

4D RECONSTRUCTION IN SPARSE CAMERA SETTINGS FOR VR

Ezgi Aysel Batı and Yiran Chen

DEPARTMENT OF DESIGN SCIENCES
FACULTY OF ENGINEERING LTH | LUND UNIVERSITY
2026

MASTER THESIS



4D RECONSTRUCTION IN SPARSE CAMERA SETTINGS FOR VR

Ezgi Aysel Batı and Yiran Chen



LUND
UNIVERSITY

4D RECONSTRUCTION IN SPARSE CAMERA SETTINGS FOR VR

Copyright © 2026 Ezgi Aysel Batı and Yiran Chen

Published by

Department of Design Sciences
Faculty of Engineering LTH, Lund University
Box 118
SE-221 00 LUND
Sweden

Subject: Virtual and Augmented Reality (MAMM15)

Supervisor(s): Günter Alce, gunter.alce@design.lth.se

Opponent(s): Ximeng Sun, Xiaoyu Sheng

Examiner: Joakim Eriksson, joakim.eriksson@design.lth.se

Abstract

This thesis explores 4D reconstruction in sparse fixed-camera settings for virtual reality visualization. Gaussian Splatting provides an efficient representation for static and dynamic scenes, but most existing pipelines rely on dense multi-view capture or sufficient camera motion. This project focuses on a constrained setting with only four fixed cameras observing a dynamic human object, where limited view coverage makes stable geometry and novel-view rendering difficult. We first evaluate vanilla 3DGS and 4DGS pipelines on public datasets and self-recorded data to identify their limitations under sparse fixed views. The baseline experiments show that the original pipelines struggle to obtain reliable camera poses and stable geometry. Geometry-based improvements are then explored, including depth-prior supervision and cross-view geometric consistency. These constraints provide additional supervision, but they do not recover unobserved regions or produce a stable full 360° reconstruction. Based on these observations, the final pipeline adopts a generative completion approach. The four cameras are calibrated, foreground masks are generated to isolate the target human object, and Diffuman4D is used to synthesize novel-view human videos. The generated multi-view data is adapted to vanilla 4DGS and trained with modified data loading to handle the larger dataset. The dynamic Gaussian representation is then exported as per-frame splat files and visualized in a Unity-based VR environment. The results show that generative completion is more suitable than geometry-only stabilization for this setting. The final reconstruction enables recognizable 360° dynamic observation, but remains limited by temporal inconsistencies in generated images and residual foreground noise.

Keywords: 4DGS, Gaussian Splatting, Novel View Synthesis, Virtual Reality

Table of contents

List of acronyms and abbreviations	7
1. Introduction	8
1.1 Background	8
1.2 Motivation	9
1.3 Aim and Scope	10
1.4 Alignment with Sustainable Development	10
1.5 Ethical Considerations	11
1.6 Related Work	12
1.6.1 3DGS under Sparse View Camera Setups	12
1.6.2 Gaussian Segmentation	14
2. Theory	15
2.1 Structure-from-Motion (SfM)	15
2.1.1 Structure from Motion	15
2.1.2 COLMAP	16
2.1.3 Camera Parameters	16
2.2 Gaussian Splatting	17
2.2.1 Neural Radiance Fields	17
2.2.2 3D Gaussian Splatting	17
2.2.3 4D Gaussian Splatting	18
2.3 Virtual Reality	19
3. Project Overview	20
3.1 Technical Setup and Data Specification	20
3.2 Evaluation of Baseline Methodology	21
3.2.1 3DGS with different Parameters	21
3.2.2 Camera and Camera Position	25
3.2.3 4D Gaussian Splatting	28
3.3 Calibration	29
3.3.1 ChArUco Calibration	29
3.3.2 Multi-Camera Calibration	30
4. Geometry Based Methods and Evaluation	32

5. Generative Solutions and Evaluation	34
5.1 Overview	34
5.2 Method Selection	34
5.2.1 Single-View Video Generation (Efficient4D)	35
5.2.2 Static Novel View Synthesis	36
5.2.3 Sparse-View Human Video Diffusion (Diffuman4D)	37
5.3 Takeaway	38
6. Segmentation	40
6.1 Segment Any 4D Gaussians (SA4D)	40
6.2 2D Segmentation and Mask Generation	40
7. VR Visualisation	42
7.1 Existing Methods	42
7.2 Our Solution	43
8. Final Pipeline	45
9. Discussion	48
9.1 Research Questions Revisited	48
9.2 Limitations and Implications for VR	50
9.3 Future Work	51
9.3.1 Confidence-Aware Training for Generated Multi-View Data	51
9.3.2 Cleaner Foreground Gaussian Reconstruction and Segmentation	52
10 Conclusion	54
References	55

List of acronyms and abbreviations

3DGS	3D Gaussian Splatting
4DGS	4D Gaussian Splatting
AI	Artificial Intelligence
D - Nerf	Dynamic Neural Radiance Fields
FPS	Frames per Second
GPU	Graphics Processing Unit / Graphics Card
HMD	Head Mounted Display / Headset
NeRF	Neural Radiance Fields
NVS	Novel View Synthesis
SAM	Segment Anything Model
SEVA	Stable Virtual Camera
SfM	Structure From Motion
VR	Virtual Reality
VRAM	Video Random Access Memory

1 Introduction

This chapter introduces the field of 3D reconstruction and state-of-the-art methods (1.1), motivation and contributions of this thesis (1.2) and the research questions (1.3). Additionally it describes alignment with sustainable development (1.4), ethical considerations (1.5) and related work (1.6).

1.1 Background

Digital reconstruction of objects and scenes in 3D has been a field of interest for many years, with an increase in research after availability of easy to use recording equipment and advancements in 3D visualization such as better graphics and virtual reality (VR) (See 2.4). The traditional and intuitive method of 3D reconstruction in computer vision and computer graphics consisted of various tools and sensors that scan to estimate the three dimensional structure of the object itself, or patching together many high-resolution images with photogrammetry. These methods were limited and not scalable in their reconstruction capabilities by the amount and detail recorded. However, with recent advancements in machine learning and neural networks it became possible to analyse the depth from 2D monocular images and videos to generate high quality renderings. With these techniques, unseen angles beyond the two-dimensional images can also be estimated and generated, referred to as Novel View Synthesis (NVS) to provide a complete three-dimensional model. In the recent years there have been two big breakthroughs that accelerated research in 3D reconstruction and novel view synthesis: Neural Radiance Fields (NeRFs) [23] and 3D Gaussian Splatting (3DGS) [10].

Neural Radiance Fields [23] are a volumetric rendering method that uses a set of images from known camera poses to render advanced graphics reconstructions of scenes including novel views. This method makes use of a non-convolutional deep neural network to obtain high detailed photorealistic results. It quickly become an industry standard for both research and application tools, leading to many variations with specialized focuses on different aspects of the NVS problem. However, this high resolution reconstruction method comes with the caveat of long and complex training.

The next breakthrough 3D Gaussian Splatting solved the primary trade-off of the NeRF method, achieving similar and at times better quality renderings with a faster, more efficient training and real-time rendering [10]. Instead of an entire volumetric reconstruction, gaussian splatting makes use of rasterization in a point-cloud structure, having ellipsoid gaussians holding information of position, rotation, size and colour. The splats together form the 3D reconstruction, allowing novel view from different angles.

Many research papers based on 3DGS have been published, and more are released each day. It is a very relevant topic in the current digital reconstruction and NVS field. One such interesting method is 4D Gaussian Splatting [34], enabling efficient reconstruction of dynamic scenes using a deformation function applied over the initial splat result to modify the gaussians per timestamp.

The main contributions of this thesis are, an evaluation of vanilla 3DGS, 4DGS and various 4DGS based methods under a four-camera sparse fixed setup, as well as a customized 4DGS viewer in VR.

1.2 Motivation

Gaussian Splatting has become a practical way to represent and render 3D scenes in recent years, and its extension to 4D Gaussian Splatting [34] allows dynamic scenes to be modelled over time. Most existing methods, however, assume dense multi-view capture, where many cameras observe the scene from different angles. With a multi-view arrangement purposefully set up in a way that achieves close to full coverage around an object, it is possible to get a very stable and detailed reconstruction of recorded object. This is, however, a very intentionally planned, and often expensive or time consuming recording setting not meant for spontaneous occurrences. In many real-life scenarios, such as surveillance or monitoring systems, only a small number of fixed cameras are available. These cameras provide sparse viewpoints and limited geometric constraints. Under such conditions, Gaussian Splatting often produces unstable geometry, floating artifacts, and poor novel-view rendering.

For dynamic scenes, these problems become more severe, as insufficient constraints lead to drifting motion and inconsistent behaviour over time. However, the aim in such a spontaneous scenario is often not a perfectly detailed and complex reconstruction of everything, but a complete and reasonably detailed model that provides enough information for recognition and intelligence. By adapting the scene set up to mimic a real surveillance scene, and moving the focus from hyper-detailed reconstruction to easy and efficient extraction of the important parts, we aim to investigate how applicable this approach is for realistic cases and the limitations it still faces.

1.3 Aim and Scope

This thesis explores solutions for fixed camera sparse-view 4D Gaussian Splatting and segmentation by evaluating various geometric and generative methods, aiming to tackle these problems in a single pipeline. This work can be viewed as a study of advantages and disadvantages of the various approaches, and possibly a baseline for a feasible future combination method. Therefore, this thesis aims to answer the following research questions:

RQ 1: How can a stable 4D Gaussian representation be reconstructed from sparse-view camera input in dynamic scenes?

RQ 2: Under a sparse fixed-camera setting, how do generative completion methods compare to geometry-based approaches in maintaining stable and consistent 4D reconstructions for dynamic objects?

RQ 3: Given a stable reconstruction, how well can dynamic objects be segmented directly in Gaussian space?

1.4 Alignment with Sustainable Development

The Sustainable Development Goals are a list of 17 goals set by the United Nations in 2015 to outline a global effort to end poverty, protect the planet and ensure peace and prosperity for everyone by year 2030. These goals range across various areas of science and development [25]. The outcome of this thesis project aligns with Goal 12: Responsible consumption and production.

Goal 12: Responsible consumption and production.

Commonly applied 3DGS pipelines include processing and rendering hundreds of images, which is a computation heavy process that may take hours of training and intensive use of electricity. In addition to this, the set up for obtaining these images often includes a multi-camera set up with 20-80 cameras recording at the same time, or a single camera that records for significantly longer time which generates even more images to process. When the scene to reconstruct is dynamic, the amount of data images to process also grows linearly per second of video for each individual camera. By investigating alternative approaches that enable less processing with few cameras, while still achieving the required amount of detail for the task, we support the premise of reducing electricity consumption and motivate smaller set-ups that use fewer cameras, which supports responsible consumption of electronic goods.

While this thesis aims to reduce overall processing and power requirement, it should be noted that this is still a project consisting of multiple steps of processing images,



Figure 1.1 Sustainable Development Goal 12

training models and rendering for virtual devices which are all tasks that use power. The outcome is not to get rid of consumption completely, but attempt to receive the desired output with less resources.

1.5 Ethical Considerations

There are several ethical concerns that should be considered throughout this work.

Confidentiality & Privacy

When working with captured images and reconstructions, the main concern that is raised regards confidentiality and privacy. Since this work involves recording of public or private spaces with multiple cameras and high detail reconstruction of the space, anything that is filmed or reconstructed should be publicly available or consented data. Information should be provided and consent should be taken for any person or private property recorded. Any recording of confidential information or persons identity, even with consent, will not be used in the public reporting.

