POLITECNICO DI TORINO

MASTER's Degree in CINEMA AND MEDIA ENGINEERING

VOLUCAP

MASTER's Degree Thesis

From Meshes to Splats: Exploring Dynamic Gaussian Splatting for Human Avatars in Volumetric Capture Workflows

Supervisors

Prof. Andrea BOTTINO

Prof. Francesco STRADA

Mr. Sven BLIEDUNG VON DER HEIDE

Candidate

Arianna FERRARIS

APRIL 2025

Abstract

Volumetric capture has revolutionized immersive media by reconstructing virtual scenes from real-world footage, captured through multi-camera setups. This method preserves authentic movements and textures without relying on computer-generated elements. While mesh-based representations have dominated volumetric pipelines, their limitations in handling complex scenarios restrict creative freedom and impose strict capture guidelines. Gaussian Splatting introduces a novel approach for real-time radiance field rendering, using 3D Gaussians to represent scenes. The core purpose of this thesis is to investigate Dynamic Gaussian Splatting as a potential alternative to meshes in *Volucap GmbH*'s volumetric capture workflow, focusing on human avatars rendering for extended reality (XR) experiences.

This research explores two primary implementation approaches for Dynamic Gaussian Splatting: frame-by-frame Gaussian Splatting and 4D Gaussian Splatting. The advantages and challenges of both these methods are analyzed to determine the most suitable approach for integration into the company's existing workflow.

This study then delves into a comparison between mesh-based and Gaussian-based outputs both across standard scenarios, where meshes generally perform well, and complex cases that challenge traditional pipelines, such as transparencies, thin fabrics and objects, dark clothing, uniform textures, reflective surfaces and hair reconstruction. The evaluation integrates quantitative and qualitative analysis, assessing Gaussian Splatting's ability to overcome mesh limitations while maintaining *Volucap*'s production standards. The findings will help determine the suitability of this technique for a potential integration into *Volucap*'s workflow.

Table of Contents

Chapter 1

Introduction

1.1 Context and Motivation

In recent years, the video entertainment industry has undergone significant transformations. Viewers increasingly find traditional 2D video insufficient, drifting toward more immersive and interactive technologies that are rapidly evolving and reshaping the industry. As demand for deeper engagement grows, new formats are emerging, offering enhanced realism and interactivity.

The first step toward immersion beyond 2D video was the 360° video (also called omnidirectional video), where a scene is recorded in all directions, allowing users to experience a complete 360° view.[1] This is made possible by using either a single omnidirectional camera or multiple cameras positioned in a circular array to capture the environment from every angle.
Omnidirectional videos provide three degrees of freedom (3 DoF), meaning the viewer can rotate their head along the pitch, yaw, and roll axes.[2] This allows them to freely choose their viewing direction, creating a more immersive and dynamic viewing experience. However, a major limitation of 360° video is the lack of physical movement: the viewer cannot move within the scene or interact with objects, reducing the sense of true immersion. This led to further advancements in the quest for a fully immersive viewing experience.

Volumetric capture is a technique that records a dynamic scene from multiple angles over time using synchronized cameras and depth sensors to capture an environment, a person, or an object. Instead of producing a flat 2D image, this method generates a full volumetric representation of the subject, allowing for true three-dimensional viewing. Through post-processing, this raw volumetric data is converted into a volumetric video, which can be viewed from any angle, maintaining realistic depth, color, and lighting.[3]
Unlike 360° video, volumetric videos offer six degrees of freedom (6 DoF): the same three rotational movements as 360° video (pitch, yaw, and roll) plus three translational movements along the X, Y, and Z axes.[1] This means that, beyond simply

changing their viewing direction, users can also physically move through the space, enhancing their sense of presence and immersion.

Since this technique constructs the virtual environment using real-world video footage, captured directly by advanced cameras or multi-camera setups, the virtual world becomes a true reflection of real-world events and physical actions.[2] Unlike computer-generated assets, which rely on digitally modeled objects and characters, volumetric video directly captures reality, preserving authentic movement, natural textures, and realistic lighting conditions.
This approach results in an unparalleled "reality illusion," offering a level of authenticity and presence that traditional computer graphics cannot easily achieve.

Volumetric videos can be viewed on a variety of devices, including desktops, mobile devices, and AR/VR platforms. However, the most immersive experience is achieved through Head-Mounted Displays (HMDs).
An HMD is a wearable display device that provides a fully immersive viewing experience, surpassing traditional flat screens. These devices use a built-in Inertial Measurement Unit (IMU) to track both head position and rotation, dynamically adjusting the displayed video to match the user's movements within the virtual environment.[2]
By allowing users to look around and move freely inside the scene, HMDs create a heightened sense of presence. This level of spatial immersion enables users to feel as though they are truly inside the recorded environment, delivering a deeply engaging and interactive experience.

As the demand for immersive and interactive content grows, companies specializing in volumetric capture have become crucial in advancing this technology. In this rapidly evolving landscape, *Volucap* has distinguished itself as a pioneering force within the volumetric capture industry.

Volucap was founded in 2018 and has since established itself as one of the world leading companies in the volumetric capture industry, renowned for its unparalleled quality and resolution in immersive content creation. *Volucap* not only produces content for XR applications, but is also deeply involved in cinematic volumetric capture, providing its cutting-edge technology to major feature film productions such as *Mickey 17* and *Matrix Resurrections*.
Furthermore, *Volucap*'s expertise extends to innovative work in the field of deepfakes, positioning the company at the forefront of technological advancements in digital human representation.

Volucap's core product consists of volumetrically captured human avatars, which are used for visualization, VFX, VR, AR, and various other applications, offering high-resolution meshes and texture sequences. The ability to create these exception-

ally realistic human avatars is made possible by *Volucap*'s state-of-the-art studio located in *Studio Babelsberg*'s FX Center. This high-tech facility is equipped with last generation cameras and a unique lighting system, enabling the high-fidelity volumetric capture of people and objects with remarkable accuracy and realism.

A crucial step in *Volucap*'s pipeline for creating human avatars is the generation and subsequent texturing of 3D meshes. Traditionally, meshes have been the dominant method for representing 3D scenes in volumetric capture, primarily due to their balance between simplicity and expressive power. They can, in fact, approximate complex geometries with high precision by adjusting the density of vertices and faces, offering flexible levels of detail. Moreover, their compatibility with modern graphics hardware makes them highly efficient for real-time rendering and simulation, ensuring smooth performance in interactive applications.[4]

However, through years of experience, the *Volucap* team has identified several limitations of meshes, particularly in challenging scenarios. Some of the most problematic cases include:

- Translucent and reflective objects (such as glass, polished metals, or glossy fabrics);
- Thin objects and fine clothing details that do not provide sufficient depth information;
- Extremely dark objects, which suffer from low contrast in depth estimation;
- Highly uniform textures, where the absence of distinguishable feature points results in poor scene reconstruction;
- Hair, difficult to capture due to its intricate geometry and fine texture details.

These limitations impact both *Volucap*'s workflow and client projects. To ensure optimal results, clients must strictly adhere to wardrobe guidelines given by the *Volucap* team, often limiting their creative freedom. Additionally, wardrobe tests must be conducted several days before a shoot, requiring additional time and resources without direct compensation.
This further motivates the exploration of alternative techniques that can overcome these constraints while maintaining high-quality visual output.

To address these challenges, alternative representations have emerged, one of the most promising being point-based techniques. In this category, a groundbreaking new approach has regained recognition as a promising solution: Gaussian Splatting.

Gaussian Splatting is a novel approach for real-time radiance field rendering that represents scenes using 3D Gaussians. This method enables state-of-the-art visual quality while maintaining competitive training times, allowing for high-quality

novel view synthesis at real-time frame rates. Scene geometry is modeled using a set of anisotropic 3D Gaussians, characterized by 3D position, opacity, anisotropic covariance, and spherical harmonics coefficients, making it particularly efficient for real-time rendering applications.
One of the key advantages of 3D Gaussians is that they serve as a differentiable volumetric representation, but they can also be rasterized very efficiently by projecting them into 2D space and applying α-blending. This makes Gaussian Splatting an attractive alternative to mesh-based scene representations.[5]

While 3D Gaussians are highly effective for static scene representation, extending them to dynamic scenes introduces additional challenges. This research explores the potential of Dynamic Gaussian Splatting as a way to overcome the limitations of mesh-based reconstruction, investigating how it can be integrated into a volumetric capture workflow to improve the representation of dynamic human avatars.

As *Volucap* continues to push the boundaries of volumetric capture, the team remains committed to exploring cutting-edge technologies that address the limitations of traditional mesh-based approaches. Gaussian Splatting, and more specifically Dynamic Gaussian Splatting, represents a promising alternative that could be capable of expanding creative possibilities.
By integrating these advanced techniques, *Volucap* aims to redefine the standards of volumetric capture, ensuring unparalleled quality for the next generation of VR, AR, and cinematic applications.

1.2 Purpose of the study and *Volucap*'s requirements

The core purpose of this thesis is to investigate the feasibility and benefits of integrating Dynamic Gaussian Splatting into *Volucap*'s current workflow. In particular, this research aims to evaluate how well Gaussian Splatting functions as a dynamic scene rendering technique for human avatars, intended for extended reality (XR) experiences.

