \exp\left(-\lambda_w \|\mu_{j,0} - \mu_{i,0}\|^2\right). \tag{2.8}$$

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:

$$\mathcal{L}_{\text{rot}} = \frac{1}{k|S|} \sum_{i \in S} \sum_{j \in \text{k-nn}} w_{i,j} \| q_{j,t} q_{j,t-1}^{-1} - q_{i,t} q_{i,t-1}^{-1} \|^2. \tag{2.9}$$

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:

$$\mathcal{L}_{\text{iso}} = \frac{1}{k|S|} \sum_{i \in S} \sum_{j \in \text{k-nn}} w_{i,j} \left| \|\mu_{j,0} - \mu_{i,0}\| - \|\mu_{j,t} - \mu_{i,t}\| \right|. \tag{2.10}$$

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 1.4. 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 3

Methodology

This chapter describes the methodology used to train, evaluate, and integrate different Gaussian Splatting representations for both static and dynamic 3D scenes. The process follows a progressive structure that begins with independent model evaluation and culminates in the design of a composition pipeline for synthetic dataset generation.

In the first part, several Gaussian-based reconstruction methods are trained and compared, including static 3D Gaussian Splatting (3DGS), dynamic 3D Gaussian Splatting (D3DGS), and 4D Gaussian Splatting (4DGS). Each method is evaluated on standardized datasets to assess reconstruction quality and computational efficiency. The goal of this comparison is to identify the most suitable representation for large-scale compositional workflows.

Based on the evaluation results, the trajectory-based D3DGS approach is selected as the primary framework for dataset construction, offering a practical balance between reconstruction fidelity, modularity, and runtime performance. The second part of this chapter therefore focuses on the composition pipeline that builds upon these trained models to generate structured multi-object scenes with synchronized annotations. This pipeline forms the core contribution of the work and connects real-world capture with scalable synthetic data generation.

3.1 Model Training

3.1.1 Data Preparation

The data used for model training was captured using a custom multi-camera rig designed for synchronized high-resolution reconstruction. The rig consists of 56

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 distributes a global synchronization signal, ensuring microsecond-level temporal alignment 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, providing dense coverage from multiple viewpoints. The geometric center of this volume defines the global coordinate origin used for all reconstructions.

OpenCV Calibration

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.

3.1.2 3D Gaussian Splatting

The training procedure for 3D Gaussian Splatting (3DGS) follows the general methodology introduced by Kerbl et al. [18]. In the reference implementation, **COLMAP** is used to estimate camera poses and to generate an initial sparse point cloud for Gaussian initialization. While this approach achieves accurate geometry, it requires a full Structure-from-Motion (SfM) pipeline for each scene, which adds significant preprocessing time and introduces scene-dependent variability.

In this setup, the reconstruction process is based entirely on the deterministic calibration obtained from the multi-camera rig described in Section 3.1.1. Since all intrinsic and extrinsic parameters are known beforehand, the training can be initialized without any external SfM reconstruction or point cloud generation. This reduces computational overhead and ensures that all reconstructions share a consistent global coordinate frame, which simplifies later composition steps.

To validate that this calibration is sufficient for 3DGS reconstruction, two variants of

the same NSTL scene are trained and compared. The first uses camera parameters derived from COLMAP, while the second uses parameters obtained directly from OpenCV. This comparison serves to evaluate the effect of the calibration method on reconstruction quality and to confirm the feasibility of a fully deterministic, SfM-free workflow for static Gaussian-based scene reconstruction.

3.1.3 4D Gaussian Splatting

Building upon the 3D Gaussian Splatting (3DGS) framework, 4D Gaussian Splatting (4DGS) extends the representation to include temporal variation, enabling the reconstruction of dynamic scenes that evolve over time. This extension is particularly relevant for the Neural Space Time Lab (NSTL), which allows synchronized multi-view recordings with precisely aligned timestamps across all cameras. By incorporating time as an additional dimension, each Gaussian is defined not only by its spatial position and appearance attributes, but also by temporal parameters that capture motion and dynamic changes.

The preprocessing of the data remains largely identical to the static 3DGS workflow. All captured images are undistorted and rectified using the OpenCV calibration described in Section 3.1.1, and the corresponding camera parameters are structured in a JSON-based format. To account for the temporal component, each image is additionally associated with a precise timestamp, defining its position within the synchronized sequence. These timestamps are crucial for integrating the NSTL recordings into a coherent spatio-temporal reconstruction, ensuring consistent alignment between spatial and temporal information.

For dynamic scene modeling, the method proposed by Wang et al. [**?**] was selected due to its balance of reconstruction quality and rendering efficiency. Compared to alternative dynamic Gaussian methods using 4D representations, 4DGS offers the best reconstruction quality on standard benchmark datasets with fast rendering speed important for dataset generation. The following sections describe the adaptations and modifications made to the original 4DGS method to accommodate the specific requirements of the NSTL data and the overall pipeline. Initial experiments using both synthetic and NSTL datasets confirmed that the method reproduces the performance reported in the reference paper.