The segmentation step of this pipeline helps ensure better confidentiality and privacy for end results. A recorded scene that is required for analysis might often include other confidential information that should not be shared in the environment. By segmenting and only rendering the relevant part, more privacy can be ensured.

Informed Consent & Voluntary Participation

All participants in the recording and reconstructions should receive sufficient information regarding the process, collected data, and their right to stop recording at any point. Participants should be informed again regarding all uses of recordings and reconstructions, including both visualisation and reporting.

1.6 Related Work

A lot of research has been done following the original 3DGS and 4DGS methods, some of which aimed to tackle the challenges we explore. These papers can be reviewed under two categories: Sparse View Gaussian Splatting and Object Segmentation.

1.6.1 3DGS under Sparse View Camera Setups

Obtaining a stable 4DGS model under sparse views is challenging, since the sparse views contribute fewer similar images making it difficult to obtain initial match pairs with COLMAP (see 2.1.2). Sparse views can cause artifacts and unstable densification. For the dynamic 4DGS case, sparse view can also cause gaussian drifting, where gaussian blobs belonging to one object become part of another object in the next timestamp. Sparse view problem can be improved with either geometric or generative approaches. Geometric approach which use of existing geometry information to provide additional information for a stable reconstruction, such as depth information. Generative approach aims to balance the missing views by creating synthetic training data using Artificial Intelligence (AI) to fill in the gaps created by missing views.

GC-4DGS [16] introduces geometry-consistent priors to improve dynamic Gaussian Splatting under sparse multi-view inputs. The method combines learning-based multi-view stereo (MVS) and monocular depth estimation (MDE), and injects reliable geometric cues into the optimization via cross-view and cross-time consistency filtering and ranking regularization. GC-4DGS primarily targets improving novel-view synthesis under sparse inputs, implicitly assuming that reconstruction quality on observed views is already sufficient. As a result, its optimization focuses on generalization to unseen views rather than explicitly stabilizing reconstructions in known views, which may introduce trade-offs in extremely sparse fixed-camera settings.

Splatography [1] addresses sparse multi-view dynamic reconstruction in film making scenarios by decomposing foreground and background and adopting a two-stage training strategy. Sparse foreground masks, typically provided only at the initial time step, are used to stabilize canonical learning before dynamic deformation modelling is introduced. While this staged design improves training feasibility under sparse views, the method ultimately evaluates performance in terms of novel-view synthesis quality. Moreover, the reliance on foreground masks introduces additional semantic or manual priors that may not be available in general reconstruction settings.

Generative Sparse-View Gaussian Splatting (GS-GS) [14] reduces sparse-view ambiguity by incorporating generative supervision. Diffusion models are used to synthesize fake novel views given known camera poses, and the generated images are used as additional supervision for Gaussian Splatting. Through iterative optimization between the generator and the GS representation, GS-GS improves visual plausibility in unseen views.

Efficient4D [27] approaches sparse dynamic reconstruction from a generative video-to-4D perspective. By first generating temporally and spatially consistent multi-view images and then training 4D Gaussian Splatting on the synthesized data, the method emphasizes efficient free-viewpoint synthesis. So Efficient4D prioritizes generative quality and efficiency over strict geometric consistency with real observations.

4 Camera 4D Gaussian Splatting (4C4D) [37] is a very recent work focusing on a very similar problem set as us, published after most of the work for this project was done and our final pipeline had been established. This work uses 4 cameras placed widely spaced all in front of dynamic object, and observes similar problems as us in terms of accuracy and completeness. While we obtained better results from generative approaches, this work focuses on a geometry based one by adding an additional Neural Decaying Function over the opacity to improve geometric modelling. Their evaluation shows improved results over the State-of-the-Art methods for sparse dynamic datasets.

Although not directly sparse multi-view dynamic reconstruction, several related but distinct problem settings have been explored. Monocular dynamic reconstruction methods, such as Simultaneous Localization and Mapping (SLAM) based approaches (e.g., Instant4D [19] and MoDGS [17]), rely on camera motion to provide multi-view constraints over time, and thus are not applicable to fixed-camera scenarios. Sparse-frame dynamic reconstruction focuses on temporal under-sampling rather than view sparsity, addressing challenges in dynamic modelling instead of geometric ambiguity. Static few-view Gaussian Splatting methods primarily target

novel-view synthesis for static scenes under limited viewpoints, and are fundamentally different from dynamic reconstruction problems.

1.6.2 Gaussian Segmentation

Segment Any 4D Gaussians (SA4D) [8] applies segment anything masks (SAM) [12] originally used for 2D images to dynamic gaussian splatting output. Using a temporal identity feature field applied with the deformation information from 4DGS output, this method achieves dynamic object segmentation with ability to make simple modifications and render high-quality anything masks, as well as reduce gaussian drifting.

These related works are studies with goals similar to ours, but not directly used in this project with the exception of Efficient4D [27] and SA4D [8]. More details will be explained in the following sections discussing methodology.

2 Theory

This chapter provides an overview of the theoretical knowledge relevant for this thesis. It begins with an introduction to Structure from motion and COLMAP (2.1). This is followed by an explanation of Gaussian Splatting (2.2), which is the main reconstruction technique for this project. Finally, the chapter concludes with briefly presenting concepts related to Extended Reality (2.3).

2.1 Structure-from-Motion (SfM)

2.1.1 Structure from Motion

Structure from Motion (SfM) is a computer vision technique for reconstruction of 3D structures from a set of 2D projections from different viewpoints. [31]. The method incrementally creates a point cloud of the 3D structure and calculates the camera positions of the images. The overall pipeline can be seen in fig. 2.1.

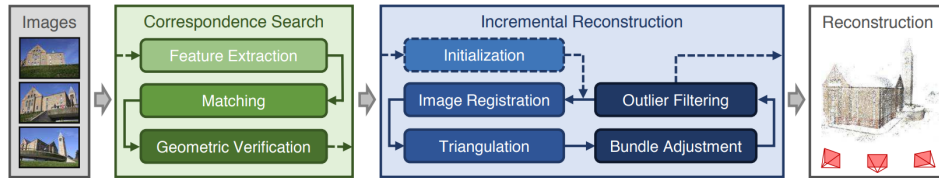


Figure 2.1 SfM pipeline [32]

The process begins with correspondence search, where overlapping parts in the scene are determined by matching points in overlapping images. This is done by determining recognizable points in images (features), matching them across images, and comparing this across images to determine the correspondence and finally verifying via estimating the geometric transformation between view planes. This is followed by incremental reconstruction, where the foundational points are used incrementally to add more images, triangulate scene points and refine the reconstruction.

Structure-from-Motion problem is the fundamental step of reconstruction 3D environments from 2D data. It is also the first step of Gaussian Splatting (See 2.2) where

recorded images are used to calculate both the camera/viewpoint locations and the initial point cloud. It is also the step where lack of varying angles from having sparse cameras causes a difference, and thus an important concept to understand for this project.

2.1.2 COLMAP

COLMAP [31] is a widely used implementation of the entire SfM pipeline. It offers features for reconstruction of ordered and unordered sets of images. The academic work behind this open-source implementation also provides a new strategy for SfM which improved the state of the art of the time, particularly in terms of handling large sets of images within a single pipeline. fig. 2.2 showcases an example COLMAP image cloud reconstruction, which has 21K registered out of 75K images.

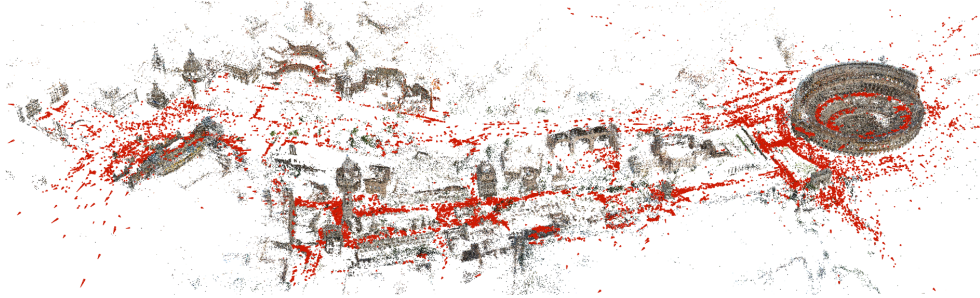


Figure 2.2 COLMAP Generated Point Cloud [31]

COLMAP is the implementation used for calculating camera positions and the initial point cloud as the first step of vanilla gaussian splatting pipeline.

2.1.3 Camera Parameters

Various physical parameters affect how a camera captures images of the real world. These physical parameters are categorized under two categories within the field of computer vision: Intrinsic and Extrinsic parameters [4]. The values representing internal characteristics of a camera, such as skew, focal length and camera center are intrinsic parameters. These values relate to how the camera itself is calibrated, and do not change unless camera configurations are changed. However, an image is not only affected by the camera itself but also its orientation. The values defining a camera's positioning in relation to the image and its rotation are extrinsic parameters. Extrinsic parameters define the coordinate changes between camera space and world space, and need to be recalculated every time camera is moved.

In order to get a good understanding of the scene and obtain an accurate reconstruction, both types of parameters need to be known. 3DGS obtains this information

using COLMAP to calculate values from the images, but when there are not enough images to run COLMAP other strategies will need to be implemented.

2.2 Gaussian Splatting

2.2.1 Neural Radiance Fields

Neural Radiance Fields (NeRFs) [23] are the state-of-the-art method for synthesizing complete restructure for complex scenes. This method uses a non-convolutional deep neural network to obtain volume density and radiance for spatial locations, and that is projected into an image using volume rendering techniques.

While NeRFs are great for photorealistic rendering quality, this method comes at the cost of costly training and rendering for high visual quality, or a trade-off for speed over quality. Vanilla 3DGS can render real-time radiance fields with comparative high visual quality [10] to NeRF at more optimized times in comparison to fastest previous methods such as Plenoxels [35] and InstantNGP [24]. Understanding the limitations of NeRF methods explains why gaussian splatting is the method used for this project.

2.2.2 3D Gaussian Splatting

Rendering 3D representations most commonly represents data as meshes or points, as this works well for fast rasterization on a GPU. NeRF methods instead use continuous scene representations of volumetric rays, which has better scene optimization but is costly in terms of rendering. 3D Gaussian Splatting [10] is a combined technique which aims to be the middle ground with better parts of both methods, and does so by representing the scene in gaussian splats.

A gaussian splat is the smallest representation of a particle of the reconstruction, and it is a 3D ellipsoid primitive carrying information of a mean position, covariance matrix and opacity, together with spherical harmonics coefficients representing colour.