By researching a new rendering technique that could replace meshes, this work seeks to address longstanding issues that *Volucap* has encountered since establishing its volumetric capture studio in 2018. Despite continuous technological advancements in both hardware and software, meshes remain fundamentally constrained, failing to accurately reproduce specific challenging scenarios that significantly impact the final output quality. These limitations have led the *Volucap* team to actively search for alternative solutions capable of resolving these problems.
This thesis proposes that a Gaussian Splatting-based workflow could offer a viable alternative, integrating this innovative radiance field rendering approach into *Volucap*'s well-established pipeline.

The case study considered in this research is the representation of human avatars, which will later be used as primary and secondary characters in extended reality experiences. This decision is driven by the fact that the majority of *Volucap*'s productions are aimed at the VR and AR market, where the company is trusted to create high-quality, photorealistic human avatars optimized for HMD visualization.

Furthermore, this thesis aims to examine the challenges of merging state-of-the-art volumetric capture systems with cutting-edge Gaussian Splatting technology, evaluating the compatibility of Gaussian-based scene representations with high-end volumetric capture data.

Like other leading volumetric capture companies, *Volucap* relies on a studio and processing pipeline tailored for mesh-based rendering. Over time, the *Volucap* team has developed custom processing tools, all of which are designed around the use of meshes. Introducing an entirely different rendering technique into this workflow presents a significant challenge, requiring adjustments to the input data, obtained from the cameras and early pipeline stages, to be compatible with Gaussian Splatting's training requirements.

To develop solutions that best fit *Volucap*'s needs, I was given a set of requirements that shaped my research and development process.
The first requirement was that all tests and results had to be derived from real footage captured at *Volucap*'s studio. Since the ultimate objective is to integrate Gaussian Splatting into *Volucap*'s pipeline, testing the workflow on external or synthetic data would have been irrelevant. Therefore, this research was conducted exclusively using volumetric capture data obtained from past *Volucap* productions, along with test footage captured specifically for this study.

A second constraint was the use of *Volucap*'s proprietary camera calibration system. During each shooting session, the *Volucap* team employs a custom calibration method, specifically developed for the company's volumetric capture studio, ensuring optimal geometric accuracy and alignment. To generate Gaussian Splatting outputs, I was required to use *Volucap*'s calibration system rather than relying on automated camera calibration software such as COLMAP, which is commonly used in radiance field rendering workflows. Since most state-of-the-art radiance field techniques depend on COLMAP for camera parameter estimation, this constraint necessitated bypassing COLMAP's calibration step and instead importing externally calibrated camera data. This proved to be a major challenge, as not all Gaussian Splatting implementations support external camera calibration, leading to incompatibility issues depending on the specific software or workflow used.

Another significant challenge involved processing Gaussian Splatting representa-

tions using high-resolution input images. Since *Volucap*'s human avatars are designed for extended reality experiences, maintaining high visual fidelity is essential, requiring the use of high-resolution images during training. However, working with large-scale, high-resolution inputs introduces severe computational challenges, causing longer processing times and increased memory consumption.

Considering all of these limitations, this research aims to determine whether Gaussian Splatting can provide a viable alternative to mesh-based rendering in *Volucap*'s workflow. By addressing the challenges of integration and evaluating its potential benefits, this study seeks to assess whether Gaussian Splatting can enhance volumetric capture while maintaining the high visual standards required for XR applications.

Chapter 2

State of the Art

2.1 Volumetric Capture

In this chapter, *Volucap*'s workflow is analyzed in detail, serving as a reference model for a comprehensive study on volumetric capture pipelines. As one of the world's leading companies in volumetric capture production, *Volucap*'s workflow serves as an excellent reference for analysis, both in terms of capture technology and processing pipeline, given its ability to produce some of the highest-quality volumetric outputs available on the market.

Volucap GmbH was founded as a joint venture in June 2018 by Fraunhofer HHI, in collaboration with *Studio Babelsberg*, *ARRI*, *UFA*, and *Interlake*. Shortly after, a commercial volumetric studio was established on the film campus of *Studio Babelsberg*, with commercial production beginning later that year following an initial testing phase. The core technology behind volumetric video production is 3D Human Body Reconstruction (3DHBR), originally developed by *Fraunhofer HHI* and later refined by *Volucap*'s team to meet the company's specific requirements.[6]

This chapter presents the current state of *Volucap*'s technology, providing an overview of the studio's hardware and an in-depth breakdown of each step in the workflow. The processing pipeline for capturing and producing volumetric video has undergone continuous improvements, incorporating new processing modules and workflow optimizations over time. As a result, *Volucap*'s volumetric capture studio has evolved significantly, achieving a highly advanced configuration that enables the production of some of the highest-quality volumetric video worldwide.

2.1.1 Studio

The system is composed of 42 custom-built cameras, arranged in stereo pairs and evenly distributed within a metal truss structure forming a cylinder measuring 6 meters in diameter and 4 meters in height. This setup allows for the full 360° capture of volumetric video. Each camera features a 9K resolution sensor with 65 Megapixels, ensuring ultra-high-fidelity recordings. Additionally, the cameras are equipped

with global shutters, a crucial feature for capturing fast motion sequences without distortion. Unlike rolling shutters, a global shutter exposes all pixels simultaneously, effectively freezing moving objects in place. This eliminates motion artifacts, making it particularly suitable for dynamic subjects and rapid movement sequences. For this reason, the use of global shutter cameras is essential to maintaining high image quality.[7]
The combined ultra-high-resolution video output from all cameras results in an enormous data volume, reaching approximately 4TB per minute when recording in 9K resolution.
The system completely relies on a vision-based stereo approach for multi view 3D reconstruction and does not require separate 3D sensors.[6]

Figure 2.1: *Volucap*'s volumetric capture studio

All cameras undergo an initial color correction and adaptation process, ensuring consistent and uniform imagery across the entire multi-view camera system. This step significantly impacts stereo depth estimation and, more importantly, enhances overall texture quality during the final texturing process of the 3D object.[8]
A flat-field correction is applied to each camera to compensate for image non-uniformities caused by variations in sensor pixel sensitivity, illumination inconsistencies, and lens imperfections such as vignetting.[9]
Additionally, white and black balance corrections are applied, along with a standard gamma LUT, ensuring harmonized imagery across all cameras in the multi-view system.

The lighting system consists of twelve *Arri SkyPanels*, two *Arri Orbiters*, and two *Nanlux Evoke 1200s*. The studio is primarily illuminated with diffused lighting from the ceiling, where six *SkyPanels* provide even lighting from all directions. An *Arri Orbiter*, positioned overhead, simulates sunlight from above, adding a direct

light source.
A three-point lighting setup is then used to illuminate the subject, with three lights serving as the main, fill, and back lights. For this purpose, the *Nanlux Evoke 1200s* and *Arri Orbiters* are utilized. This setup ensures a uniformly lit textured mesh with minimal internal shadows. The diffused lighting further provides optimal conditions for re-lighting 3D models during later design phases, such as in VR experiences.[6]
Additionally, all lights are flickering and synchronized with the camera shutter speed, preventing visible light flickering during capture.

2.1.2 Shooting pipeline

Shooting days at *Volucap* vary significantly, as each production differs based on several factors, including the type of content being captured, the number of clients and cast/crew present and the number of required shots.
A typical shooting day begins with booting up the system and preparing it for capture. The lights, cameras and recorders are turned on, along with the timecode system and the synchronization setup that ensures the cameras' flickering is aligned with the lights' flickering. Once everything is operational, a calibration and clean plate recording is conducted. This is a critical step before filming begins, as it ensures that all cameras are precisely calibrated and that their exact positions within the studio space are accurately recorded.

Camera calibration is a process that estimates lens and image sensor parameters to accurately model optical characteristics. These parameters are essential for correcting lens distortion, measuring object dimensions in real-world units and determining the camera's position within the volumetric capture studio. The key parameters estimated during calibration include intrinsic and extrinsic parameters as well as distortion coefficients.
To compute these parameters, a set of 3D world points and their corresponding 2D image points is required. These correspondences are typically obtained by capturing multiple images of a calibration pattern, such as a checkerboard, allowing for precise camera parameter estimation and improved accuracy in 3D scene reconstruction.[10]

The camera calibration algorithm calculates the camera matrix by estimating both extrinsic and intrinsic parameters.
Extrinsic parameters (rotation R and translation t) define the rigid transformation that maps the 3D world coordinate system to the camera's 3D coordinate system, specifying the camera's position and orientation in the scene. The optical center serves as the origin of the camera's coordinate system, with the x- and y-axes defining the image plane.
Intrinsic parameters include the focal length, the optical center (principal point), and the skew coefficient and define the projective transformation from the camera's 3D coordinates to 2D image coordinates, detailing the internal features of the camera.[10]

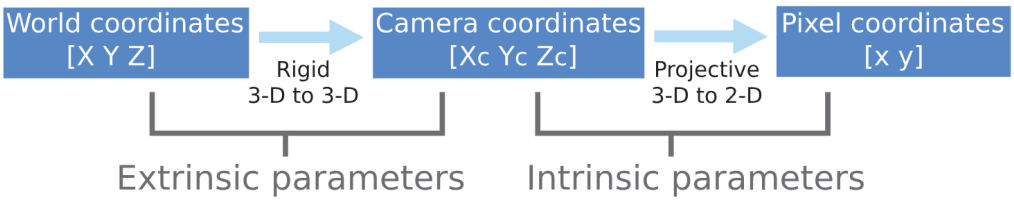

Figure 2.2: Camera calibration process
Source: [10]

Determining camera parameters involves processing the captured images that include the calibration checkerboard. This process is time-consuming and is typically performed by the team at a later stage, resulting in an XML file containing all camera parameters. Therefore, during a standard shooting day, only the calibration checkerboard is captured, postponing the parameter estimation to subsequent processing sessions.