However, practical limitations quickly emerged when applying the method to real NSTL captures. The memory footprint grows rapidly with sequence length, as a large portion of the generated Gaussians are short-lived and contribute only marginally to the overall scene representation. Empirical analysis of trained NSTL models revealed

that approximately 80% of all Gaussians exhibit minimal temporal persistence, a finding consistent with the redundancy problem described by Yuan et al. [48]. This redundancy leads to inefficient GPU memory usage and restricts training to short temporal windows of only a few seconds.

To mitigate this issue, a background-aware training strategy was introduced, leveraging the fixed-camera setup of the NSTL rig. Since each camera captures an empty reference frame prior to the motion sequence, these static background images are directly embedded into the rendering process. Pixels that are fully explained by static geometry are filled from the reference image, while Gaussians are only optimized for regions of change, such as moving subjects, shadows, or reflections. This approach eliminates the number of Gaussians required for static areas almost completely and allows the available memory to be focused on dynamic content.

While this modification improved efficiency and reconstruction quality, the overall complexity and high memory requirements of 4DGS still limited its scalability. As a result, the focus of this work shifted toward a simplified, modular approach based on Dynamic 3D Gaussian Splatting (D3DGS), described in the following section. This method retains the ability to represent motion through temporally segmented static reconstructions, while providing a significantly lighter and more flexible pipeline suitable for multi-object composition.

3.1.4 Dynamic 3D Gaussian Splatting

While 4DGS provides a unified spatio-temporal representation, its high memory consumption and complexity limit its practicality for large-scale dataset generation. As an alternative, Dynamic 3D Gaussian Splatting (D3DGS) [28] offers a trajectory-based approach that models dynamic scenes as sequences of static 3DGS reconstructions. The methods uses a fixed number of Gaussians that are tracked over time, allowing each Gaussian to move and change appearance according to learned trajectories. This allows for efficient representation of motion while maintaining the modularity and interpretability of static Gaussian models, which is particularly advantageous for compositional workflows.

Dataset Preparation and Mask Generation

Dynamic 3D Gaussian Splatting requires a very similar structure to 4DGS for dataloading. However it requires per-frame object masks to disentangle dynamic objects from the static background and to maintain temporal consistency during training

[28]. While the reference implementation primarily relies on binary foreground/background masks, this pipeline extends that step by integrating **SAM2** [33], 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.

Training Adjustments

Training largely follows the original D3DGS methodology, with minor adaptations for the NSTL environment. Parameters controlling brightness and contrast of rendered images were fixed to improve color consistency, and the floor-loss term was disabled as it induced artifacts in early experiments. All other hyperparameters and loss functions were retained as in the reference implementation.

The use of SAM2 masks ensures that Gaussians are primarily optimized for dynamic objects, while the static background is implicitly handled by the consistent camera setup and training procedure. This reduces memory usage and allows the model to focus its capacity on regions of interest, improving reconstruction quality for moving elements in the scene.

3.2 Composition Pipeline

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

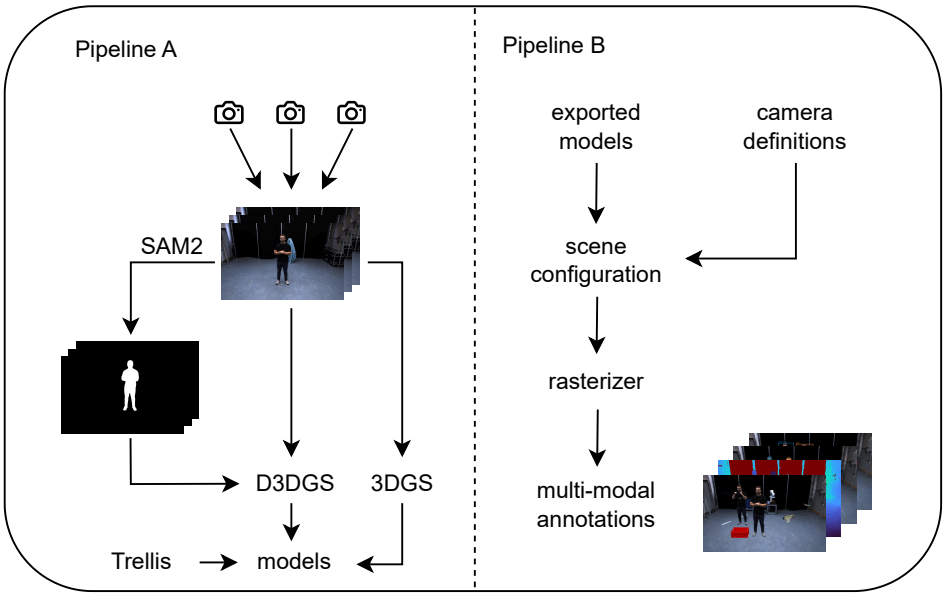


Figure 3.1: **Overview of the proposed composition pipeline. Pipeline A** converts multi-view captures into per-object 3DGS and D3DGS models via calibration, mask generation, and model training. **Pipeline B** composes exported models into multi-object scenes, producing synchronized RGB, depth, segmentation, and occlusion annotations.

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. An overview of the entire process, including both reconstruction and composition stages, is illustrated in Figure 3.1.

3.3 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.

3.3.1 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.

3.3.2 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 [28]. While the reference implementation primarily relies on binary foreground/background masks, this pipeline extends that step by integrating **SAM2** [33], 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.