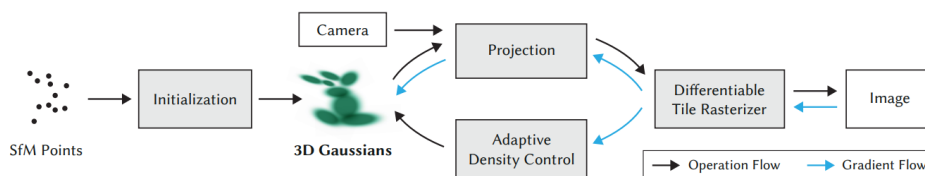


Figure 2.3 3DGS Pipeline [10]

3DGS takes the same input as NeRF methods: many static images covering different viewpoints around the scene, which is used for getting SfM calibrated cameras and an initial set of sparse point cloud also obtained with SfM process. With this input an initial set of 3D gaussians is created, and the properties of the splats are optimized with adaptive density control steps. During this step gaussians are added or removed and modified to get a better representation. As final step, GPU sorting algorithms are used to render real-time rasterization. The efficiency of this method compared to previous versions comes from the tile-based rasterized which allows α -blending of splats and prioritizing visibility order. An overview of the pipeline is seen on fig. 2.3. A representation of how the adaptive density control part optimizes gaussians with modifications is seen on fig. 2.4

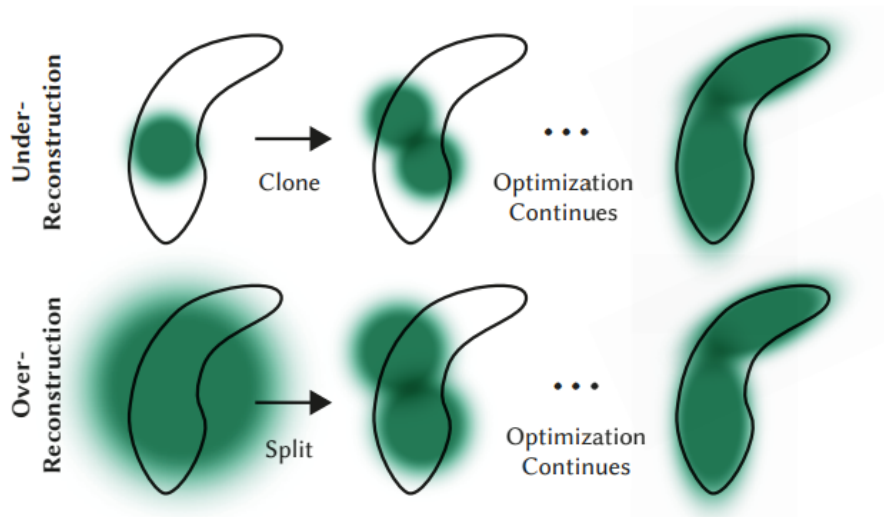


Figure 2.4 Adaptive Density Control [10]

3DGS introduces the baseline method of using gaussian splats in novel view synthesis, and all the evaluated methods in the scope of this project are based on this work primarily.

2.2.3 4D Gaussian Splatting

While 3DGS is a photorealistic and efficient method for novel view synthesis, it is strictly restricted to static scenes. 4D-GS [34] extends on this work by proposing an efficient method to representing dynamic scenes, using a novel representation technique which combines 3D gaussians with 4d neural voxels. This is a faster and more cost-efficient method than applying 3DGS for each individual frame.

In this method, only one set of 3D gaussians is kept. This set of gaussians is combined with a deformation field network, which includes a temporal-spatial structure encoder and a Gaussian deformation decoder. At each timestamp, the set of 3D gaussians is modified in accordance to the information from the deformation function. This modification includes both position and shape modifications for accurate motion representation. The overall pipeline can be seen in fig. 2.5

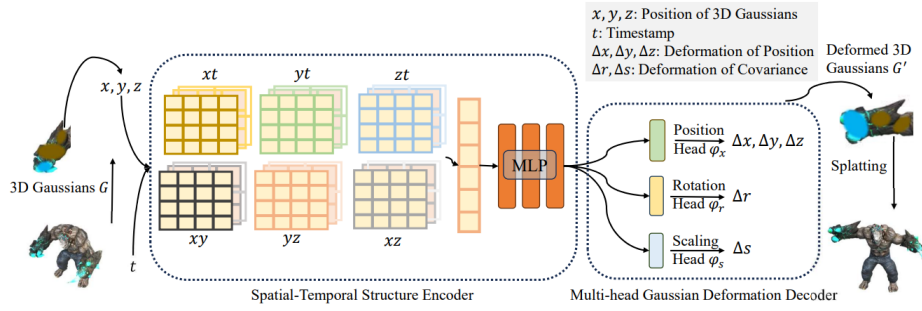


Figure 2.5 4DGS Pipeline [34]

2.3 Virtual Reality

As this project uses virtual reality (VR) for visualization of the final 3D renderings, it is important to mention what VR is and why it is a good choice for this purpose.

There are various definitions of VR, but the overall encompassing definition is that VR is a synthetic computer-generated environment or simulation where the user is isolated from external observation of their surrounding environment [7]. The building blocks making up VR environment are 3D artificial environment and a head tracked display. This is typically accompanied with additional tracking and sensory input and outputs to increase the feeling of being there.

VR has been useful in improving discovery in fields such as palaeontology or investigating brain tumours, and is an effective method for big data visualization [3]. This is due to its immersive and realistic nature, where users experience an intuitive method of interaction with the environment, particularly in 3D. [20]. As our end results are 3D modelling of real-life objects, we believe the most intuitive and easily observable way for interacting with the results is a simple VR environment.

3 Project Overview

This chapter elaborates on the technical setup and benchmark dataset (3.1). Then 3.2 explains the results of our evaluation of the baseline methodology, and finally 3.3 briefly talks about the calibration process.

3.1 Technical Setup and Data Specification

Throughout this project we used the same technical setup, recording environment and public datasets to keep a fair and easily comparable measurement. Self recorded evaluations in this project were trained and rendered on a Linux Ubuntu machine with a single NVIDIA GeForce RTX 5090 GPU and 32 GB of VRAM (Video Random Access Memory) using a conda environment with CUDA 13. For the baseline evaluations, a Linux Ubuntu machine with a single NVIDIA GeForce GTX 1650 GPU and 4 GB of VRAM (Video Random Access Memory), also using a conda environment with CUDA 13. As different works have different version requirements, separate virtual environments were set up according to individual guidelines. For the visualization segment of the pipeline, Meta Quest 3 [21] headset and Unity 6 game engine were used, along with OpenXR SDK for VR controls.



Figure 3.1 Meta Quest 3 HMD [21]

As training input, both publicly available training datasets and custom self recorded data were used. The publicly available training datasets can be accessed online *Tanks and Temples* [13] for static and *Neural 3D Video* [15] for dynamic.

For the self recorded data, a 4.5x4.5 m² indoor area was used as setting. This environment was used with different camera placement set ups of four cameras, with

the goal of achieving broad coverage around the area. Two different models of dome cameras were used for recording, so the resulting images were manually fixed to have 1024x1024 resolution. Different static and dynamic scenes were recorded, changing in complexity in terms of movement and amount of objects in the area. Camera placements tested have been visualized in fig 3.2 along with the names they will be referred with throughout the rest of this paper.

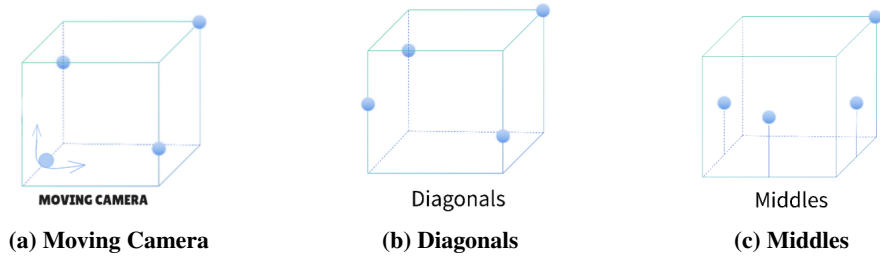


Figure 3.2 Camera Placements

3.2 Evaluation of Baseline Methodology

As the first step of our pipeline, we set up and verified the results of vanilla 3DGS and 4DGS. The following subsections will detail our process and show results from various evaluation trainings. Overall, our evaluation of both vanilla 3DGS and 4DGS shows that modifications are necessary for a good reconstruction under sparse views.

3.2.1 3DGS with different Parameters

First, we set up 3DGS[10] and trained using publicly available tanks and temples[13] dataset. From this dataset, we tested and evaluated with *Truck* and *Playroom* datasets. For multiple runs each dataset, we changed training parameters to record and compare changes on training time, qualitative success of the output and splat count.

Training Truck01 (fig. 3.3) was ran with the parameter modifications : densify grad threshold = 0.2, densification interval = 1000, test iterations = -1. These modifications shorten the training process, allowing the training to run with less VRAM. The training took 45 minutes and the resulting reconstruction has 95,268 splats . Visual result can be seen in figure. This training priorities efficiency and fast results with the parameters. This results in a fast training at the cost of many floaters and imperfect reconstructions. There is also a lot of fragmentation, particularly in the environment around the truck.



Figure 3.3 Reconstruction Truck 01

To see if our processing/converting step is also good, using the same image dataset we did the preprocessing manually and used exact same parameters for training to see how much the results differs. This second training took 38 minutes and resulting reconstruction has 42,750 splats. The resulting reconstruction and some details are shown in fig. 3.4. We decided that the loss of details for minimal training time efficiency is not a reasonable trade-off. The loss of detail is visible well in the truck logo text, where in truck01 the first line "San Pedro" is more legible and the letters are sharper.

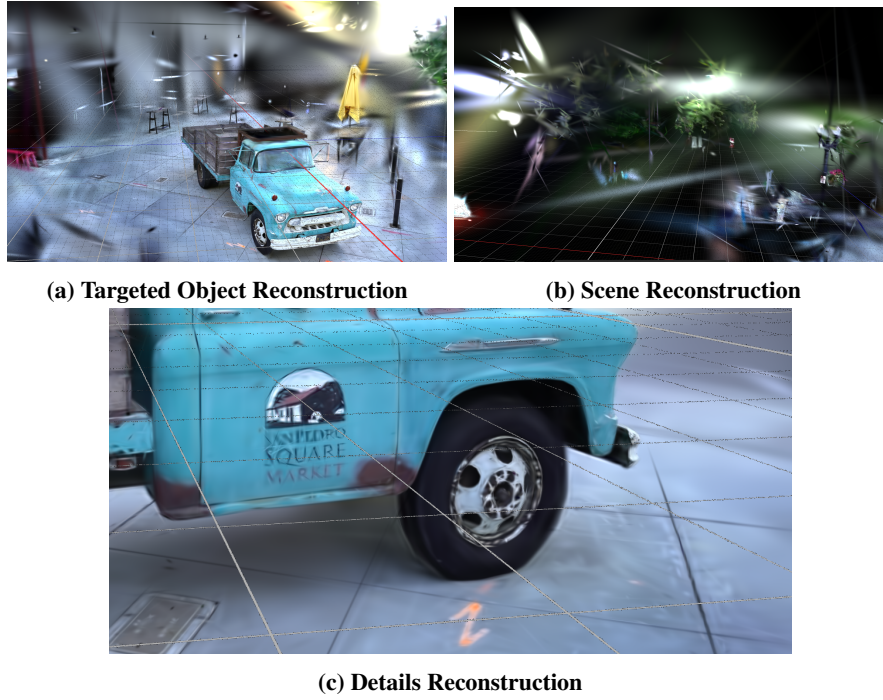


Figure 3.4 Reconstruction Truck 02

We also tested out how much GPU matters, so we used the same dataset to train on the CPU only. With parameters data device = CPU and $r = 2$ (reduce quality by 1/2, so that CPU can handle the training). This took 52 minutes of training, and resulting reconstruction has 1,075,856 splats. Although removing the memory limitation resulted in much higher splat count, which should allow for better details, CPU processing did not receive the same level of smooth blending. The edges between gaussians ended up a lot sharper, resulting in a segmented look. The result can be seen in fig.3.5. Looking at the writing detail, it can be seen that although this one has more detail and more legible writing the overall image result is not as smooth as the previous results. However, the high splat count allowed for more of the outer environment to be generated, showing parts of the street not visible in previous results.