Once camera calibration recording is complete, a clean plate capture is performed. Clean plates are recordings of an empty studio, used to develop a statistical background model for each camera. This model is fundamental for the standard segmentation mode for foreground-background separation.[11]
In the current *Volucap* pipeline, this separation is primarily achieved using a custom AI-based tool that detects subjects against the background and generates a segmentation mask. Clean plates still remain useful, especially for removing external objects present during the shoot, such as chairs or props that need to be erased in post-processing.
After these preliminary steps, the actual shooting process begins.

During a shoot, two teams work closely together to ensure a smooth process. The first is the *Volucap* team, responsible for the technical execution of the shoot. Their tasks include monitoring and operating the cameras, ensuring proper audio recording, adjusting clothing and props for optimal capture, updating the shot list for post-production, and coordinating with clients to ensure their needs are met. The second team is the client's team, which typically consists of a director, producer, hair and makeup artists, and other professionals depending on the project, along with the actors. Seamless collaboration between these two teams is essential for a successful shoot and high-quality final output.

Once the shooting begins, the volumetric capture studio (also called rotunda) is completely sealed off from the outside. To allow the client's team to monitor the shoot in real-time, two large screens placed outside the rotunda display live feeds from the cameras, enabling the director and crew to observe the performance and provide live feedback to the actors.

For the actors, performing in a volumetric capture environment is fairly different from a traditional on-set experience. They are often alone inside the rotunda, unless the scene requires multiple actors. Additionally, being surrounded by 42 cameras from all angles can be somewhat disorienting. To mitigate this, one of the cameras has been designated as the "main camera", serving as a fixed reference point to help actors maintain their orientation. Despite being physically separated from the actors, the director guides the performance remotely by watching the live feed on the monitors and providing real-time instructions via a microphone system that feeds directly into the rotunda.
When the director calls "Action", the camera operator starts the recording and monitors the process to ensure everything runs smoothly. All 42 cameras record simultaneously, remaining perfectly synchronized through the timecode system. Meanwhile, the camera operator updates the shot list, logging take details such as the shot name, type (standard take, calibration, or clean plate), actor, director's notes, and technical remarks.

Over the course of the day, one or two additional camera calibrations may be performed, depending on the length of the shoot. Since actors move around inside the rotunda, slight shifts in the rig structure may occur, which could impact the accuracy of the initial calibration. Performing multiple calibrations throughout the day helps to ensure precise data capture.

Once filming is complete, all recorded takes are exported onto *Volucap*'s network storage system. After the export is finalized, post-processing can begin.

2.1.3 Post-production pipeline

Each step of the pipeline is executed by a dedicated script. This paragraph will provide a detailed breakdown of their functionality, offering insight into how *Volucap* leverages its custom-built tools to produce state-of-the-art dynamic 3D human avatars.

The first step of the pipeline is Data Conversion. During filming, the cameras capture scenes in a raw data format, which results in extremely large files, hard to handle. Therefore, a conversion is applied to each frame of every take, including camera calibrations and clean plates, compressing the raw data into JPG format. This conversion significantly reduces file size, making the data more manageable for subsequent pipeline stages while ensuring compatibility with the libraries and tools used in the following steps.

Once Data Conversion is complete, the Masking step begins. This crucial process handles foreground-background separation for each image, ensuring that only the main subject, typically an actor, is included in the binary mask, while the studio background is entirely removed. Masking serves a dual purpose: it eliminates unnec-

essary elements from the captured images and significantly reduces the amount of data that must be processed in the following pipeline stages.
The masking tool, developed by the *Volucap* team, is AI-based and trained to identify the background from the foreground by recognizing human figures within the frame. For optimal performance, the AI relies on clear edges and strong contrasts, which help define the subject's silhouette.

Figure 2.3: Original capture

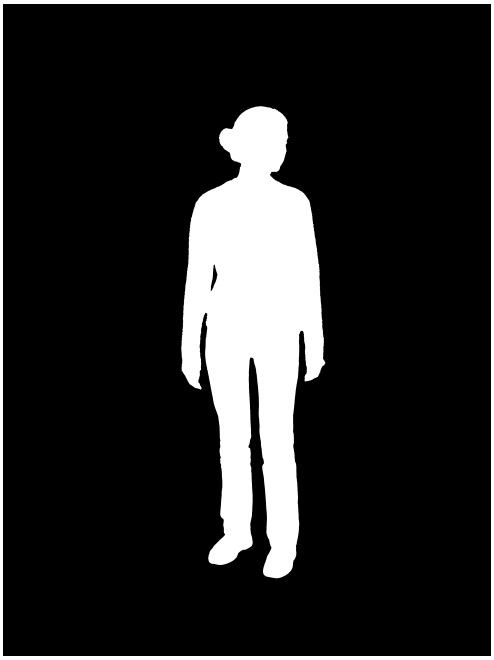

Figure 2.4: Mask

Before the implementation of this AI-driven approach, foreground-background separation was achieved through background subtraction (difference keying), where the clean plate was essential for accurately isolating the actor from the scene.

The next steps in the process are Rigging and Mask Filtering, aimed at generating a 3D skeleton that accurately represents the actor's pose in each frame. This skeleton is then used for mask filtering, identifying masks with missing body parts that should be discarded.
The process begins with the creation of 2D skeletons for each input image, simulating the actor's pose from every available camera angle. These individual skeletons are then triangulated to form a single 3D skeleton, ensuring that its pose is as close as possible to the actual positioning of the actor across all views.
Since the 3D skeleton is derived from multiple perspectives, it provides a more reliable reference than any single 2D view. This makes it particularly useful for evaluating the accuracy of the AI-generated masks. To verify each mask, the 3D skeleton is reprojected into image space, aligning it with the viewpoint of the corresponding camera. This allows us to determine which body parts should be visible in that specific mask based on the skeleton's positioning. If a mask is found to be missing

body parts that should be present, the Mask Filtering step removes it from the dataset, ensuring that only complete and accurate masks proceed to the next stage of the pipeline.

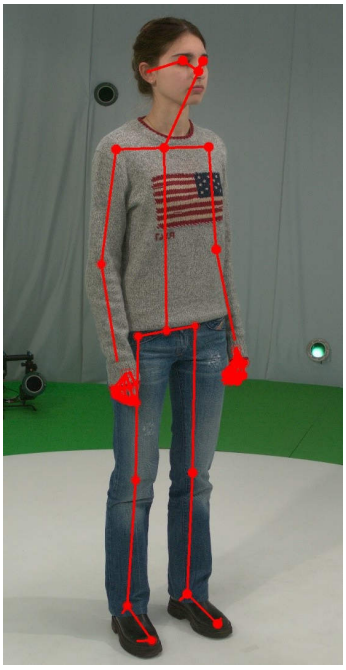

Figure 2.5: Skeleton

Rigging becomes more complex when multiple actors are captured in the same scene. Since the cameras are positioned all around them, the actors' positions within the frame continuously change depending on the viewing angle. For example, in a front-facing view, Actor A may appear on the right and Actor B on the left, whereas in a rear view, their positions are reversed, with Actor B now on the right and Actor A on the left. This constant switching can create confusion for the rigging tool, which must consistently track which skeleton belongs to which actor to ensure accurate triangulation.
To address this challenge, a technique based on image histograms was implemented. By analyzing the colors of the actors' clothing, the system can determine their relative positions in each view, helping to correctly identify and match the skeletons. This method significantly improves the reliability of the rigging process, ensuring that each actor's movements are accurately reconstructed across all perspectives.
Removing faulty masks is crucial, as it significantly reduces the need for manual corrections in later pipeline stages while also contributing to the creation of a more accurate final mesh. This step is, however, optional.

The next step focuses on generating the 3D Model using a custom-built tool that processes image files, the previously created and filtered masks, and the camera calibration XML file obtained from the calibration data processing.
The process starts with detecting feature points across the images and matching them between all available views. These corresponding points are then triangulated

into a sparse 3D point cloud using the camera calibration data, effectively converting 2D features into 3D space. The resulting 3D points provide an optimal estimate that minimizes reprojection error across views, ensuring that each point aligns as closely as possible with its corresponding 2D projections in all images.
The processing of camera calibration data, which estimates the camera parameters, combined with the generation of a sparse point cloud, represent the key steps of a technique known as Structure-from-Motion (SfM) [12].