3.4 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.

3.4.1 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.

3.4.2 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 subframe 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.

3.4.3 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.

3.4.4 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.

3.4.5 6D pose estimation

For each object instance, a rigid transformation $T_{\text{obj}\to\text{cam}} \in \mathbb{R}^{4\times4}$ is estimated to map canonical object coordinates into the current camera frame. The translation component is derived from the displacement of the object centroid, computed either over all Gaussian means or a designated subset of anchor points:

$$t = \bar{x}_t - \bar{x}_0, \tag{3.1}$$

$$\bar{x}_t = \frac{1}{|\mathcal{A}|} \sum_{i\in\mathcal{A}} x_i^{(t)}. \tag{3.2}$$

Rotational change is obtained from the per-anchor quaternions $q_i^{(t)} = (w, x, y, z)$ using Markley's quaternion averaging method [29]. A mean quaternion is computed for both the reference and current frame:

$$q_{\text{mean}}^{(t)} = \arg\max_{\|q\|=1} q^\top \left(\sum_i q_i^{(t)} q_i^{(t)\top} \right) q, \tag{3.3}$$

and the relative rotation is given by

$$q_\Delta = q_{\text{mean}}^{(t)} \otimes \overline{q_{\text{mean}}^{(0)}}, \tag{3.4}$$

$$R_\Delta = \text{quat2mat}(q_\Delta). \tag{3.5}$$

The resulting rotation is combined with an optional initial alignment R_0, yielding

$$R_{\text{obj}} = R_\Delta R_0. \tag{3.6}$$

Finally, the object-to-camera transform is assembled as

$$T_{\text{obj}\to\text{cam}} = \left(T_{\text{w}\to\text{c}}^{-1} \right) \begin{bmatrix} R_{\text{obj}} & \bar{x}_t \\ 0 & 1 \end{bmatrix}. \tag{3.7}$$

This transformation represents a temporally consistent 6D pose estimate that captures the object's centroid and orientation changes over time. Although these poses are not absolute ground truth, they provide reliable internal motion estimates that can be leveraged for downstream tasks such as motion tracking, pose refinement, or activity recognition.

3.4.6 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 [43]. 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.

3.4.7 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 4

Results

4.1 Model Evaluation

4.1.1 Comparison of Results in NSTL with COLMAP and OpenCV

To validate the 3D Gaussian Splatting method in the Neural Space Time Lab (NSTL), a new dataset was captured and prepared using two different calibration approaches: COLMAP and OpenCV. The scene was trained, rendered, and evaluated using SSIM, PSNR, and LPIPS metrics to assess the impact of the calibration method on reconstruction quality, as described in the methodology section (Chapter **??**).

The results are summarized in Table 4.1. Both calibration methods achieve high reconstruction quality, with COLMAP showing slightly better metric values. The differences can be attributed to the optimized estimation of camera poses through the structure-from-motion (SfM) procedure.

Table 4.1: Comparison of metrics for the scene from the NSTL dataset.

	SSIM ↑	PSNR (dB) ↑	LPIPS ↓
COLMAP	0.931	35.58	0.18
OpenCV	0.927	34.56	0.252

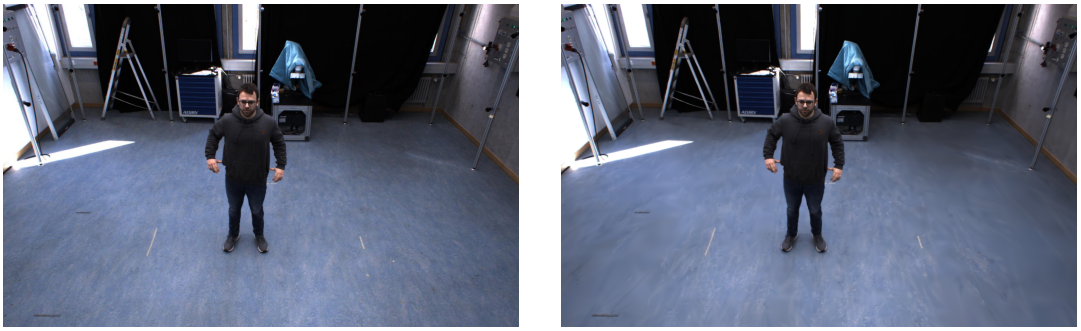
Figure 4.1 shows a visual comparison for the COLMAP-based reconstruction. The rendered image closely resembles the ground truth, with clear textures and edges in the central region, while slight blurring is visible near the floor and image borders.

Figure 4.2 presents the OpenCV-based reconstruction. Texture fidelity in the central region remains high, but quality decreases more noticeably toward the edges,

particularly at the floor and in background elements such as a ladder.

(a) Ground Truth (b) Rendered Image

Figure 4.1: Comparison between ground truth and rendered image using COLMAP camera poses: The reconstruction shows clear textures in the central region, with minor blurring near the floor and image borders.



(a) Ground Truth (b) Rendered Image

Figure 4.2: Comparison between ground truth and rendered image using OpenCV calibration: The reconstruction maintains good texture fidelity in the central region, with stronger blurring near the floor and background.