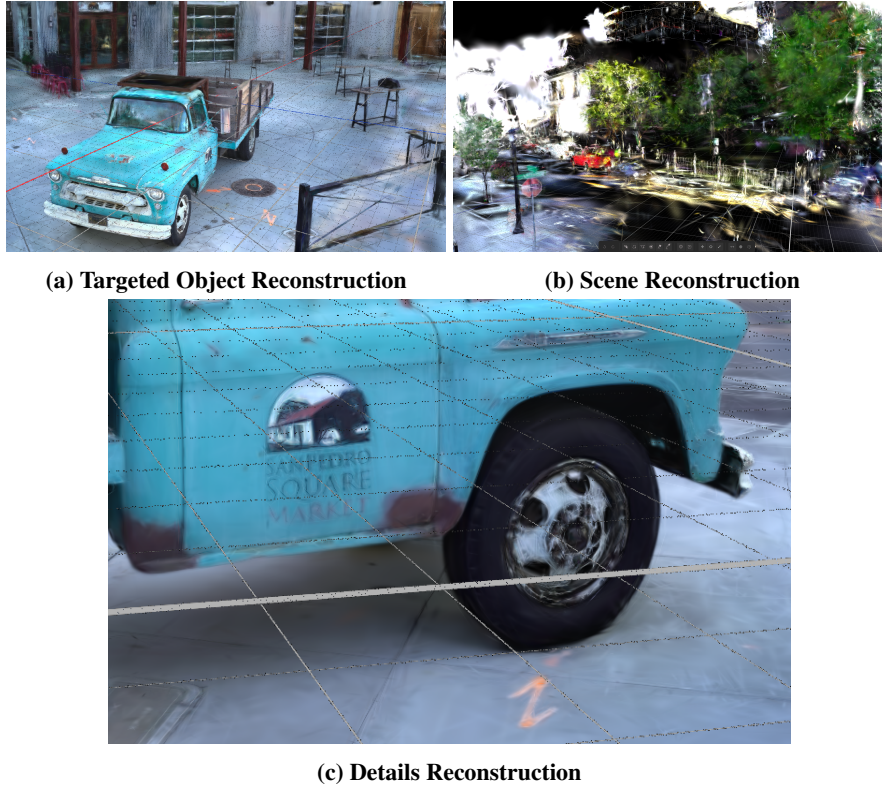


Figure 3.5 Reconstruction Truck 03

Finally, we wanted to test with the default parameters, which resulted in the slowest output with the most hyper-realistic reconstruction.

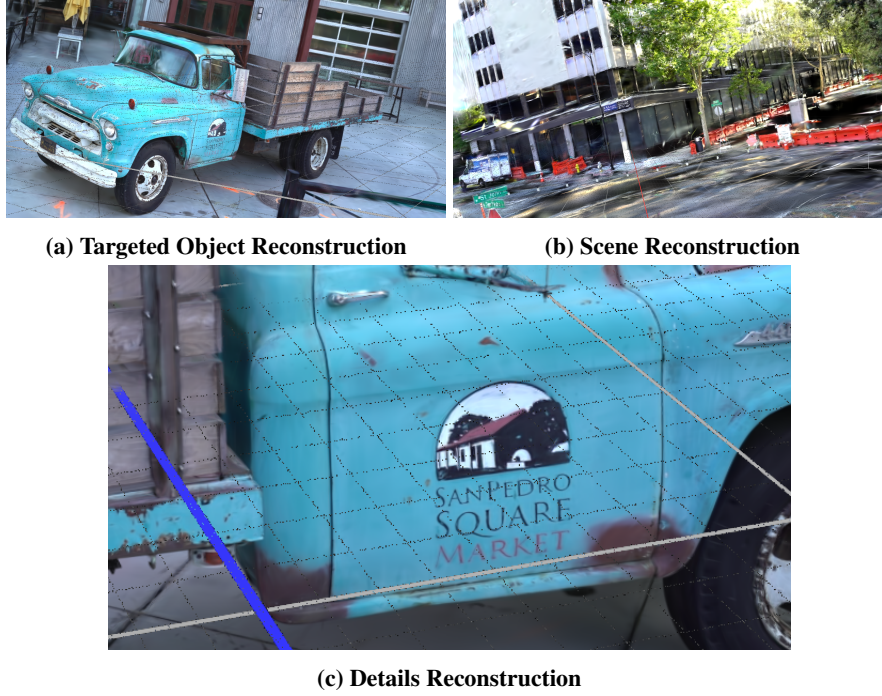


Figure 3.6 Reconstruction Truck 04

3.2.2 Camera and Camera Position

After being confident that our set up successfully runs 3DGS with publicly available datasets, we moved on to testing with our own data. First, we picked a small test object (fig 3.7) and using a single camera, we recorded a one minute input video going around the object at different heights for three rounds. As this recording was meant only to verify our own input can work with the method, image resolution and full view coverage were not prioritized. The one minute input video was split into images with 5 FPS (Frames per Second), generating a total of 300 input images. We ran vanilla 3DGS with default preprocessing, reducing the image quality to 1/4. The training took 42 minutes on a GeForce GTX 1650 GPU and 4 GB of VRAM.



Figure 3.7 Small doll

The resulting reconstruction has 795,800 splats, and views from different angles are shown in fig 3.8. Even though the image quality was reduced and the view coverage was not complete, a full reconstruction of the object was obtained, with surprisingly good amount of detail as fur is a difficult material for good reconstructions. Although it was not one of our primary benchmarks and thus not at all a focus of the input video, the environment had a recognizable and detailed reconstruction. There were a lot of floating artifacts and blurry gaussians, but this was to be expected considering the recording conditions and the environment not being isolated and kept static.

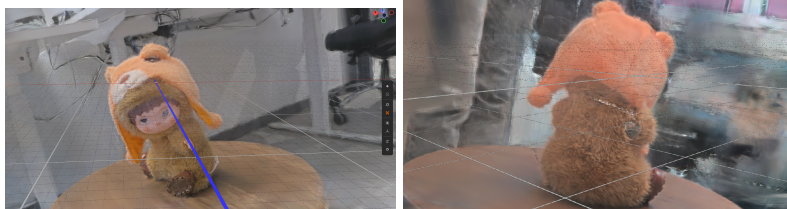


Figure 3.8 Doll 3D Reconstructed

As the next step we set up our scenario environment with four cameras in different set ups to go through the pipeline using only fixed cameras. First, we recorded static and dynamic short scenes with *Middles* and *Diagonals* (See section 3.2) setups. The purpose of *Diagonals* setup was to create parallax difference in order to observe if providing different viewing angles of the same object helps differentiate background and foreground even when the cameras are fixed. The purpose of *Middles* setup was to create an evenly distributed view with more coverage around average human eye-level area likely to have important details, while keeping the realistic case ceiling camera height for one of them. For both setups, however, the static scene 3DGS pipeline failed on the image preprocessing stage, due to not finding enough good initial pair matches (See section 2.1). This result verified that the vanilla 3DGS method cannot generate reconstructions with only four fixed cameras.

Before moving on to evaluating various improvements, we tried one more setup *Moving Camera* which is a modified version of *Diagonals*, where one of the cameras slowly moves between two fixed cameras. This setup was only used to test whether good results can be obtained if the preprocessing completes successfully by providing some initial pairs but not complete coverage. With this modification, the entire pipeline could be ran and a reconstruction was obtained. The most complete part of the resulting reconstruction can be seen in fig 3.9. However, the tested case with moving camera did not satisfy our scenario prerequisites and the resulting 3D reconstruction was not stable enough for easy recognition.



Figure 3.9 Reconstruction from Moving Camera

3.2.3 4D Gaussian Splatting

For dynamic reconstruction, we first evaluated Vanilla 4DGS [34] using public dataset images. This is because since our current setup cannot reconstruct a static scene, a more complex dynamic scene will also not be reconstructed. For this evaluation we used synthetic dataset bouncing balls from D-NeRF [28] and real life data cook spinach from Neural 3D video [15] dataset. D-NeRF [28] is a dynamic neural radiance field method, but in this work we only used their dataset. These two datasets for chosen as one is a small scale dataset covering the most simple requirements for an easy reconstruction, and the other one is real life data with many environmental details for comparison and a moving human target similar to our goal case. The reconstruction outputs in comparison to ground truth can be seen in figures 3.10 and 3.11. Note that these are example screenshots from a dynamic reconstruction.

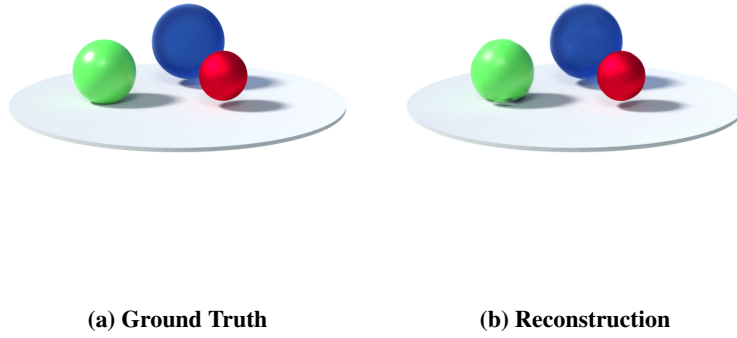


Figure 3.10 Synthetic Data



Figure 3.11 Real Life Data

Since the sparse settings could not generate a reconstruction of the static scene, we decided to also evaluate sparse vanilla 4DGS on a static scene instead of a dynamic one, as motion will make getting matching points an even more complex task. With the *Diagonals* setup, we tested preprocessing data ourselves step by step to see the amount of matches. Taking different frames for several iterations, feature matching resulted in 12, 5, 5, 7 points matching from two images. This means only two cameras can be matched for seeing the same point at a time, and there are very few matches. Next we changed to *Middles* set up to test if matching heights makes it easier to get more initial pairs for feature matching. This resulted in more matches, but still not enough to continue with next steps.

Overall, our evaluation of both vanilla 3DGS and 4DGS shows that additional information is required for a good reconstruction under sparse views. As the problem with sparse input is observable from the first preprocessing step of obtaining matching initial point pairs to calculate camera poses, our work initiated with solutions to camera pose calculation by introducing external pose prior to bypass the original 4DGS data processing related to COLMAP camera poses initialization stage.

3.3 Calibration

As observed in the evaluation of baseline methodology (see section 3.2), the main prerequisite to reconstructing the scene is to accurately determine the camera poses. A setup with only four cameras makes it difficult to obtain poses accurately using COLMAP. Having fixed and known camera setup can be a replacement by manually calculating required parameters. The main advantage of having fixed cameras in a known environment is that we can use the cameras' positions and orientations to calculate intrinsic and extrinsic matrices, manually obtaining the required pose information.

3.3.1 ChArUco Calibration

ArUco Markers [5] are an automatically generated, fiducial marker system popularly used for camera pose estimation. They are square markers that encode a unique value useful for correct detection of marker and image correction. These markers can be generated using openly provided dictionaries which are optimized for distance within marker and the number of bit transitions. An example set of ArUco Markers can be seen in fig. 3.12

ArUco markers are popular for fast detection and versatility, but can have accuracy problems particularly with the corner positioning. Another method for marker detection and calibration is using chess boards, with which the corners can be identified more precisely as they are always surrounded by two black squares [26]. The lim-



Figure 3.12 Aruco Markers

itation of the chessboard method is that the entire board must be visible with no occlusions, which makes it tricky particularly with multi camera calibration.

The name Charuco is a mesh of the names of the two methods, and describes the combined method visualized in fig. 3.13 This system allows for using the advantages of both calibration methods to get camera values.