A key subprocess of 3D Model creation is 3D Reconstruction, which follows the principles of Multi-View Stereo (MVS) [13], that reconstructs dense 3D geometry from multiple images by estimating depth maps and fusing them into a point cloud. This relies on stereo depth estimation, taking as an input the set of images and the camera parameters. In this case, the sparse point cloud is also considered: this guides the selection of stereo image pairs by identifying which images have sufficient overlapping feature points.
As described in the volumetric capture studio overview, the cameras are arranged in stereo pairs, evenly distributed within the cylindrical setup. These stereo configurations provide essential 3D information from their respective viewpoints, forming the basis for depth computation.
Initially, 2D depth maps are computed. The chosen approach employs an iterative algorithmic structure [14, 15] that compares projections of 3D patches between the left and right images of each stereo pair. This is achieved through point transfer via homography mapping, which defines the relationship between corresponding points in both images. Given that the camera system is fully calibrated, the transformation of a plane in 3D space across two images can be accurately represented using the homography matrix. This ensures precise depth estimation by leveraging the geometric relationships between cameras and their respective viewpoints.[8]

From the 2D depth maps, 3D points are generated, each containing normal information that describes surface orientation. This 3D information obtained from all stereo pairs is then merged using a visibility-driven patch-group generation algorithm [16], which eliminates occluded points to improve foreground segmentation. The algorithm applies optimized visibility rules in both the 2D image space and the 3D point cloud domain, ensuring that only the most reliable points are retained. This results in a high-resolution 3D dense point cloud, often containing tens of millions of points per frame.
For each of these points, a confidence value is assigned based on the reliability of the depth estimation algorithm. This confidence level reflects how certain the algorithm is about the accuracy of a given point's depth calculation. The confidence is higher in areas with a greater density of points, as the abundance of data allows for more precise estimations. This information plays a crucial role in the subsequent Mesh Cleaning process, ensuring that only the most accurate data contributes to the final reconstruction.

To make the data compatible with standard rendering engines, the dense point cloud needs then to be converted into a single, consistent mesh.[8]

The meshing process consists of two main steps. First, Screened Poisson Surface Reconstruction (SPSR) is applied [17], to efficiently generate a watertight mesh, significantly reducing geometric complexity while closing holes caused by occlusions or data imperfections. Next, the mesh is trimmed and cleaned based on sampling density, ensuring outliers and artifacts are removed. Finally, the surface is further simplified using Quadric Error Metrics [18], allocating more triangles to detailed areas while reducing their number in simpler regions, maintaining mesh topology and boundary integrity to enhance overall quality.[8]
After completing these steps, the final sequence of meshes is generated, optimized for both accuracy and efficiency.

Between 3D Model creation and the next step lies the only manual stage of the pipeline: Mask Fixing.
After generating the meshes, it is crucial to verify that all body parts and clothing elements of the captured subject are fully represented in each frame. In some cases, certain limbs or fabric details may be missing due to incomplete masks, leading to gaps in the final mesh. To ensure the highest quality output, this step is essential.
The process itself is straightforward: thanks to the GUI of the meshing tool, the meshes (one per frame) can be inspected from all angles, allowing for a thorough check to ensure that all expected details are present and intact. If an issue is detected in one or more frames, a manual mask correction is performed.
First, the affected masks are identified, as those where a specific body part or clothing item is absent. Once located, the missing section is manually added back to the mask, followed by a Remesh process, which updates the 3D model to incorporate the corrected areas.
Although manual intervention in this step can significantly improve the final product, the extent of required corrections varies. When scenes are captured following wardrobe guidelines, the need for manual fixes is minimal. However, in more challenging cases, mask fixing becomes a necessary step. To reduce reliance on human intervention, the Rigging and Mask Filtering step was recently introduced, helping to automate part of this process and streamline the workflow.

The next step in the process is Mesh Cleaning, where imperfections in the meshes are corrected to create smoother surfaces and remove unwanted fragments. This is achieved by applying a series of filtering algorithms sequentially, with multiple iterations to refine the results.
One of the predominant cleaning steps is Smoothing, which leverages the confidence values calculated during depth estimation to determine the level of smoothing required in different areas. Regions with low confidence values indicate less accurate depth estimation, often resulting in noticeable flickering in the meshes. To mitigate this, the

smoothing process is applied more aggressively in these areas, while high-confidence regions, where flickering is minimal or absent, receive a lighter treatment.
Another essential part of the cleaning process is Removal of Small Parts. This step evaluates the bounding box size of each independent object in the scene. If an object's bounding box falls below a certain threshold, the algorithm assumes it to be an unwanted residual, such as floor or background artifacts, and automatically removes it. However, in cases where the scene includes small props that need to be preserved, a dedicated setting can be activated to retain these objects.
Additionally, the Holes Filling algorithm plays a crucial role by identifying and filling surface gaps, addressing potential reconstruction errors caused by missing feature points.
These are just a few of the numerous operations applied during Mesh Cleaning, a fundamental step that significantly enhances the overall quality of the final 3D mesh.

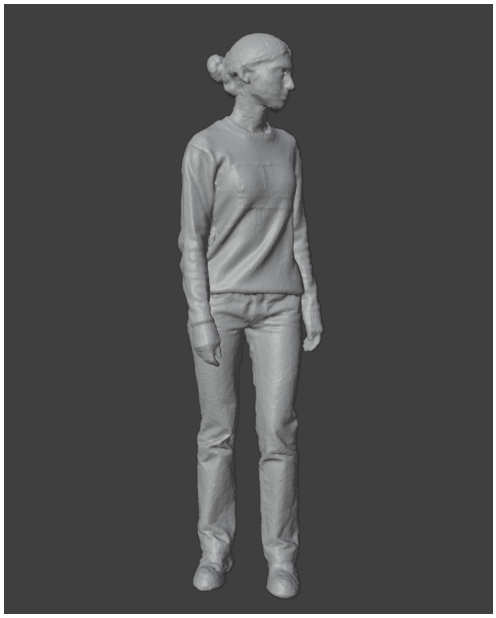

Figure 2.6: Original model

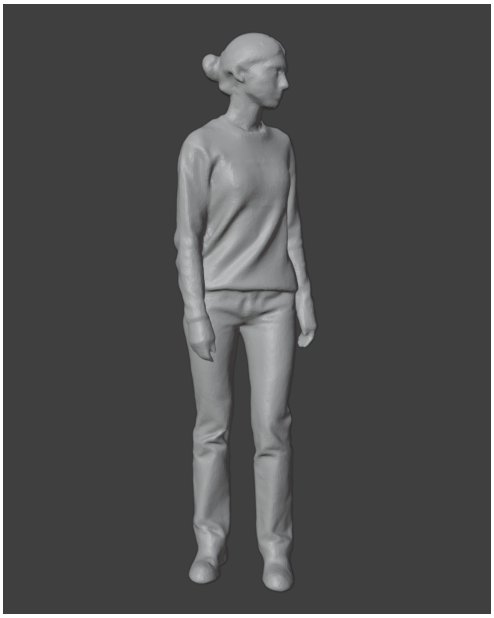

Figure 2.7: Cleaned model

The following step in the process is Tracking, which reduces the total number of meshes in a sequence by morphing them, ensuring a more stable representation over time.
The Mesh Cleaning step produces independent, cleaned meshes per frame, each with its own individual topology, which can lead to inconsistencies when viewed as a sequence. To address this, a mesh registration process is applied, creating sequences of meshes with identical topology and enhancing temporal stability. The process begins by defining key meshes, which serve as reference points in the sequence; succeeding meshes are then calculated by reshaping the key frame to match the geometry of neighboring frames, while preserving topology and local structures. To better handle sudden changes in topology within the mesh sequence, this process is performed bi-directionally, ensuring a more stable transition between frames. Whenever the

deviation of a registered mesh reaches a predefined threshold, a new key mesh is introduced.
This step is highly beneficial, as it allows new meshes to be generated only for key frames, while for the registered meshes, only the 3D vertex positions need to be adjusted. Additionally, it plays a crucial role in the next stage of the pipeline, Unwrapping, making the overall process more efficient by maintaining the same texture atlas for meshes with the same topology.[6]

During the Unwrapping step, the texture information of adjacent triangles is grouped into a texture patch, provided that the normal vectors across all triangles remain within a defined threshold. If this threshold is exceeded, a new texture patch is generated. Once the texture patches are defined, they are arranged into a texture atlas, starting with the largest patch placed in the bottom-left corner.
Two key mechanisms are integrated into the texture atlas in order to ensure the efficient application of a block-based video coding method. First, the topology is preserved across each registered mesh sequence, meaning that the shape and position of all patches remain unchanged inside of the texture atlas, while only the content within each patch is updated. Second, the gaps between patches are interpolated, preventing excessive data rates when encoding image blocks that span across patch boundaries. In addition to improving encoding efficiency, this method also simplifies texture grading across the registered sequence of meshes, further enhancing the overall quality and stability of the final output.[6]

The next step is Texturing, where the previously generated texture atlases are applied to the 3D meshes to ensure accurate surface detail representation. These texture atlases contain all necessary color information, mapped to corresponding UV coordinates on the mesh. By assigning these UV coordinates to each vertex, the correct texture regions are projected onto the 3D model, ensuring a seamless appearance. This process allows the mesh to accurately retain its original visual characteristics, preserving both fine details and color consistency across frames.