Despite the slightly better metrics of COLMAP, OpenCV is preferred for NSTL applications, as explained in the methodology section (Chapter **??**). Deterministic calibration with OpenCV provides a coordinate system in real-world units, ensuring precise and reproducible spatial positioning. This is crucial for integration with other NSTL systems and for future extensions to dynamic scenes using 4D Gaussian Splatting. The similar reconstruction quality in the central region demonstrates that OpenCV, despite slightly lower metrics, is a robust alternative due to its consistency and lower dependence on scene-specific optimizations. These results provide a solid foundation for the investigation of dynamic scenes in the following sections.

4.2 4D Gaussian Splatting

4.2.1 Yang et al.

Method Validation with D-NeRF and DyNeRF Datasets

To validate the method proposed by Wang et al. [44], the D-NeRF and DyNeRF datasets were used. Specifically, the "Mutant" scene from the D-NeRF dataset [32] and the "cut roasted beef" scene from the DyNeRF dataset [24] were employed. Unlike the reference paper, where metrics for D-NeRF were averaged across all scenes, a scene-specific analysis was performed here. For DyNeRF, the reference values directly correspond to the "cut roasted beef" scene under investigation. The results are presented in Tables 4.2 and 4.3. They show a close agreement with the reference values; deviations for D-NeRF are below 0.5 dB (PSNR) and 0.03 (SSIM), which can be attributed to the use of a single scene and slightly different training conditions. For DyNeRF, the implementation matches the reference values almost exactly.

Figure **??** illustrates the comparison between ground-truth and rendered views for the "Mutant" scene. The reconstructions achieve high visual consistency, with only fine details, such as the scar on the torso, appearing slightly less sharp.

Figure 4.5 highlights the dependency of reconstruction quality on object motion in the "cut roasted beef" scene. The top row shows a frame with high object motion (knife, tongs), where blurring and partial reconstruction errors occur. In contrast, the bottom row depicts a frame with minimal motion, where nearly all regions are reconstructed accurately.

Table 4.2: Comparison of metrics for the "Mutant" scene from the D-NeRF dataset.

	SSIM ↑	PSNR (dB) ↑	LPIPS ↓
Reference (averaged) [44]	0.98	34.09	0.02
Validation	0.974	34.135	0.023

Table 4.3: Comparison of metrics for the "cut roasted beef" scene from the DyNeRF dataset.

	SSIM ↑	PSNR (dB) ↑	LPIPS ↓
Reference [44]	0.98	33.85	n.a.
Validation	0.954	33.163	0.132

Figure 4.3: Example reconstruction from a test camera: While central image elements are consistently reconstructed, the ceiling exhibits strong errors due to insufficient training data coverage.

Figure 4.4: Example reconstruction from a test camera: During high-motion frames (top), some blurring and incomplete reconstructions occur; in low-motion frames (bottom), the reconstruction quality is high across the scene.

Results with the NSTL Dataset

Experiments with the newly captured NSTL dataset highlight the influence of the number of Gaussians on reconstruction quality. As shown in Table 4.5, increasing the number of Gaussians initially leads to significant improvements, particularly from 2 million to 3.5 million Gaussians. The gains between 3.5 and 5 million Gaussians are less pronounced.

Figure **??** shows representative frames at the start and end of the sequence. With 2 million Gaussians, reconstruction deficiencies are evident, especially in the face and hands, which appear blurred and incomplete. Increasing to 3.5 million Gaussians improves facial reconstruction and clarifies hand contours, though some blurring remains. The 5 million Gaussian model shows no substantial improvement over the 3.5 million model, although fingers are partially reconstructed. Both metrics and visual inspection confirm that increasing the Gaussian count enhances reconstruction quality, particularly in fine details such as hands and fingers.

Due to memory constraints, the full sequence of 200 frames could not be trained with higher Gaussian counts. By reducing the sequence to 50 frames (1.6 seconds), finer modeling was possible. Evaluation of this shortened sequence yielded an SSIM of 0.907, a PSNR of 31.13 dB, and an LPIPS of 0.217. Visual comparison further supports this quality improvement. Figure **??** shows four representative rendered frames; the face exhibits high detail, and the hands are mostly reconstructed consistently. Even during overhead movements, the hands remain clear with only minor reconstruction artifacts.

Thus, the shortened sequence significantly outperforms the full sequence (see Table 4.5). The improved quality can be explained by the reduced temporal complexity, allowing more Gaussians to model fine motions, particularly in the hands.

Table 4.4: Comparison of metrics for the NSTL "4DGS dataset" with varying numbers of Gaussians and sequence lengths.