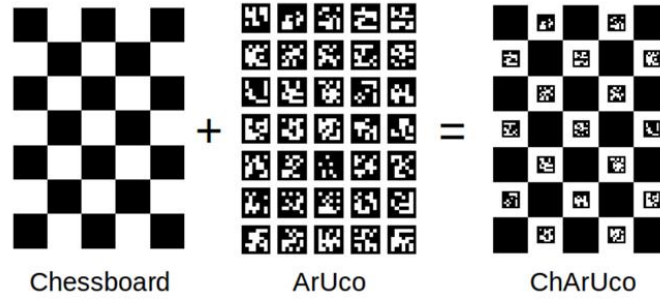


Figure 3.13 Charuco Method

We found the camera poses obtained from this calibration method to be good, so we used it instead of COLMAP initial poses for our work. Eventually, for Diffuman4D (See section 5.2.3) we needed very precise camera poses where results from this method were not enough, so we did further research to find a more advanced multi-camera calibration method.

3.3.2 Multi-Camera Calibration

Accurate camera parameters are essential for both the generation and reconstruction stages. We do multi-camera joint calibration using MC-Calib[29], a robust toolbox for the simultaneous calibration of multi-camera systems via fiducial marker-based patterns. As the calibration target, we construct a 30×30×30 cm cubic object with a ChArUco board printed on each face. This 3D calibration object allows multiple calibration boards to be observed at the same time across cameras with partially overlapping fields of view, which is well-suited to our four-camera configuration

where adjacent cameras share limited common visibility. The calibration procedure produces for each camera a set of intrinsic parameters, which are focal lengths f_x, f_y , principal point (c_x, c_y) , and lens distortion coefficients $(k_1, k_2, p_1, p_2, k_3)$, as well as a 4×4 extrinsic pose matrix representing the camera's position and orientation in world space.

4 Geometry Based Methods and Evaluation

Before moving to generative completion methods, we first explored whether additional geometric constraints could stabilize 4DGS under our sparse fixed-camera setting. These experiments were conducted on a static scene instead of a dynamic one. The reason is that static reconstruction is a simpler version of the same sparse-view problem. If the geometry cannot be reconstructed reliably when the scene is static, it is unlikely to become more stable when human motion, temporal deformation, and changing occlusions are introduced.

The first geometry-based attempt was to add a depth prior into the 4DGS training process [16, 17]. Since the vanilla 4DGS optimization mainly relies on image reconstruction loss [34], the model can still fit the training views while placing Gaussians at incorrect depths. This problem becomes more severe under sparse views, where only a few cameras provide geometric constraints. To reduce this ambiguity, we generated external depth information using COLMAP dense reconstruction based on the calibrated camera poses [31, 32]. The 4DGS renderer was then modified to output a rendered depth map together with the RGB image. During training, each rendered depth map was compared with its corresponding COLMAP depth prior using a masked log-depth L1 loss. Invalid depth pixels were ignored, and the depth loss was added to the original photometric loss with a small weight.

This depth loss provided an additional geometric anchor for each training view. However, the resulting reconstruction was still not stable enough for free-viewpoint rendering. Some known camera views became partially recognizable, but the model still showed fragmented geometry, floating artifacts, duplicated structures, and unstable shapes when rendered from novel viewpoints. This indicates that per-view depth supervision alone does not solve the sparse-view ambiguity. It constrains the geometry along each individual camera ray, but it does not force different cameras to agree on the same shared 3D surface.

To address this limitation, we further explored cross-view geometric consistency [16]. The idea was to use synchronized images from different cameras at the same timestamp. A depth point from one camera can be back-projected into 3D space using the known camera parameters, and then reprojected into another camera view. The projected depth can then be compared with the rendered depth in the target

view. In this way, the constraint is no longer applied only within one camera view, but across multiple views of the same scene. To support this, the training sampler was modified so that each training batch could contain multiple camera views from the same timestamp.

Although cross-view consistency gives a stronger geometric constraint than independent depth supervision, it still did not produce a usable reconstruction in our setting. The output remained broken and noisy, with repeated parts appearing in different directions. In known views, only limited parts of the scene were recognizable, while the reconstruction quickly collapsed when the viewpoint moved away from the training cameras. This result suggests that the main limitation was not only the lack of depth regularization, but the lack of visual coverage itself. With only four fixed cameras, large parts of the scene remain unobserved, and geometry-based constraints cannot recover reliable content in those regions.

Based on these observations, the geometry-based route was not selected as the final solution. Depth priors and cross-view consistency can provide additional constraints, but they do not create the missing observations required for full 360° reconstruction. Therefore, the project moved toward generative completion methods, where additional novel-view images are synthesized before training the final 4DGS model.

5 Generative Solutions and Evaluation

This chapter begins with an overview (5.1), details the methods we tried and the selection (5.2) and finally ends on our takeaway (5.3).

5.1 Overview

Reconstructing a dynamic human object with full 360° coverage from only four fixed cameras has an obvious challenge: the sparse viewpoint distribution causes large portions of the object unseen, making geometry-based reconstruction methods ill-posed. To address this, we attempt an alternative generative completion strategy that uses a video diffusion model to synthesize dense multi-view data from the sparse camera inputs, which are then used to reconstruct a 4D Gaussian Splatting representation of the object.

The pipeline consists of four stages. First, the four cameras are calibrated to obtain accurate intrinsic and extrinsic parameters. Second, foreground segmentation masks are extracted for each view. Third, a spatio-temporal diffusion model is used to generate photorealistic images of the targeted object at a large number of novel viewpoints, effectively supplementing the sparse input. Finally, the generated multi-view images are used to train a 4DGS model, reconstructing a dynamic 3D representation that provides free-viewpoint observation.

5.2 Method Selection

Generating full coverage observations from sparse camera inputs requires a model capable of producing spatially and temporally consistent novel-view images of a dynamic human object. We evaluate three candidate approaches and analyse their limitations before deciding Diffuman4D [9] as our final choice.

5.2.1 Single-View Video Generation (Efficient4D)

Efficient4D [27] is a video-to-4D generation framework that takes a single-view monocular video as input, generates spatially and temporally consistent multi-view images using a modified SyncDreameer [18] model, and reconstructs a 4D Gaussian representation from the generated images. Even though the method performs well on its provided benchmark dataset, it is designed for single-view input and does not incorporate any human-specific priors. When applied to our own data, the generated multi-view images have severe body distortion and implausible motion, including mirrored limb positions and incomplete motion. There are also depth inconsistencies, where 3D objects are reconstructed as flat. Some results can be seen in fig 5.1. Furthermore, the reconstruction quality obtained via its associated 4DGS branch doesn't meet our expectation, and the output format is incompatible with our 4DGS viewer. We therefore exclude Efficient4D from further consideration.



Figure 5.1 Efficient4D results

5.2.2 Static Novel View Synthesis

Stable Virtual Camera (SEVA) [36] is a diffusion model for static novel view synthesis that takes one or more images as input and generates images at arbitrary target camera poses. Since SEVA operates on static scenes, we design a two-stage strategy to handle dynamic content: for each selected timestamp, the corresponding frames from the four cameras are provided as input to SEVA to generate a static multi-view video at that moment; this process is repeated across all selected timestamps to produce a set of per-timestamp multi-view videos, which are then reorganized into a dynamic multi-view dataset and fed into 4DGS for reconstruction.

Experiments on the cook spinach [15] dataset demonstrate that this strategy is achievable when the target viewpoints are restricted to the region covered by the input cameras, giving a recognizable reconstruction with visible motion. However, when applied to our setup for full 360° dynamic human view synthesis, the generated results are insufficient for high-quality reconstruction. We conclude this to three compounding factors. First, our four cameras are separated by distances of several meters, and the target viewpoint trajectory has a full 360° range, which is far beyond the angular range that SEVA can reliably generalize to. Second, since SEVA processes each timestamp independently, variations in the object’s pose across timestamps cause the independently generated multi-view videos to be mutually inconsistent. Third, the capture space is open beyond the 4×4 meter frame structure with no physical boundaries, introducing complex and varying background content that further causes unstable generation. These factors together prevent SEVA from producing sufficiently consistent multi-view data for reliable 4DGS reconstruction in our setting.

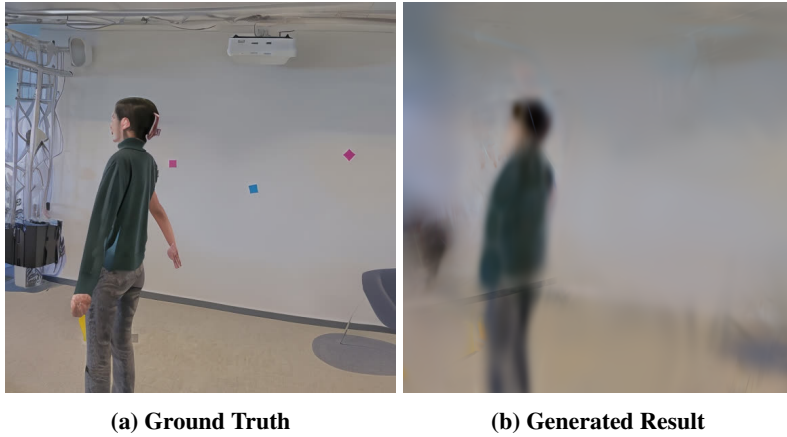


Figure 5.2 SEVA Result

5.2.3 Sparse-View Human Video Diffusion (Diffuman4D)

Given the sparse four-camera input, we adopt Diffuman4D [9] to generate dense multi-view human videos. Diffuman4D is a spatio-temporal diffusion model designed for high-fidelity novel view synthesis of human performances from sparse-view video inputs. It takes as input a set of synchronized multi-view videos along with their corresponding camera parameters and foreground masks, and produces photorealistic images of the object at a large number of target viewpoints.

The core of Diffuman4D is a sliding iterative denoising mechanism that operates over a 4D latent grid, where each latent encodes the appearance, camera pose, and human pose at a specific viewpoint and timestamp. Denoising is performed by alternately sliding a window along the spatial (multi-view) and temporal dimensions, allowing information to propagate across the full grid without requiring all frames to be processed simultaneously. This strategy ensures spatio-temporal consistency across the generated views while keeping GPU memory consumption tractable. In addition, 2D human skeletons are first estimated for each frame using Sapiens [cite], a foundation model for human vision tasks, and subsequently triangulated across views to obtain a 3D skeleton sequence and projected into each target viewpoint as a structural conditioning signal, which further improves the accuracy of human body shape and motion in the generated outputs.

In our pipeline, the four calibrated cameras serve as the input views, and Diffuman4D generates images at a set of target viewpoints specified by their camera poses. The outputs consist of foreground-only RGBA images stored in the images alpha directory, together with a transforms.json file containing the camera intrinsic and extrinsic parameters for all generated views. These outputs form the multi-view dataset used for the subsequent 4DGS reconstruction stage.

Diffuman4D requires target camera poses to define the viewpoints for novel view synthesis. To achieve full 360° coverage around the object, we generate a set of virtual camera poses distributed uniformly around the capture space, derived from the four real camera poses obtained during calibration.

The scene center $\mathbf{p}^*$ is estimated by solving a least-squares problem over the four real camera positions and their viewing directions. The mean camera radius $\bar{r}$ is then computed as the average distance from the real camera centers to $\mathbf{p}^*$. Virtual cameras are arranged in multiple concentric rings at varying radii and heights relative to $\mathbf{p}^*$, with cameras uniformly distributed in azimuth within each ring. Each virtual camera is oriented toward $\mathbf{p}^*$ using a standard look-at formulation, and its intrinsic parameters are set to the average of the four real camera intrinsics. The resulting virtual camera poses, together with the four real camera poses, are passed to Diffuman4D as target viewpoints.