Figure 2.8: Texture atlas

The final step of the pipeline is the Encoding, where the registered mesh sequence is compressed and multiplexed into an MP4 file using a standard mesh encoder. This process ensures that the sequence can be seamlessly integrated into external software environments, such as Unity or Unreal Engine, through dedicated plugins.
During encoding, the three elementary streams (Mesh, Texture Atlas, and Audio) are combined into a single MP4 file, making it ready for efficient transmission and storage. On the receiving end, Unity and Unreal plugins facilitate the integration of volumetric video assets into AR and VR applications. These plugins contain a built-in de-multiplexer and associated decoders, enabling real-time decoding of the mesh sequence for smooth playback and interaction within the target application.[6]

Figure 2.9: Encoded file

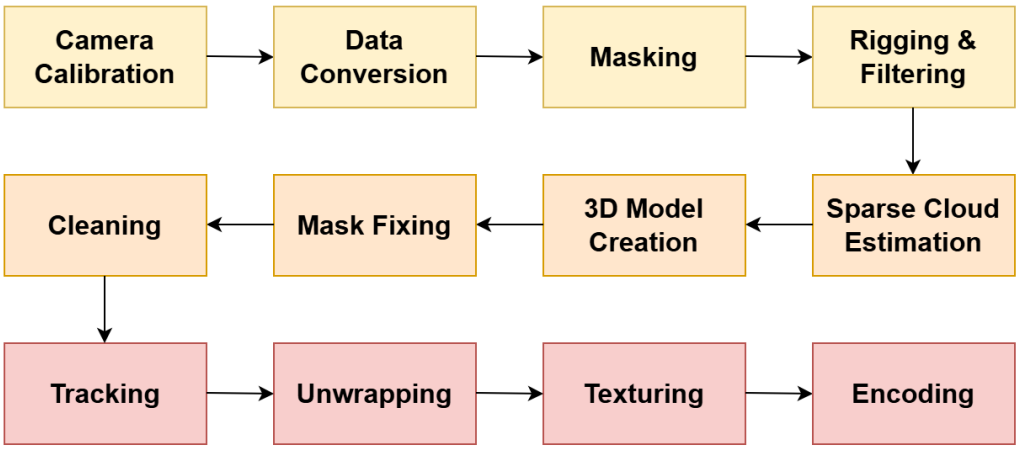

Figure 2.10: *Volucap* pipeline

2.2 Limitations of Mesh-Based Reconstruction

In order to reconstruct volumetrically captured scenes, the mesh-based approach remains the most widely used method.
As previously mentioned, meshes have traditionally been the dominant technique for representing 3D scenes in volumetric capture, primarily due to their balance between simplicity and expressive power. By adjusting the density of vertices and faces, meshes can accurately approximate complex geometries with high precision, allowing for flexible levels of detail. Additionally, their compatibility with modern graphics hardware makes them highly efficient for real-time rendering and simulation, ensuring smooth performance in interactive applications.[4]

Since its founding in 2018, *Volucap* has relied on a mesh-based approach to reconstruct volumetrically captured scenes. For the company's purposes, meshes have proven to be an effective solution, offering easy manipulation and low memory consumption, a crucial factor for integrating dynamic human avatars into XR interactive experiences. Over the years, each step of the pipeline handling meshes has been significantly optimized, reducing the need for manual corrections and improving temporal stability in dynamic reconstructions. In its current state, the mesh-based representation of dynamic scenes delivers exceptional high-quality results, meeting industry standards and enabling the creation of highly realistic human avatars. These avatars are optimized for various digital outlets, including film, virtual reality (VR), augmented reality (AR), and interactive simulations.

Despite these advantages, mesh-based reconstruction still presents several limitations, many of which remain difficult to overcome. Through years of research, experimentation, and extensive testing, the *Volucap* team has identified specific scenarios that, for various reasons, continue to pose significant reconstruction challenges.

One of the features that turns out to be extremely hard to reconstruct accurately with a mesh-based representation is hair.
Accurately capturing and modeling hair is a crucial factor in creating realistic avatars, as it significantly contributes to personal identity and individualization. However, hair reconstruction poses a major challenge due to its intricate geometry and fine texture details, which make it difficult to capture with standard volumetric methods. Additionally, the wide variation in hairstyles, both in appearance and structure, further complicates the process, making it particularly challenging to develop a single, efficient solution that can accurately reconstruct and adapt to the diverse range of hairstyles across different individuals.[19]
The *Volucap* team has tested and analyzed various hairstyles to determine which can be accurately reconstructed and which should ideally be avoided. The results show that tight hairstyles with minimal or no loose strands provide the best capture quality. This includes tight buns and braids for long hair, as well as slicked-back or

compact styles for short hair. In contrast, loose hair, regardless of texture, poses significant challenges; the problems increase when dealing with curly hair. The more intricate the geometry and finer the details, the more difficult it becomes to achieve a proper reconstruction. This challenge arises because loose strands lack clear feature points, making it difficult for depth estimation algorithms to track and reconstruct them accurately. Additionally, overlapping layers and semi-transparent regions further complicate the process, as mesh-based methods struggle to differentiate between individual strands, often leading to missing or incorrectly merged sections.

Transparent objects introduce another complex scenario where traditional mesh-based methods struggle to capture accurate geometry and appearance.
Accurately reconstructing materials like glass and crystal remains an ongoing and complex issue due to multiple factors. First, these objects do not possess their own colors but instead derive their appearance from their surroundings, making traditional color or texture-based reconstruction techniques ineffective. Second, transparent materials interact with light in intricate ways, including reflection, refraction, scattering, and absorption, making it nearly impossible to trace their full light paths. Additionally, the refractive properties of such objects depend on their refractive index, which is often unknown and varies between materials.[20]
For all these reasons, detecting feature points on translucent objects is extremely difficult, making these items nearly impossible to reconstruct accurately. As a result, the company actively avoids capturing translucent materials. The primary limitation in this regard concerns reading glasses: individuals recorded in the volumetric studio are always advised to remove them, even though these may be a defining characteristic of their appearance.

Another category of objects that proves extremely challenging to reconstruct using a mesh-based approach is thin objects. This includes not only very thin items like sheets of paper but also fine clothing details, which are often part of common outfits.
Thin objects with virtually no volume present unique difficulties in reconstruction, particularly when it comes to shape representation and the fusion of depth information. A major challenge specific to thin objects arises when they are viewed from opposite perspectives: points from different surfaces usually share the same position but have opposite normals. Measurement noise can then lead to conflicting results, causing invisible points to occlude visible ones, a phenomenon known as the problem of opposed surfaces.[21]
When considering real-world case studies, the joint reconstruction of shape and appearance for thin, translucent objects presents significant challenges in computer graphics, particularly when dealing with complex layered materials such as paper or leaves. These multi-layered translucent materials are common in everyday life, yet achieving a high-quality representation remains difficult. Their appearance is heavily influenced by non-diffusive transmittance caused by multiple light scattering

between layers, resulting in a striking appearance when back-lit and significant visual differences between the two sides when front-lit.[22]
For all these reasons, thin objects pose significant challenges for reconstruction using typical mesh-based methods. To mitigate these issues, the company deliberately avoids capturing objects that are too thin and makes efforts to tape down or conceal fine clothing details in the selected outfits.

Texture-less and dark objects or fabrics present another significant challenge for mesh-based reconstruction, as traditional methods struggle to capture their geometry accurately.
One of the key steps in 3D model creation is identifying corresponding feature points across multiple images. However, when an object lacks a recognizable texture, this process becomes difficult. If the surface is completely texture-less or very dark, feature detection algorithms struggle to identify key points, making it nearly impossible to establish accurate feature correspondences across images [23]. Since the system relies on recognizable patterns and textures to function correctly, surfaces without distinct visual details lead to incorrect or incomplete reconstructions.
To minimize these issues, *Volucap*'s capture guidelines recommend avoiding dark or texture-less materials whenever possible. However, in cases where capturing such objects or fabrics is unavoidable, a thin layer of powder is applied to the surface. This creates a subtle texture, helping the feature detection algorithm recognize matching points across multiple images and improving the reconstruction process.

The final complex scenario analyzed in this research is reflective surfaces, which, like the previous cases, pose significant challenges for mesh-based reconstruction.
Multi-view stereo methods work by first estimating depth maps from multiple viewpoints and then fusing these depth maps in a post-processing step to reconstruct the surface. A fundamental assumption of depth estimation is photometric consistency, the idea that the appearance of a point on an object remains consistent across different views. However, this assumption breaks down for glossy or reflective surfaces, as their appearance changes depending on the viewpoint. As a result, multi-view stereo methods fail to reconstruct these objects accurately.[24]
To prevent complications during capturing, the *Volucap* team advises clients to exclude reflective objects and fabrics whenever possible. In cases where their use is unavoidable, the team employs the same solution as in the previous scenario: applying a fine layer of powder to reduce reflectivity and make the surface more opaque. By minimizing reflections, this approach helps mitigate potential reconstruction issues and improves overall capture quality.

All of these complex and problematic scenarios create significant limitations for both *Volucap*'s workflow and its clients' projects, affecting both the production process and the creative flexibility of those involved.