Sequence length	Gaussians	SSIM ↑	PSNR (dB) ↑	LPIPS ↓
200 Frames	2M	0.887	28.67	0.253
	3.5M	0.898	29.78	0.231
	5M	0.901	30.23	0.228
50 Frames	5M	0.907	31.13	0.217

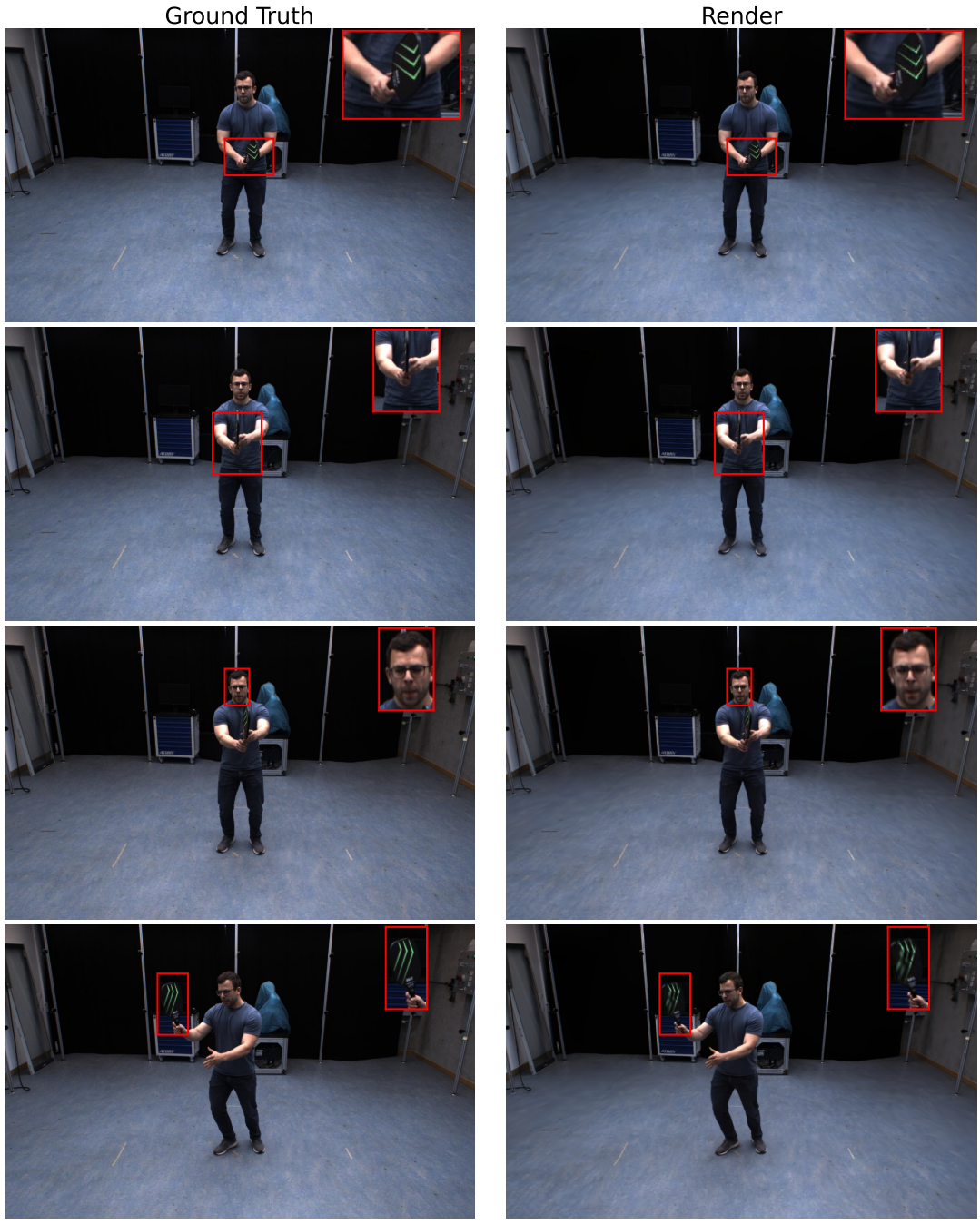


Figure 4.5: Comparison of ground truth (left) and rendered images (right) for the 4D Tennis dataset with 50 images.

Results of the Pruning Approach

The experiments show that the proposed pruning approach can significantly reduce the model's memory footprint without substantially compromising image quality. Compared to the previous best model with 5M Gaussians (9.6 GB memory), the number of Gaussians could be reduced by up to 70%. At the same time, evaluation metrics improved slightly (see Table 4.5).

The model with pruning applied after densification (1.5M Gaussians) already shows a notable improvement over the baseline. Further integration of pruning during densification (1.8M Gaussians) led to additional metric gains. This indicates that short-lived and redundant Gaussians are indeed removed early and replaced with more relevant ones.

Visual comparisons (Fig. 4.7) present a more nuanced picture. While global metrics suggest improvements, Gaussians are also removed in critical areas such as the fingers. Pruning after densification shows some improvement in the fingers, whereas pruning during densification results in reduced quality in fingertip modeling. This highlights that metrics capture visual quality only partially, as they emphasize improvements in static regions over losses in dynamic regions, which occupy less spatial area.

Analysis of Active Gaussians Notably, the number of active Gaussians per timestep increases significantly due to pruning (from 20% to over 35%), as shown in Fig. 4.6 (a). Rendering times are also reduced because fewer Gaussians need to be projected onto the 2D image plane. Covariance values, however, change only slightly, indicating that the temporal persistence of individual Gaussians is minimally affected by pruning (see Fig. 4.6 (b)). This may also be related to the scene complexity. Unlike DyNeRF scenes, NSTL recordings contain large-scale movements. To capture detailed hand motions, Gaussians with low covariance are required to model fine-grained movements.