Table 5.1 Comparison of generative methods evaluated in this project.

Method	Required input	Strength	Limitation in our setting	Decision
Efficient4D	Single-view monocular video	Provides a complete video-to-4D generation and reconstruction pipeline	Designed for single-view input and does not include human-specific priors. On our data, it produced body distortion, mirrored limbs, incomplete motion, and depth inconsistency	Excluded
SEVA	Single or multiple images with target camera poses	Flexible pose-conditioned novel view synthesis and can generate recognizable results when target views are close to input views	Designed for static scenes. In our per-timestamp strategy, independently generated timestamps were not consistent enough for full 360° dynamic human reconstruction	Excluded
Diffuman4D	Sparse synchronized real-camera videos with foreground masks, real camera poses, and target camera poses	Designed for sparse-view human video synthesis; uses spatio-temporal denoising and skeleton conditioning; outputs foreground RGBA images and camera parameters	Requires accurate calibration and clean masks. Generated videos can still contain small temporal appearance inconsistencies	Selected

5.3 Takeaway

Diffuman4D [9] is specifically designed for the task of novel view synthesis of human performances from sparse-view video inputs, which closely matches our problem setting. Its sliding iterative denoising mechanism ensures spatio-temporal consistency across a large number of generated viewpoints and timestamps, which solves the main challenge of the previous two models that is cross-timestamp consistency. Its skeleton-based conditioning provides structural guidance that is partic-



(a) From 40 virtual cameras setup

(b) From 58 virtual cameras setup

Figure 5.3 Diffuman4D results - Wave in Place



(a) From 76 virtual cameras setup

(b) From 100 virtual cameras setup

Figure 5.4 Diffuman4D results - Walking Person

ularly effective for handling complex human motion and clothing deformation. Furthermore, its outputs include per-frame foreground RGBA images at all target view-points along with corresponding camera parameters, forming a complete multi-view dataset directly suitable for 4DGS reconstruction. Based on these properties and the superior visual quality of its generated results on our data, we adopt Diffuman4D as the core generation module in our pipeline.

6 Segmentation

Our initial research question had the goal of segmenting a dynamic object from the reconstruction within gaussian space. For this, we began with the method SA4D [8] but along the pipeline we ended up preferring an initial segmentation of 2D images, where we only generate the dynamic object and not the rest of the environment.

6.1 Segment Any 4D Gaussians (SA4D)

SA4D [8] segments objects directly in the gaussian space. Their pipeline consists of several stages including removal of artifacts and using segment anything masks to determine object to be extracted. While the code is publicly available, there is no official structured code release. Also, several stages of the pipeline have details omitted. While this work accomplishes our initial goal of segmentation in the gaussian space, as there is no official release recreating the results requires several manual steps.

The segmentation process includes manual labelling of objects and SA4D makes use of Tracking-Anything-with-DEVA [2] to initially segment videos. This model and the continuing pipeline stages required high processing power. As our goal required only one object to be segmented and not the entire scene to be labelled, we decided to make the pipeline lighter by replacing this labelling with creating 2D segmentation masks ourselves. This modification also aligns with our sustainable development goal, as having a lightweight pipeline with less power consumption was one of the targeted improvements.

6.2 2D Segmentation and Mask Generation

We used SAM2 [30] on frames to create black and white segmentation masks for the dynamic target. While this step was initially meant to replace DEVA for SA4D, we decided that if we can reconstruct the object after segmenting in the 2D space, it requires less processing overall and only outputs the requirements, which is a cleaner and more efficient pipeline. During the testing of various sparse methods, we also tried using the black and white segmentation masks or a white background - coloured target segmented image as input. While some branches like *multipleview*

which is originally meant for self-recorded data failed to match enough points for reconstruction; *Dnerf* branch of vanilla 4DGS with memory use modifications successfully reconstructed only the object. Our chosen sparse method Diffuman4D has its own mask creation stage in the pipeline, however created masks are not very clean and have cross-object merging particularly in furniture cluttered environment. Because of this, we used our SAM2 mask script to replace the mask creation step of Diffuman4D. An example of how the masks differ can be seen in fig. 6.1.

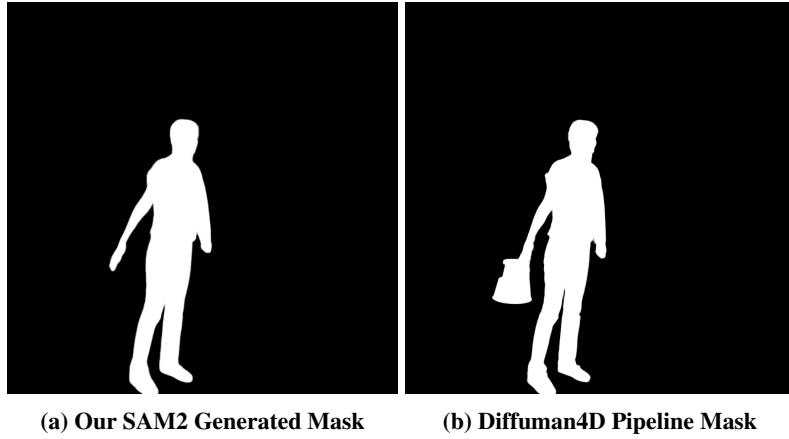


Figure 6.1 Masks Generated by Different Methods

7 VR Visualisation

Final part of the pipeline is importing dynamic gaussian to a VR application to offer the user an easy and simple way of observing and interacting with the reconstruction from different angles. The VR environment is intentionally kept simple for better observation of the target object.

Web viewers such as SuperSplat¹ which are useful for visualizing 3D splats cannot display dynamic scenes.

7.1 Existing Methods

While the term 3DGS is a general reference to method established by Kerbl et al [10], the same cannot be said for 4DGS or dynamic gaussian splatting. Most commonly used and referred version is [34] however there are still different methods. As 4DGS output has not been standardized yet there are no established official frameworks for it. The provided 4DGS web viewer is not compatible with linux, so a different approach is required.

Meta Spatial SDK [22], one of the VR SDK options compatible with Meta Quest, has recently released a developer feature for representing Splats, which at the time of this project is an experimental feature and unstable. The feature only allows for a single gaussian splat to be rendered at a time, only allows for relatively small splat counts and requires 10-15 seconds switching between individual splats. All of these observations lead to the conclusion that although it is promising to have an upcoming official release for visualizations, at the time it is unusable particularly for the case of dynamic scenes.

There are several open source implementations to represent 3D gaussian splats in unity. The most popular one is "Gaussian Splatting playground in Unity" [6]. This implementation makes use of the graphics pipeline to add gaussians rendering stage. An alternative open source approach to displaying gaussians in unity is SplatVFX [33]. This plug-in simulates the gaussian splats as a visual effect particle with infinite lifetime, with additional help from shaders to apply gaussian blurring between splats. This method works very well for VR since there is visual effect support, and it is not

¹ <https://supersplat.at/>

heavy rendering. All of these open source implementations are focused on static 3D models, so additional modifications are needed for dynamic.

7.2 Our Solution

The vanilla 4DGS paper we worked with uses deformation table on a single 3DGS output to apply positional changes as a time function, but we can't run this deformation algorithm within unity in real-time while rendering for Virtual reality due to the limiting processing power of the quest device. so as our solution, we combined multiple methods together. We apply the deformation function to the singular 3DGS output and generate per frame 3D gaussians. we use the method from Splatvfx to render one frame at a time. We modified the plug-in code so that instead of having the rendering bound to the splat, we have a custom render manager which renders two at a time with next frame being invisible to get a smooth transition between sequential frames, resulting in a stop motion animated dynamic gaussian result.

We intentionally kept the VR environment simple and empty in order to focus on the object, we added some simple ease of interaction features. These include the option of using the joystick to walk and turn instead of physically, and teleporting to directly reach behind object or anywhere within the virtual room. Some visuals from the VR visualization can be seen in 7.1 and 7.2.

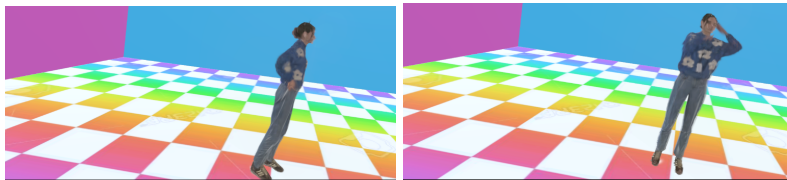


Figure 7.1 VR Screenshot for Macarena



Figure 7.2 VR Screenshot for Walk

While the method used satisfies the requirements for working in stand alone mode, this severely limits ease of interaction as heavy renderings cause lagging which may cause nausea in most users. Also, stand alone mode requires smaller file sizes and overall project size, which limits our dynamic time and quality. As representation and

easy interaction were the primary goals over stand alone VR, for our demonstrations we built our visualizer only in PCVR mode.

8 Final Pipeline

The evaluations and experimentation with different methods led us to a final pipeline, including additional steps from the initial calibration and continuing with our best results method tested. The entire pipeline can be seen in (FIG pipeline), and a breakdown of each step is given.

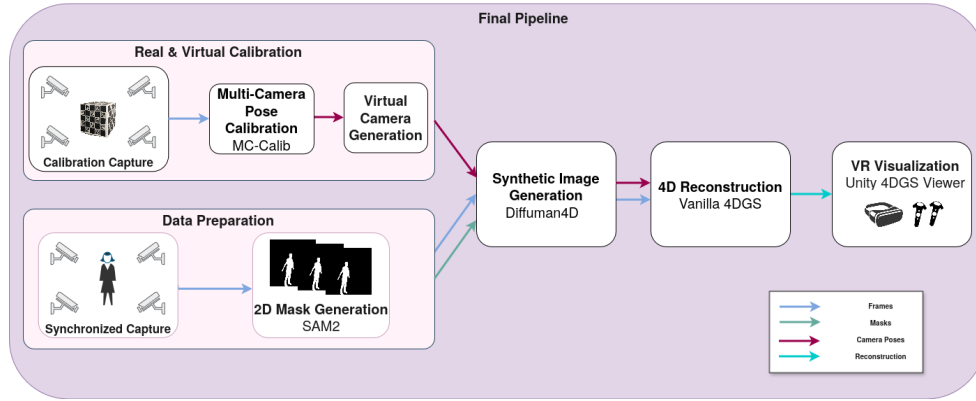


Figure 8.1 Entire Pipeline

Calibration Capture and Multi-Camera Pose Calibration

In order to do fixed camera calibration, a three minute video is captured holding a charuco cube. The cube is held in view of each camera for 30 seconds, slowly rotating and tilting. This video only needs to be captured once, and the same calibration result can be used for many reconstructions as long as the cameras are not modified before the synchronized capture.

Since MC-Calib[29] outputs camera poses in OpenCV convention, where the Y-axis points downward and the Z-axis points forward, a coordinate system conversion is required to match the OpenGL/NeRF convention expected by Diffuman4D(ref), in which the Y-axis points upward and the Z-axis points backward. This is achieved by negating the second and third columns of the rotation component of each extrinsic matrix. The converted parameters are serialized into a transformsjson

file, and it serves as the camera configuration input for the subsequent generation stage.