Before each shoot, clients receive strict wardrobe guidelines designed to avoid all the problematic cases mentioned above. While necessary for ensuring a smooth capture process, these restrictions often limit creative freedom, forcing clients to make compromises on their artistic vision. Outfits must strictly adhere to specific material and color requirements, occasionally preventing them from using certain fabrics, patterns, or dark tones. Similarly, hairstyle choices are highly constrained, as styles that do not comply with reconstruction requirements must be altered or avoided altogether. Accessories such as large jewelry or reading glasses frequently have to be removed, further restricting personal styling choices. These limitations directly impact the creative flexibility available to clients, making it challenging to fully execute their original vision.

Additionally, wardrobe tests must be conducted several days before the actual shoot, requiring the entire system to be activated, along with a new calibration and clean plate capture. Each outfit must be individually tested, and once recorded, the resulting data needs to be processed and analyzed, consuming valuable time and computational resources. This process ties up workstations and personnel, leading to an increased operational workload. Despite the effort and resources involved, clients only pay for the official shooting day, meaning that wardrobe tests are an uncompensated expense for *Volucap*. This results in considerable financial and time losses, further adding to the inefficiencies caused by these reconstruction challenges.

Ultimately, these challenges not only increase the complexity of pre-production planning but also restrict artistic choices, making it difficult to fully realize certain creative visions within the constraints of volumetric capture technology.

These ongoing limitations highlight the need for alternative methods that can overcome these constraints while maintaining the high-quality visual output expected from *Volucap*. To address these issues, this research explores new approaches beyond traditional mesh-based representations, focusing on alternative techniques that could improve the reconstruction of challenging cases. The next section introduces Gaussian Splatting, a promising method that may provide a viable solution to these longstanding problems.

2.3 3D Gaussian Splatting

2.3.1 Related work

Traditionally, meshes and point-based representations have been the most commonly used methods for 3D scene reconstruction due to their explicit structure and efficient compatibility with GPU/CUDA-based rasterization. These representations allow for fast real-time rendering, making them a preferred choice in many applications.

Standard mesh-based scene reconstruction relies on Structure-from-Motion (SfM) [25] and Multi-View Stereo (MVS) [13], which can achieve high-quality results in many cases. However, these methods often struggle with "over-reconstruction," where MVS incorrectly generates non-existent geometry or unreconstructed regions, where parts of the scene fail to be reconstructed. This limitation arises because MVS relies on identifying corresponding features across multiple images. As a result, surfaces with low-texture regions (e.g., uniform-colored objects) or challenging material properties (e.g., reflective or transparent surfaces) often lead to incomplete, inaccurate, or entirely missing geometry.[5]

Recent advancements in Neural Radiance Fields (NeRFs) have introduced an alternative approach based on continuous volumetric scene representations, which optimize a multi-layer perceptron (MLP) for view synthesis using volumetric ray-marching.
Unlike meshes, which explicitly store geometric structures, NeRF represents a scene as a 5D function that maps spatial coordinates (x, y, z) and viewing direction (θ, ϕ) to volume density and emitted radiance. This enables photorealistic novel view synthesis by optimizing a continuous volumetric scene function using a sparse set of input images with known camera poses, obtained through camera calibration methods such as Structure-from-Motion (SfM) [25]. The model learns to assign accurate colors and densities to spatial points, effectively reconstructing complex real-world geometry and appearance.[26]
Unlike traditional multi-view reconstruction techniques, NeRF does not rely on Multi-View Stereo to generate an explicit 3D model of the scene. Instead, it learns an implicit function that directly synthesizes novel viewpoints without requiring a dense geometric reconstruction. This makes it particularly advantageous in handling view-dependent effects such as reflections and semi-transparency, which are challenging for traditional mesh-based methods.
The trade-off, however, is computational cost: achieving high visual quality with NeRF usually involves lengthy training times. Pure NeRF models use large neural networks that are costly to train and render, and even accelerated variants must trade off speed for quality. While NeRF has demonstrated outstanding rendering fidelity, its high computational demands have limited its practical application in real-time environments.
Some advanced radiance field methods refine NeRF's approach by storing scene information in structured formats like voxels [27], hash grids [28], or points [29]. These methods retrieve and interpolate stored values instead of relying entirely on a neural network, making rendering more efficient while still benefiting from a continuous scene representation. While the continuous nature of these methods helps optimization, their reliance on stochastic sampling for rendering remains computationally expensive and can result in noise.[5]

An alternative approach to address some of the challenges associated with meshes

is point-based rendering. Instead of using a connected mesh, point-based methods represent the scene as an unstructured set of points, often enriched with color or other attributes.
While this approach accurately captures the underlying data, it has notable limitations, including the presence of holes, aliasing artifacts, and a fundamentally discontinuous nature. To mitigate these issues, pioneering research in high-quality point-based rendering has introduced "splatting" techniques, which expand point primitives beyond a single pixel. This is accomplished using shapes such as circular or elliptical discs, ellipsoids, or surfels [30, 31, 32, 33].
Over time, point-based rendering has been further refined through approaches that enhance points with neural features and employ convolutional neural networks (CNNs) for rendering [34, 35]. These advancements have enabled faster and even real-time view synthesis. However, a significant drawback remains: these methods still depend on Multi-View Stereo (MVS) for initial geometry reconstruction, inheriting as a result MVS-induced artifacts. Therefore, they continue to suffer from issues such as over or under-reconstruction, particularly in challenging cases like featureless surfaces, reflective materials, or thin structures.[5]

2.3.2 Overview of 3D Gaussian Splatting

In the reference research [5], 3D Gaussians are introduced as a more flexible scene representation, enabling real-time rendering through a tile-based algorithm optimized for projecting Gaussians, while removing the dependency on MVS geometry.

3D Gaussian Splatting is an innovative technique for real-time radiance field rendering that represents scenes using 3D Gaussians. This approach integrates optimization with state-of-the-art visual quality and competitive training times, while its tile-based splatting method enables real-time rendering for 1080p resolution. The main goal of 3D Gaussian Splatting is to allow real-time rendering for scenes reconstructed from multiple images while achieving optimization times comparable to the most efficient existing methods for real-world scene representation.[5]

The input for this method [5] is a set of images that capture a static scene, along with the corresponding cameras calibrated by SfM, which produces a sparse set of points without normals. Starting from these points, the goal is to optimize a scene representation that allows high-quality novel view synthesis. Achieving this requires a primitive that inherits the features of differentiable volumetric representations while remaining both unstructured and explicitly defined. Differentiability allows the system to be continuously optimized during training, improving the quality of the final outcome; moreover, an unstructured and explicit representation ensures greater flexibility for readjustments and faster rendering, avoiding costly volumetric ray-marching (as in NeRF).
3D Gaussians are chosen as the ideal representation method, since they are differ-

entiable and can smoothly be projected to 2D splats allowing fast α-blending for rendering. From the sparse point cloud, a set of 3D Gaussians is defined; these gaussians are identified by 3D position p, anisotropic covariance, opacity α and spherical harmonics (SH) coefficients.

Due to the extreme sparsity of SfM points, estimating normals turns out to be extremely difficult. Likewise, optimizing these normals, which are prone to significant noise, would present considerable challenges.
By modeling geometry with 3D Gaussians defined by a mean position μ and full 3D covariance matrix Σ [36], it's possible to avoid noisy normal estimation and enable efficient projection into 2D splats for rendering. The 3D Gaussians are defined like this:

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

A straightforward method would be to optimize the covariance matrix Σ directly to generate 3D Gaussians that represent the radiance field. However, directly optimizing the covariance matrix is problematic, as gradient descent (used for optimization of all of the parameters) can easily produce invalid matrices. To address this, the covariance matrix Σ of a 3D Gaussian, which describes the configuration of an ellipsoid, is divided into two separate components:

- S is a scaling matrix, which controls the size of the Gaussian;
- R is a rotation matrix, parameterized using a quaternion (q) to properly represent rotational transformations.[5]

This decomposition allows for independent optimization of shape and orientation while maintaining numerical stability. The covariance matrix is then reconstructed using the equation:

$$\Sigma = RSS^T R^T$$

where S is a diagonal matrix storing the Gaussian's anisotropic scaling factors, and R ensures proper rotation alignment.
This formulation of anisotropic covariance, appropriate for optimization, enables the refinement of 3D Gaussians to adapt to various geometric structures in captured scenes, leading to a quite compact representation.[5]

During rendering, these 3D Gaussians are projected into 2D splats, where their covariance matrices are transformed using the viewing transformation matrix W. The covariance matrix Σ' [36] in camera coordinates becomes:

$$\Sigma' = JW\Sigma W^T J^T$$

where S is the Jacobian matrix of the affine transformation. This step ensures that the Gaussian is correctly represented in image space, preserving its orientation and spatial coherence.[5]

2.3.3 Optimization

The optimization process at the core of this approach [5] generates a dense set of 3D Gaussians that effectively represent the scene for free-view synthesis. This step involves refining not only the position p, opacity α, and covariance Σ of each Gaussian but also the Spherical Harmonic (SH) coefficients, which are responsible for capturing view-dependent color information. To ensure an accurate representation, these optimizations are coupled with an adaptive mechanism that dynamically adjusts the density of the Gaussians throughout the scene.