These results illustrate a fundamental trade-off of the pruning approach: While reducing the number of Gaussians and the associated memory savings leads to significantly improved quantitative metrics, qualitative results show deterioration in areas with complex motion, such as the hands. Metrics primarily benefit from sharper reconstructions of static backgrounds, whereas dynamic details lose precision due to the removal of relevant Gaussians. This suggests that pruning strategies, although promising for efficiency, should be complemented by adaptive methods that evaluate each Gaussian not only for persistence but also in the context of dynamic motion. Such an approach could achieve a better balance between memory usage, numerical

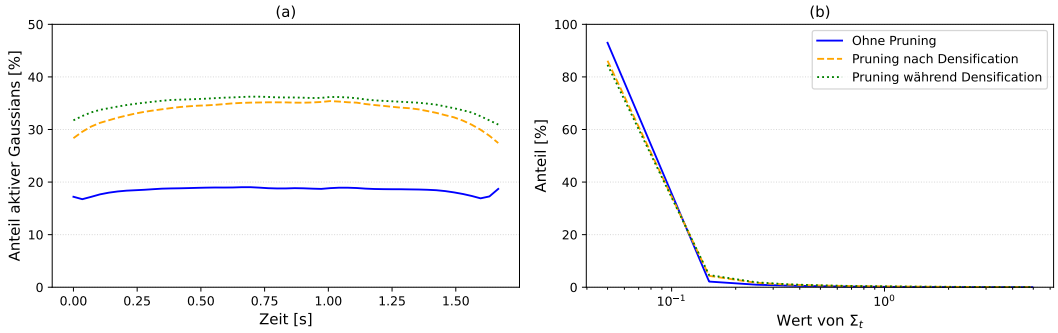


Figure 4.6: Comparison of different pruning strategies: (a) proportion of active Gaussians over time, (b) distribution of Σ_t.

metrics, and perceived visual quality in the long term.

Table 4.5: Comparison of metrics for the NSTL "4DGS dataset" with varying Gaussian counts and memory usage.

Number of Gaussians	Memory (Gb)	SSIM ↑	PSNR (dB) ↑	LPIPS ↓
5M	9.6	0.907	31.13	0.217
1.5M	2.9	0.926	32.83	0.164
1.8M	3.5	0.940	33.12	0.133

(a) Ground Truth 1

(b) Render 1 (Pruning during Densification)

(c) Ground Truth 2

(d) Render 2 (Pruning during Densification)

Figure 4.7: Comparison of ground truth (left) and rendered images (right) for camera C-1-M. While background details improve with pruning, foreground regions (hands) exhibit qualitative weaknesses.

4.2.2 Volley Dataset

Experiments show that capturing a cleaned-up scene already results in a significant improvement (see Table 4.6). Without explicit pruning, the total number of Gaussians reduced to 3.7M (7.1 GB), while quality metrics increased significantly compared to the original 5M Gaussian scene.

Table 4.6: Comparison of metrics between the original and cleaned scene.

Scene	Number of Gaussians	SSIM ↑	PSNR (dB) ↑	LPIPS ↓
Original	5M	0.907	31.13	0.217
Cleaned	3.7M	0.937	35.8	0.159

Qualitative results (Fig. **??**) confirm the numerical improvements. The scene is reconstructed more sharply and consistently, especially in static regions previously burdened by unnecessary background Gaussians. Global player movements are modeled accurately in most frames.

However, a closer analysis reveals a more nuanced picture. While early frames are nearly perfectly aligned with ground truth, the swing motion shows initial limitations. The light green stripes on the racket appear blurred and lose detail, obscuring sharp edges. This indicates that modeling fast movements and fine structures remains challenging. On the positive side, the hand without the racket is reconstructed much more consistently and with more detail compared to previous experiments. This demonstrates a key advantage of the cleaned scene: model capacity is focused on relevant objects and motions. Moreover, the number of Gaussians was not limited by memory during training, as the 5M maximum capacity was not reached. Observed limitations are therefore due to the challenge of representing fast movements with high fidelity rather than model size constraints.

Overall, these experiments show that careful selection of a cleaned scene can achieve substantial improvements without complex pruning strategies. Previous results indicated that redundant Gaussians negatively affect memory and quality; the new scene confirms this observation under realistic conditions.

4.2.3 Training with Fixed Background

Integrating a fixed background into the training process leverages the specific structure of the NSTL setup. Before the start of a sequence, a reference image of the empty scene can be captured from each camera, so that static scene elements are represented