Data Collection and Pre-processing

To capture dynamic human performances from multiple viewpoints, we construct a synchronized four-camera capture system within a controlled indoor environment. The capture space is built as a 4×4 meter open frame structure, consisting of four vertical poles and four horizontal bars forming the top perimeter. One camera is mounted on each vertical pole at the upper-middle position, with all four cameras placed at a consistent elevation. The orientation of each camera is adjusted so that the object appears fully within the frame and occupies the maximum possible proportion of the image area. Each recording session captured simultaneously across all four cameras.

All captured footage is pre-processed into a unified resolution of 1024×1024 pixels. For camera calibration, frames are extracted at 2 FPS. For the subsequent generation stage, frames are extracted at 10 FPS.

Foreground Segmentation and Mask Generation

To separate the human object from the background, foreground segmentation masks are generated for each camera view. Since the default background removal script provided by Diffuman4D produced no clean separation between the object and the background, we replace it with a custom segmentation pipeline that provides cleaner foreground masks. The resulting masks are stored in the fmask directory as required by Diffuman4D, and are used during the generation stage to suppress background regions and focus synthesis on the human object.

Multi-View Data Generation and Novel View Synthesis via Diffuman4D

Given the sparse four-camera input, we adopt Diffuman4D [9] to generate dense multi-view human videos. Diffuman4D is a spatio-temporal diffusion model designed for high-fidelity novel view synthesis of human performances from sparse-view video inputs. It takes as input a set of synchronized multi-view videos along with their corresponding camera parameters and foreground masks, and produces photorealistic images of the object at a large number of target viewpoints.

The core of Diffuman4D is a sliding iterative denoising mechanism that operates over a 4D latent grid, where each latent encodes the appearance, camera pose, and human pose at a specific viewpoint and timestamp. Denoising is performed by alternately sliding a window along the spatial (multi-view) and temporal dimensions, allowing information to propagate across the full grid without requiring all frames to be processed simultaneously. This strategy ensures spatio-temporal consistency across the generated views while keeping GPU memory consumption tractable. In addition, 2D human skeletons are first estimated for each frame using Sapiens [11],

a foundation model for human vision tasks, and subsequently triangulated across views to obtain a 3D skeleton sequence and projected into each target viewpoint as a structural conditioning signal, which further improves the accuracy of human body shape and motion in the generated outputs.

In our pipeline, the four calibrated cameras serve as the input views, and Diffuman4D generates images at a set of target viewpoints specified by their camera poses. The outputs consist of foreground-only RGBA images stored in the images alpha directory, together with a transforms.json file containing the camera intrinsic and extrinsic parameters for all generated views. These outputs form the multi-view dataset used for the subsequent 4DGS reconstruction stage.

Data Format Adaptation

The outputs of Diffuman4D are reorganized into the format required by the dnerf branch of 4D Gaussian Splatting [34]. This includes restructuring the images and camera parameters into the expected directory layout, converting the camera poses into the dnerf JSON format with normalized timestamps, normalizing the scene scale, and generating a random point cloud as the initial Gaussian initialization.

Training Configuration

We train the 4DGS model using the dnerf branch of the official 4D Gaussian Splatting implementation with default parameters. Since the full dataset consists of a large number of frames, directly loading all images into memory exceeds GPU memory capacity. To address this, we modify the data loading pipeline, by reading images from disk on demand during training rather than being preloaded in full. Training is performed using the PyTorch DataLoader path enabled via the dataloader=True flag, which avoids Out of Memory (OOM) issues because of the full dataset preloading.

Gaussian Splatting Reconstruction

The frames from both real and synthetic data, as well as the camera transforms are given to vanilla 4DGS as input. The reconstruction is trained with the dnerf settings and default parameters.

Post Processing

The result from 4DGS is a single point cloud .ply file and deformation information in multiple .pth files. However, our viewer takes per timestamp splat files. We use the "export per frame 3DGS" script provided in 4DGS repository to extract per frame ply files, then reformat them into the smaller scale splat format.

VR Visualization

The splat files are imported into a unity project, where the customized viewer plays the per frame splats in sequence.

9 Discussion

This section revisits the three research questions defined at the beginning of the thesis and discusses how they were addressed through the final pipeline and the experimental process.

9.1 Research Questions Revisited

RQ 1: How can a stable 4D Gaussian representation be reconstructed from sparse-view camera input in dynamic scenes?

The experiments show that a stable 4D Gaussian representation cannot be reliably reconstructed from only four fixed sparse cameras using the original 4DGS pipeline alone. The sparse camera setting does not provide enough overlapping observations for COLMAP-based preprocessing or for stable novel-view reconstruction. Geometry-based constraints, such as depth loss and cross-view consistency, can add useful supervision, but they are not sufficient to recover unseen parts of the scene.

In our final pipeline, the problem is addressed by combining calibrated camera poses with generative completion. The four real cameras are first calibrated to obtain accurate intrinsic and extrinsic parameters. Foreground masks are then used to focus the reconstruction on the target human object. Diffuman4D is used to generate additional novel-view videos from the sparse input views, creating a denser multi-view dataset. This generated dataset is then converted into the required 4DGS format and used to train the final 4D Gaussian representation.

Therefore, the stable reconstruction is not achieved by directly applying 4DGS to the four real cameras. Instead, the sparse-view input is first expanded into a denser view setting through generative completion. This makes the reconstruction more suitable for free-viewpoint visualization and later VR playback. However, the stability of the final 4DGS model still depends on the consistency and quality of the generated views.

RQ 2: Under a sparse fixed-camera setting, how do generative completion methods compare to geometry-based approaches in maintaining stable and consistent 4D reconstructions for dynamic objects?

The comparison shows that geometry-based approaches and generative completion methods address different parts of the sparse-view problem. Geometry-based methods try to regularize the reconstruction using additional constraints, such as depth priors or cross-view reprojection consistency. These methods can help reduce ambiguity in observed regions, but they cannot provide visual information for parts of the object that are never seen by the sparse cameras.

This limitation was visible in our geometry-based experiments. Even when depth supervision and cross-view consistency were added, the reconstruction remained fragmented and unstable under novel viewpoints. The model could partially fit some known camera views, but it still produced floating artifacts, duplicated structures, and broken geometry when rendered from unseen directions. This indicates that the main limitation was not only the lack of geometric regularization, but also the lack of view coverage.

Generative completion methods are more suitable for this setting because they directly synthesize the missing views before the 4DGS reconstruction stage. However, the tested generative methods were not equally suitable. Efficient4D was designed for single-view video input and did not provide stable human reconstruction on our data. SEVA could generate recognizable results in more restricted settings, but it processes timestamps independently and is not designed for temporally consistent dynamic human reconstruction. Diffuman4D was the most suitable method because it is designed for sparse-view human videos and includes spatio-temporal consistency and human pose guidance.

Based on these results, generative completion was more effective than geometry-based methods for our final goal. The reason is that full 360° dynamic human reconstruction requires not only geometric regularization, but also additional visual observations. For this project, generating these missing observations before 4DGS training was more practical than trying to recover them only through geometric losses.

RQ 3: Given a stable reconstruction, how well can dynamic objects be segmented directly in Gaussian space?

In the final pipeline, dynamic objects were not segmented directly in Gaussian space. The original plan was to evaluate Gaussian-space segmentation methods such as SA4D. However, after selecting Diffuman4D as the generative completion method, this step became unnecessary for the current scope. Diffuman4D is designed for human performance synthesis, and its preprocessing and generation pipeline already focuses on the foreground human subject.

In our implementation, foreground segmentation is performed before the generation and reconstruction stages. The input images are processed with foreground masks,

and the generated outputs are foreground-only RGBA images. As a result, the 4DGS model is trained mainly on the target human object rather than on a full scene containing both foreground and background. This means that object separation is handled before reconstruction, instead of being performed as post-processing in Gaussian space.

For the goal of visualizing a single dynamic human object in VR, this approach is more direct and practical. Removing the background before training prevents irrelevant scene content from becoming part of the Gaussian representation and reduces the need to label or separate Gaussians after reconstruction. However, this also means that RQ3 was not fully evaluated in its original form. This thesis does not demonstrate direct Gaussian-space segmentation of dynamic objects from a complete reconstructed scene. Instead, it shows that for a target-object reconstruction pipeline, foreground segmentation at the image preprocessing stage can replace Gaussian-space segmentation.

If the future goal is to reconstruct a complete scene with multiple dynamic or static objects, Gaussian-space segmentation methods would still be relevant. For the current scope, where the target is one dynamic human object, pre-segmentation before 4DGS reconstruction is the more suitable solution.

9.2 Limitations and Implications for VR

Although the final pipeline provides a recognizable dynamic human reconstruction that can be observed from different directions, several limitations remain when the result is used in VR. These limitations are not only related to reconstruction quality, but also affect how the user can view and interact with the reconstructed object.

The first limitation is visual clarity during immersive observation. The final reconstruction preserves the overall human shape and motion, but the object is not consistently sharp in all regions. In a normal rendered video, such blur or small artifacts may be less noticeable because the camera path is fixed. In VR, however, the user can move closer to the object, change viewing angle freely, and inspect details from arbitrary positions. This makes fuzzy surfaces, floating Gaussians, and boundary noise more visible than in a fixed 2D rendering. Therefore, the current result is more suitable for general observation and recognition than for detailed inspection of fine appearance.

Another limitation is related to dynamic playback in VR. The original 4DGS representation contains deformation information over time, but the VR viewer does not run this deformation process directly in real time. Instead, the dynamic result is exported as per-frame splat files and played sequentially in Unity. This makes visualization feasible, but it also limits temporal resolution, file size, and headset

performance. As a result, the current implementation is more suitable as a PCVR demonstration of dynamic Gaussian playback than as a fully optimized VR system.

Overall, these limitations show that the pipeline is effective for demonstrating sparse-camera dynamic human reconstruction in VR, but its current use case is mainly visual observation. More demanding VR applications, such as close-range inspection, object editing, multi-object interaction, or standalone deployment, would require further improvements in reconstruction quality and real-time rendering efficiency.

9.3 Future Work

9.3.1 Confidence-Aware Training for Generated Multi-View Data

One limitation observed in the final pipeline is that the quality of the 4DGS reconstruction is strongly affected by the temporal consistency of the generated images. Diffuman4D produces clear and usable foreground images at individual frames and viewpoints. But when the generated virtual-camera sequences are viewed as videos, small frame-to-frame inconsistencies can still be observed. Compared with the real camera sequences, where clothing, facial appearance, and body texture remain almost unchanged over time, the generated sequences always contain subtle changes in local details. For example, the apparent size of shoes, clothing texture, or small body details may change slightly between neighboring frames.

Although these inconsistencies are small in individual images, they become important when the images are used as training data for 4DGS. The reconstruction model treats the generated frames as supervision, so it has to fit slightly different appearances of the same object across time. This can lead to a final 4DGS model that is recognizable from all directions and preserves the overall human motion, but still appears fuzzy or blurry instead of sharp. This suggests that improving the reconstruction quality requires not only better generated images, but also a better way of handling generated data during 4DGS optimization.

A possible future direction is to introduce confidence-aware training for generated multi-view data. Efficient4D uses a confidence-weighted loss to reduce the influence of inconsistent generated pixels during 4D reconstruction. Its confidence maps are estimated from temporal consistency: if the current frame can be well reconstructed from neighboring frames through video interpolation or warping, the corresponding pixels are treated as more reliable; if the current frame differs strongly from the neighboring-frame prediction, the corresponding pixels receive lower confidence. This idea is relevant to our pipeline because the main artifacts in our generated data are also temporal appearance changes rather than complete failure in every frame.