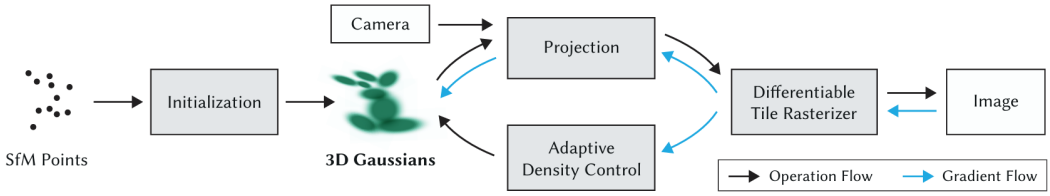

Figure 2.11: Optimization process of 3D Gaussian Splatting
Source: [5]

The process follows an iterative approach, where images are rendered and compared against the dataset's training views. Since initial geometric estimates may contain errors, the optimization must be capable of both generating new geometry and refining or eliminating inaccurate structures. The covariance parameters play a crucial role in maintaining a compact representation, as large, uniform regions can be effectively modeled with fewer but larger anisotropic Gaussians.
To facilitate this optimization, the considered implementation [5] employs Stochastic Gradient Descent (SGD) techniques, making use of GPU-accelerated frameworks and allowing for the integration of custom CUDA kernels. A key challenge in this process is efficient rasterization, as it represents the primary computational bottleneck.

Starting with the sparse point cloud produced by Structure-from-Motion (SfM), 3D Gaussian Splatting [5] adaptively adjusts the number of Gaussians and their distribution within the scene. This ensures that the initially sparse representation is refined into a denser and more accurate one. To enhance stability during the early optimization stages, the process begins with a reduced-resolution image (specifically, one-fourth of the final resolution) and progressively increases it through two upsampling steps at 250 and 500 iterations.
Once the warm-up phase concludes, the density of Gaussians is adjusted every 100 iterations. Additionally, Gaussians with α values below a predefined threshold ϵ_α are removed to improve efficiency. This adaptive control strategy focuses on both under-reconstructed and over-reconstructed regions: in the former, missing geometric details are addressed by adding Gaussians, while in the latter, excessive coverage is refined. These problematic areas are often characterized by large view-space positional gradients, indicating that the optimization process is actively working to

improve their accuracy.
To further enhance the scene representation, large Gaussians in regions with significant variance are split into smaller ones. This is done by replacing a single Gaussian with two new ones, each scaled down by an experimentally determined factor ϕ = 1.6. The new positions are sampled using the original Gaussian's Probability Density Function (PDF). On the other hand, in under-reconstructed areas where additional detail is required, Gaussians are not split but cloned. A duplicate Gaussian is generated and shifted in the direction of the positional gradient.

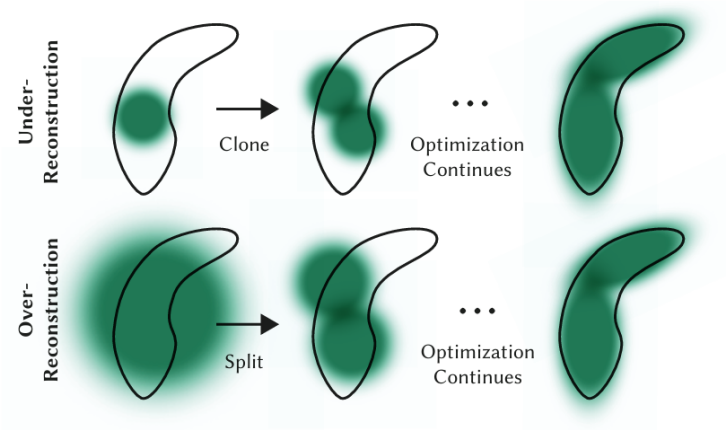

Figure 2.12: Adaptive control of Gaussians
Source: [5]

These two approaches allow for an efficient and adaptive refinement process: in the case of splitting, the total volume remains the same while increasing the number of Gaussians, whereas in cloning, both the volume and number of Gaussians expand dynamically to account for missing details.
To prevent an excessive increase in Gaussian density, particularly near the input cameras where optimization can get stuck, a regulation mechanism is applied. Every 3000 iterations, the opacity α of all Gaussians is set close to zero, allowing the optimization process to selectively increase opacity only for Gaussians that are necessary. This method helps in moderating the overall number of Gaussians while also utilizing a culling approach to remove those with an opacity value below a threshold ϵ_α. Additionally, Gaussians that grow too large in world space or develop a significant footprint in view space are periodically removed. Since Gaussians can shrink, expand, and overlap during optimization, this step ensures that unnecessary or oversized Gaussians are eliminated, helping to maintain an efficient and accurate representation.[5]

2.3.4 Rasterization

This method [5] employs a tile-based rasterizer designed for efficient sorting and α-blending, including for anisotropic splats, while avoiding previous limitations [37] on the number of splats receiving gradients. The screen is initially divided into

16×16 tiles, and 3D Gaussians are culled based on their view frustum intersection. Only Gaussians with a 99% confidence interval inside the frustum are retained, while outliers near the near plane or far outside the view are discarded using a guard band to prevent unstable covariance projections.
Each Gaussian is instantiated according to its tile coverage and assigned a sorting key based on view space depth and tile ID. The sorting is performed, based on these keys, through a fast GPU Radix sort [38], eliminating the need for per-pixel ordering. Instead, alpha-blending follows this initial sorted order, approximating true blending behavior. While this approximation can introduce minor inaccuracies, the effects become negligible when splats approach pixel size. This significantly enhances training and rendering performance without visible artifacts.
After sorting Gaussians, a list for each tile is created by identifying the first and last depth-sorted entry that splats to a given tile. For rasterization, each tile is assigned a dedicated thread block, which first loads Gaussians into shared memory and then traverses the lists front-to-back, accumulating color and opacity for each pixel. To maximize parallelism, Gaussians are processed collaboratively, and computation stops once a pixel's alpha reaches saturation ($\alpha = 1$). Threads periodically check the tile's saturation status, terminating processing once all pixels are fully covered.
Unlike previous methods [37], this approach imposes no limit on the number of splats contributing to gradients, enabling the handling of complex scenes with varying depth without requiring scene-specific hyperparameter tuning. During backpropagation, the system must recover the exact sequence of blended Gaussians per pixel. Instead of storing long lists in global memory, which would introduce dynamic memory overhead, the method reuses the sorted Gaussian array from the forward pass and traverses it back-to-front. Only points contributing to color in the forward pass are processed, optimizing efficiency. To compute gradients, rather than storing every opacity step, each point retains the final accumulated alpha, which is divided by its own alpha in the back-to-front traversal to retrieve intermediate coefficients for gradient computation. This allows accurate gradient updates while minimizing memory consumption, ensuring the method remains scalable even for high-complexity scenes.
In conclusion, this rasterizer enables efficient rendering and fast sorting, facilitating approximate α-blending (including for anisotropic splats) while eliminating previous constraints on the number of splats that can receive gradient updates.[5]

While 3D Gaussian Splatting has proven highly effective for static scene reconstruction, its application is currently limited to motionless scenarios. To address this constraint, the following two sections will explore two distinct approaches for extending Gaussian Splatting to dynamic scene reconstruction.

2.4 Frame-by-frame Gaussian Splatting

While 3D Gaussians are highly effective for representing static scenes, extending them to dynamic environments introduces additional challenges, primarily in modeling complex point motions from sparse input [39].

A straightforward and widely used approach for handling motion with 3D Gaussians is the frame-by-frame approach. This technique involves training and rendering each frame of a dynamic scene independently using Gaussian Splatting. Once processed, these frames are sequenced chronologically to reconstruct the complete motion. In this approach, every frame is treated as an independent static scene, with its own set of 3D Gaussians optimized separately through the standard training and rendering pipeline. The resulting frames are then assembled into a continuous sequence using specialized software tools such as *Postshot* or *SuperSplat*, which facilitate smooth playback.

The primary advantage of this method is its simplicity. Since each frame is optimized independently, it bypasses the complexities of modeling temporal deformations or motion fields across frames, making implementation computationally straightforward.
However, the frame-by-frame approach has notable drawbacks. The most significant limitation is its high memory consumption. Since each frame is stored as a separate file with its own set of Gaussians, no temporal compression is applied. As a result, storage requirements increase significantly for longer sequences, quickly becoming impractical for animations beyond a few seconds.
Additionally, temporal inconsistency is another major issue. Since each frame is computed separately, the set of 3D Gaussians can fluctuate in size and shape between frames, leading to visual artifacts such as flickering during playback. This lack of coherence between consecutive frames undermines the realism of the reconstructed motion.

Given these limitations, a more effective approach would be to model correspondences over time, explicitly representing Gaussian motion and shape changes rather than simply sequencing independently computed frames. To address this, we introduce 4D Gaussian Splatting [39].

2.5 4D Gaussian Splatting

This section introduces 4D Gaussian Splatting (4D-GS) [39] as a technique for extending Gaussian Splatting to dynamic scene rendering.

4D Gaussian Splatting [39] is presented as a comprehensive representation for

dynamic scenes, aiming to achieve real-time rendering while maintaining high training efficiency and optimized storage. Instead of treating each frame independently, this approach models Gaussian motion and shape transformations through an efficient Gaussian Deformation Field Network, which consists of a Spatial-Temporal Structure Encoder and a compact Multi-head Gaussian Deformation Decoder.
Unlike frame-by-frame methods that require separate sets of Gaussians for each timestamp, 4D-GS maintains a single set of 3D Gaussians, which are dynamically transformed over time by the Gaussian Deformation Field. This transformation process captures both motion and deformation, ensuring consistency across frames.
In terms of performance, 4D-GS enables real-time rendering of dynamic scenes, reaching 82 FPS at 800×800 resolution for synthetic datasets and 30 FPS at 1352×1014 resolution for real-world datasets.