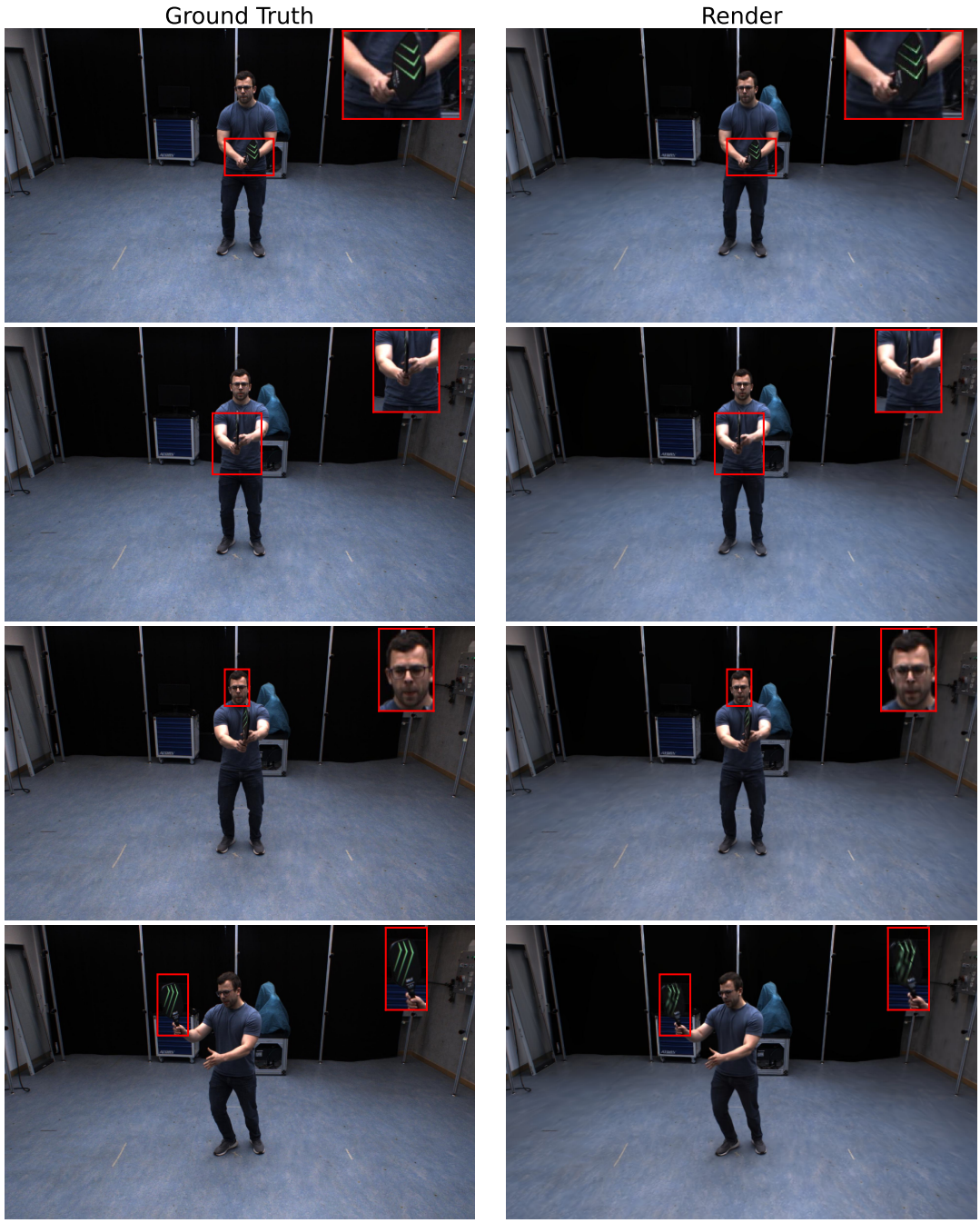


Figure 4.8: placeholder

directly via these background images rather than Gaussians. This allows the model capacity to focus on dynamic objects.

Results from training with 50 images show that this approach yields high-quality reconstructions. Both the racket surface and the player's hands are modeled with sharp edges and consistent geometry throughout the sequence. In contrast, training with 300 images leads to increased reconstruction errors, particularly blurring of the hands, face, and fine structures on the racket.

Numerical evaluation with PSNR, SSIM, or LPIPS is not meaningful in this scenario. Since the background is explicitly embedded via reference images and not modeled with Gaussians, reconstructed background areas appear largely black in the renderings. This distorts metrics, which would not reflect the quality of dynamic reconstruction. Instead, renderings reveal isolated bright Gaussians on the floor. These artifacts are due to inconsistencies between static and dynamic captures, especially shadows present only during motion that are absent in the reference image.

The degradation with 300 images can be attributed to increased optimization complexity. With six times more training images, both the number of Gaussians to be placed and the diversity of scene content increases, leading to reduced model capacity for focusing on fine details in individual regions.

4.3 Composition Pipeline Evaluation

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.

4.4 Compositional Scene Generation

Figure 4.9 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

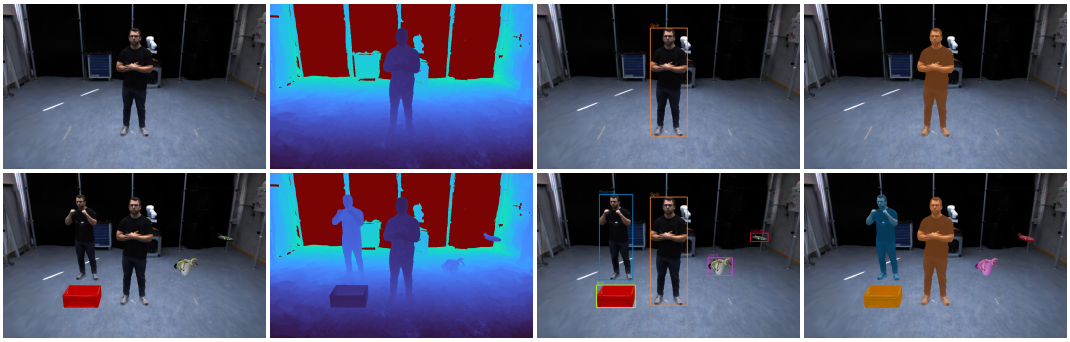


Figure 4.9: **Compositional scene generation.** Example of multi-object scenes composed from independently reconstructed Gaussian Splatting models. Each row shows one scene configuration with RGB, depth, bounding boxes, and segmentation overlays. All semantic annotations are generated automatically and remain spatially consistent across views.