However, this mechanism cannot be directly reused without modification. Effi-

cient4D is designed for a single-view video-to-4D setting, and its reconstruction branch expects a specific generated data format. Our pipeline instead uses sparse multi-view human videos, Diffuman4D-generated virtual-camera sequences and the vanilla 4DGS implementation. Therefore, future work would need to adapt both the confidence computation and the 4DGS training loss to this different setting. This includes computing confidence maps for each generated virtual camera over time, aligning them with the corresponding RGBA frames, and modifying the vanilla 4DGS loss so that inconsistent regions contribute less to the optimization. Such a confidence-aware strategy could help reduce blur and texture averaging caused by generated-frame inconsistency, while still preserving the view coverage provided by Diffuman4D.

9.3.2 Cleaner Foreground Gaussian Reconstruction and Segmentation

Another direction for future work is to obtain a cleaner Gaussian representation of the target object. In this thesis, we did not separately implement Gaussian-space segmentation. Instead, segmentation was moved to the preprocessing stage: the input and generated data were already foreground-focused, and the final 4DGS model was trained mainly on images containing only the human object. This was sufficient for the current goal of visualizing one dynamic human subject, but it does not fully solve the problem of producing a clean foreground-only Gaussian model.

This limitation became visible when comparing different 4DGS training branches. The *multipleview* branch was able to preserve clearer human features in some tests, but it was not suitable for the foreground RGBA images used in our final pipeline. When white-background images or transparent-background images were adapted for this branch, the reconstructed model still contained sparse white Gaussians around the human body. These residual Gaussians made the foreground object less clean and introduced visible noise around the target. The *dnerf* branch handled the foreground-only data more cleanly in this respect, but the reconstruction quality was less sharp than what the *multipleview* branch could sometimes preserve.

This suggests that foreground segmentation before training is helpful, but not always sufficient. Even if the 2D images contain only the target person, the 4DGS optimization can still create noisy or unnecessary Gaussians around the object boundary. Similar residual Gaussians can also be observed in example results from related generated-data 4DGS pipelines, where a sparse halo of points remains around the target object. This indicates that clean object-level Gaussian reconstruction is still an open practical issue when using generated or foreground-segmented training data.

Future work could therefore explore alpha-aware or mask-aware 4DGS training, where the foreground mask or alpha channel is used not only to remove the background image content, but also to constrain opacity and Gaussian placement during

optimization. More advanced Gaussian-space segmentation methods could also be used to separate the target object from residual background or boundary artifacts. These directions would be especially important if the final output is intended for VR, because floating or noisy Gaussians around the object become more visible when the user can observe the reconstruction from arbitrary viewpoints.

10 Conclusion

Novel View Synthesis and 3D reconstruction are constantly explored and improved by research, and will see many improvements in the upcoming years. Gaussian Splatting is one of the corner stones of this advancement, providing a baseline for many works and improving upon state of the art particularly in terms of efficiency. This thesis explored gaussian splatting particularly in dynamic scenes, focusing on its evaluation for real use case friendly sparse camera environments. This evaluation was done by testing vanilla methodology, attempting geometry based and generative solutions, and finally attempts on improving upon the quality of best results to provide a usable end to end pipeline. Final pipeline begins from the environment setup and recording stages, and describes all steps to end on a virtual reality display of results. This work focuses on the tasks of sparse view and dynamic target segmentation for Gaussian Splatting, but there are many future improvements to be made both in these areas and other aspects such as training speed or memory efficiency.

References

- [1] A. Azzarelli, N. Anantrasirichai, and D. R. Bull. *Splatography: Sparse multi-view dynamic Gaussian Splatting for filmmaking challenges*. 2025. arXiv: 2511.05152 [cs.CV]. URL: <https://arxiv.org/abs/2511.05152>.
- [2] H. K. Cheng, S. W. Oh, B. Price, A. Schwing, and J.-Y. Lee. “Tracking anything with decoupled video segmentation”. In: *ICCV*. 2023.
- [3] C. Donalek, S. G. Djorgovski, A. Cioc, A. Wang, J. Zhang, E. Lawler, S. Yeh, A. Mahabal, M. Graham, A. Drake, S. Davidoff, J. S. Norris, and G. Longo. “Immersive and collaborative data visualization using virtual reality platforms”. *2014 IEEE International Conference on Big Data (Big Data), Big Data (Big Data), 2014 IEEE International Conference on* (2014), pp. 609–614.
- [4] A. Fusiello. *Computer Vision. [Elektronisk resurs] : Three-Dimensional Reconstruction Techniques*. Springer, Cham, Switzerland, 2024.
- [5] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez. “Automatic generation and detection of highly reliable fiducial markers under occlusion”. *Pattern Recognition* **47** (2014), pp. 2280–2292. DOI: 10.1016/j.patcog.2014.01.005.
- [6] *Gaussian splatting playground in unity*. 2026. URL: <https://github.com/aras-p/UnityGaussianSplatting>.
- [7] M. A. Gigante. “1 - virtual reality: definitions, history and applications”. In: R. Earnshaw et al. (Eds.). *Virtual Reality Systems*. Academic Press, Boston, 1993, pp. 3–14. ISBN: 978-0-12-227748-1. DOI: <https://doi.org/10.1016/B978-0-12-227748-1.50009-3>. URL: <https://www.sciencedirect.com/science/article/pii/B9780122277481500093>.
- [8] S. Ji, G. Wu, J. Fang, J. Cen, T. Yi, W. Liu, Q. Tian, and X. Wang. “Segment any 4d gaussians”. In: 2024. arXiv: 2407.04504 [cs.CV]. URL: <https://arxiv.org/abs/2407.04504>.
- [9] Y. Jin, S. Peng, X. Wang, T. Xie, Z. Xu, Y. Yang, Y. Shen, H. Bao, and X. Zhou. “Diffuman4d: 4d consistent human view synthesis from sparse-view videos with spatio-temporal diffusion models”. In: *International Conference on Computer Vision (ICCV)*. 2025.
- [10] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. “3d gaussian splatting for real-time radiance field rendering”. *ACM Transactions on Graphics* **42:4** (2023). URL: <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>.

- [11] R. Khirodkar, T. Bagautdinov, J. Martinez, S. Zhaoen, A. James, P. Selednik, S. Anderson, and S. Saito. “Sapiens: foundation for human vision models”. *arXiv preprint arXiv:2408.12569* (2024).
- [12] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. “Segment anything”. *arXiv:2304.02643* (2023).
- [13] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. “Tanks and temples: benchmarking large-scale scene reconstruction”. *ACM Transactions on Graphics* **36**:4 (2017).
- [14] H. Kong, X. Yang, and X. Wang. “Generative sparse-view gaussian splatting”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2025, pp. 26745–26755.
- [15] T. Li, M. Slavcheva, M. Zollhoefer, S. Green, C. Lassner, C. Kim, T. Schmidt, S. Lovegrove, M. Goesele, R. Newcombe, and Z. Lv. *Neural 3D Video Synthesis from Multi-view Video*. 2022. arXiv: 2103.02597 [cs.CV]. URL: <https://arxiv.org/abs/2103.02597>.
- [16] Y. Li, J. Cao, P. Ruan, D. Saxena, S. Zhu, and Y. Cao. *Geometry-Consistent 4D Gaussian Splatting for Sparse-Input Dynamic View Synthesis*. 2025. arXiv: 2511.23044 [cs.CV]. URL: <https://arxiv.org/abs/2511.23044>.
- [17] Q. Liu, Y. Liu, J. Wang, X. Lyv, P. Wang, W. Wang, and J. Hou. *MoDGS: Dynamic Gaussian Splatting from Casually-captured Monocular Videos with Depth Priors*. 2025. arXiv: 2406.00434 [cs.CV]. URL: <https://arxiv.org/abs/2406.00434>.
- [18] Y. Liu, C. Lin, Z. Zeng, X. Long, L. Liu, T. Komura, and W. Wang. “Syncdreamer: generating multiview-consistent images from a single-view image”. *arXiv preprint arXiv:2309.03453* (2023).
- [19] Z. Luo, H. Ran, and L. Lu. *Instant4D: 4D Gaussian Splatting in Minutes*. 2025. arXiv: 2510.01119 [cs.CV]. URL: <https://arxiv.org/abs/2510.01119>.
- [20] B. Lyu and Y. Wang. “Immersive visualization of 3d subsurface ground model developed from sparse boreholes using virtual reality (vr)”. *Underground Space* **17** (2024), p. 188.
- [21] Meta. *Meta quest 3*. Accessed: 2026-05-16. 2026. URL: <https://www.meta.com/en/quest/quest-3/>.
- [22] Meta. *Meta spatial sdk*. Accessed: 2026-05-22. 2026. URL: <https://developers.meta.com/horizon/develop/spatial-sdk/>.
- [23] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. “Nerf: representing scenes as neural radiance fields for view synthesis”. In: *ECCV*. 2020.
- [24] T. Müller, A. Evans, C. Schied, and A. Keller. “Instant neural graphics primitives with a multiresolution hash encoding”. *ACM Trans. Graph.* **41**:4 (2022), 102:1–102:15. DOI: 10.1145/3528223.3530127. URL: <https://doi.org/10.1145/3528223.3530127>.
- [25] U. Nations. *The 17 sustainable development goals*. Retrieved March 4, 2026. 2015. URL: <https://sdgs.un.org/goals>.
- [26] OpenCV. *Charuco detection*. Accessed: 2026-05-16. 2026. URL: https://docs.opencv.org/3.4/df/d4a/tutorial_charuco_detection.html.

- [27] Z. Pan, Z. Yang, X. Zhu, and L. Zhang. “Efficient4d: fast dynamic 3d object generation from a single-view video”. *International Journal of Computer Vision* (2026).
- [28] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. “D-nerf: neural radiance fields for dynamic scenes”. *arXiv preprint arXiv:2011.13961* (2020).
- [29] F. Rameau, J. Park, O. Bailo, and I. S. Kweon. “Mc-calib: a generic and robust calibration toolbox for multi-camera systems”. *Computer Vision and Image Understanding* (2022), p. 103353. ISSN: 1077-3142. DOI: <https://doi.org/10.1016/j.cviu.2021.103353>. URL: <https://www.sciencedirect.com/science/article/pii/S1077314221001818>.
- [30] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer. “Sam 2: segment anything in images and videos”. *arXiv preprint arXiv:2408.00714* (2024). URL: <https://arxiv.org/abs/2408.00714>.
- [31] J. L. Schönberger and J.-M. Frahm. “Structure-from-motion revisited”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [32] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. “Pixelwise view selection for unstructured multi-view stereo”. In: *European Conference on Computer Vision (ECCV)*. 2016.
- [33] *Splatvfx*. 2026. URL: <https://github.com/keijiro/SplatVFX>.
- [34] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang. “4d gaussian splatting for real-time dynamic scene rendering”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2024, pp. 20310–20320.
- [35] A. Yu, S. Fridovich-Keil, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. “Plenoxels: radiance fields without neural networks”. *CoRR* **abs/2112.05131** (2021). arXiv: 2112.05131. URL: <https://arxiv.org/abs/2112.05131>.
- [36] J. Zhou, H. Gao, V. Voleti, A. Vasishta, C.-H. Yao, M. Boss, P. Torr, C. Rupprecht, and V. Jampani. “Stable virtual camera: generative view synthesis with diffusion models”. *arXiv preprint arXiv:2503.14489* (2025).
- [37] J. Zhou, Z. Yang, L. Han, W. Zhang, K. Shi, S. Xu, and Y.-S. Liu. “4c4d: 4 camera 4d gaussian splatting”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2026.