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.

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.

4.5 Multi-view Occlusion Handling

To assess the robustness of the compositing mechanism under occlusions, Figure 4.10 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.

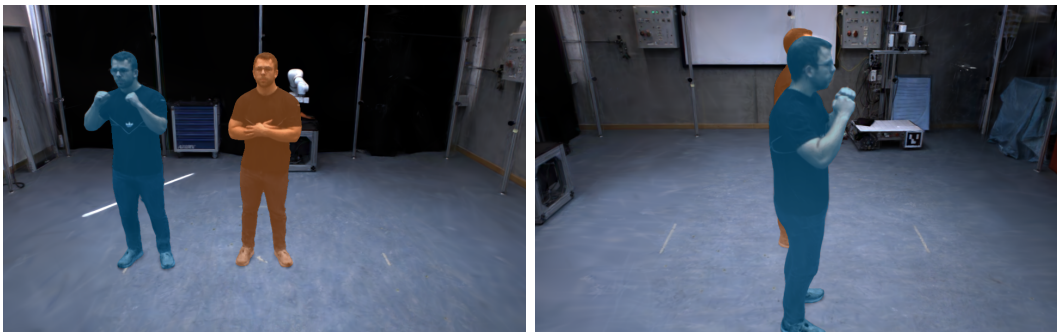


Figure 4.10: **Occlusion handling across views.** Segmentation overlays from front and side viewpoints of two interacting subjects. Even under strong mutual occlusion, instance masks remain consistent and well aligned with visible contours.

4.6 Scene Generalization

Figure 4.11 demonstrates the adaptability of the proposed system by recontextualizing dynamic subjects within a novel virtual environment reconstructed from real imagery using COLMAP [34]. 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.

4.7 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

Figure 4.11: **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 4.12: **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.

segmentation. Figure 4.12 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.

4.8 Qualitative Pose Estimation

Figure 4.13 (top row) shows qualitative 6D pose estimates for a dynamic human sequence across four time steps. Each frame visualizes the canonical object coordinate

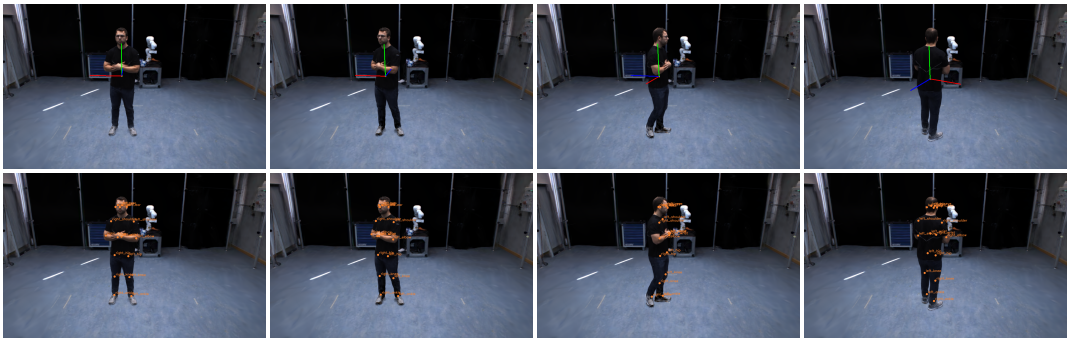


Figure 4.13: **Qualitative analysis of pose and keypoint propagation.** Four frames of a dynamic subject showing estimated object poses (top) and propagated keypoints (bottom). The coordinate systems evolve consistently over time, while keypoint trajectories remain coherent despite local misalignments.

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.

4.9 Keypoint Propagation

The lower row of Figure 4.13 visualizes the temporal keypoint propagation results obtained with a Detectron2-based detector [43]. 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 5

Summary and Outlook

5.1 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.

5.2 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.

5.3 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.

References

[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 Beks, 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 Rolland, 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] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *CVPR*, 2022.

[25] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *CVPR*, 2024.

[26] 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.

[27] Guilin Liu, Fitsum 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.

[28] 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.

[29] F. Landis Markley, Yang Cheng, John L. Crassidis, and Yaakov Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193–1197, 2007.

[30] 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.

[31] Li Niu, Wenyan 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.

[32] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021.

[33] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[34] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. Facebook AI Research.

[44] 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.

[45] 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.

[46] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *ECCV*, 2024.

[47] 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.

[48] Yuheng Yuan, Qiuhong Shen, Xingyi Yang, and Xinchao Wang. 1000+ fps 4d gaussian splatting for dynamic scene rendering, 2025.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001.