2.5.1 4D Gaussian Splatting Framework

The 4D Gaussian Splatting [39] framework extends traditional 3D Gaussian Splatting [5] by incorporating temporal deformations, enabling the rendering of dynamic scenes. This process begins with a given view matrix $M = [R, T]$, which defines the camera's rotation and position, along with a timestamp t that specifies the moment in time being rendered. To model scene motion, the framework integrates both 3D Gaussians $\mathcal{G}$ and a Gaussian Deformation Field Network $\mathcal{F}$, which learns how these Gaussians transform over time.

To generate a novel-view image at time t, a differential splatting process $\mathcal{S}$ is applied, defined as:

$$\hat{I} = \mathcal{S}(M, \mathcal{G}')$$

where $\mathcal{G}'$ represents the updated set of Gaussians after deformation:

$$\mathcal{G}' = \Delta\mathcal{G} + \mathcal{G}$$

Here, $\Delta\mathcal{G}$ denotes the predicted deformation applied to each Gaussian.
The deformation process is driven by a Gaussian deformation field network $\Delta\mathcal{G} = \mathcal{F}(\mathcal{G}, t)$, which encodes, through a spatial-temporal structure encoder $\mathcal{H}$, both spatial and temporal features of 3D Gaussians:

$$f_d = \mathcal{H}(\mathcal{G}, t)$$

These features are then passed through a multi-head Gaussian deformation decoder D, which decodes the features and predicts the necessary deformation for each Gaussian:

$$\Delta\mathcal{G} = \mathcal{D}(f)$$

Once this transformation is applied, the deformed Gaussians $\mathcal{G}'$ represent the scene at the specified timestamp.

This approach ensures temporal coherence, meaning that the deformed Gaussians maintain smooth transitions over time, reducing flickering and inconsistencies. Instead of requiring a separate set of Gaussians for each time step, 4D-GS dynamically updates the existing Gaussians through learned deformations, preserving the efficiency of the differential splatting.[39]

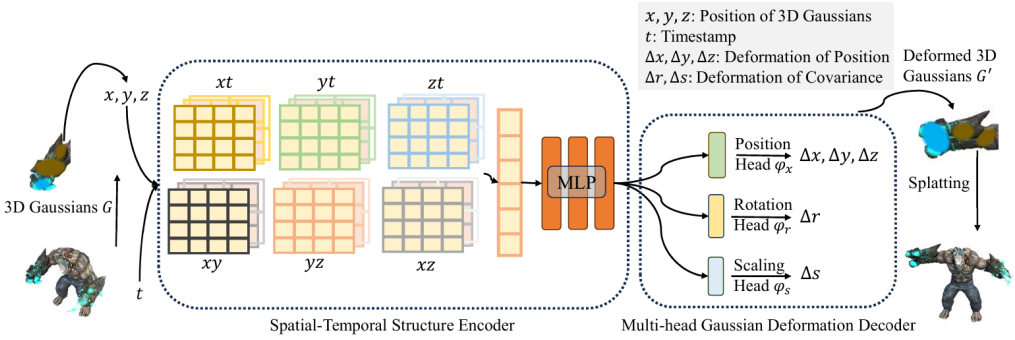

Figure 2.13: 4D Gaussian Splatting Framework
Source: [39]

2.5.2 Gaussian Deformation Field Network

The network responsible for learning the Gaussian Deformation Field incorporates an efficient Spatial-Temporal Structure Encoder $\mathcal{H}$ and a Gaussian Deformation Decoder $\mathcal{D}$, which predict the deformation of each 3D Gaussian.[39]

The **Spatial-Temporal Structure Encoder** $\mathcal{H}$ [39] is designed to effectively model the features of 3D Gaussians over time by capturing both spatial and temporal information.
Since nearby 3D Gaussians tend to share similar spatial and temporal properties, the encoder efficiently represents these relationships. The method adopts a 4D K-Planes [40] module to decompose the 4D neural voxel into six multi-resolution 2D planes: this reduces memory consumption while still preserving essential structural information. Each 3D Gaussian in a given area is mapped into these planes; the deformation of the Gaussians is also encoded in close temporal voxels.

Mathematically, the Encoder consists of six multi-resolution plane modules $R_l(i,j)$ and a small MLP ϕ_d. The spatial-temporal encoding is represented as:

$$\mathcal{H}(\mathcal{G},t) = \{R_l(i,j), \phi_d | (i,j) \in \{(x,y),(x,z),(y,z),(x,t),(y,t),(z,t)\}, l \in \{1,2\}\}$$

where $\mu = (x,y,z)$ represents the mean position of the 3D Gaussian. Each voxel module is defined by:

$$R(i,j) \in \mathbb{R}^{h \times lN_i \times lN_j}$$

where h represents the hidden dimensionality of features, N denotes the resolution

of the voxel grid, and l represents the upsampling scale. This structure encodes information from the 3D Gaussians within six 2D voxel planes while also considering the temporal dimension.

In order to compute separate voxel features, bilinear interpolation is applied at the four nearest grid vertices of each voxel plane. The final feature vector is then obtained by merging the interpolated values:

$$f_h = \bigcup_l \prod \text{interp}(R_l(i,j))$$

where (i,j) belongs to one of the six spatial-temporal coordinate pairs. Finally, a tiny MLP ϕ_d merges these features into a unified feature representation:

$$f_d = \phi_d(f_h)$$

which serves as the final encoded feature for each 3D Gaussian.[39]

The **Multi-head Gaussian Deformation Decoder** [39] is responsible for computing the deformation of 3D Gaussians after their features have been encoded. This decoder, denoted as $\mathcal{D} = \{\phi_x, \phi_r, \phi_s\}$, consists of separate MLPs that independently predict different types of transformations for each Gaussian:

- Position Deformation ($\Delta\mathcal{X}$): Computed by $\phi_x(f_d)$, this term determines the positional displacement of the Gaussian in 3D space;
- Rotation Deformation (Δr): Given by $\phi_r(f_d)$, this component modifies the rotation parameters;
- Scaling Deformation (Δs): Obtained through $\phi_s(f_d)$, this factor adjusts the scale of the Gaussian.[39]

The final deformed Gaussian parameters are determined using the following equation:

$$(\mathcal{X}', r', s') = (\mathcal{X} + \Delta\mathcal{X}, r + \Delta r, s + \Delta s)$$

where:

- $\mathcal{X}'$ is the new position;
- r' is the updated rotation;
- s' is the adjusted scale.[39]

After applying these transformations, the deformed Gaussians are represented as:

$$\mathcal{G}' = \{\mathcal{X}', s', r', \sigma, \mathcal{C}\}$$

incorporating their new spatial attributes while maintaining opacity (σ) and color ($\mathcal{C}$). These updated Gaussians are then used for rendering, ensuring the dynamic adaptation of the scene.[39]

2.5.3 Optimization

The optimization process in 4D Gaussian Splatting [39] follows a structured approach to refine dynamic scene representations efficiently. It begins with an initialization phase, where Structure-from-Motion (SfM) is utilized to generate a sparse 3D point cloud, providing a robust starting point for Gaussian optimization. Similar to its 3D counterpart, 4D-GS optimizes the 3D Gaussians for an initial 3000 iterations, refining their structure before incorporating temporal deformations.
Once the foundational 3D Gaussians are established, the optimization process proceeds by rendering images using these Gaussians ($\hat{I} = \mathcal{S}(M, \mathcal{G})$), rather than directly using the temporally deformed 4D Gaussians ($\mathcal{G}'$). This gradual transition helps stabilize the training process, allowing the deformation model to learn smooth motion trajectories. A loss function is subsequently incorporated to supervise the training process.[39]

Chapter 3

Gaussian Splatting Implementation Process

In this chapter, the implementation steps used to integrate Dynamic Gaussian Splatting inside of *Volucap*'s pipeline will be explored.
Both Frame-by-frame Gaussian Splatting and 4D Gaussian Splatting will be analyzed and their potential integration within the company's workflow will be considered.

Mesh-based and Gaussian-based rendering techniques substantially differ in the way they deal with scene reconstruction. While mesh-based rendering explicitly represents surfaces using connected polygons, Gaussian-based rendering employs a volumetric representation where the scene is modeled as a collection of 3D Gaussians. Meshes require predefined connectivity and rely on Multi-View Stereo techniques for 3D reconstruction. On the other hand, Gaussian-based rendering is explicit, unstructured, but more importantly differentiable: this allows for smooth optimization via gradient descent and efficient rendering through splatting techniques, avoiding also the use of MVS, which often causes issues while reconstructing complex scenarios.[5] Considering their deep structural differences, the workflow of these two methods, in order to obtain a reconstructed 3D scene, results quite different. However, the first few steps of the pipeline used for meshes are common to the first few steps needed in order to integrate a Gaussian-based rendering inside of a volumetric capture workflow.

As previously mentioned, Gaussian Splatting requires as an input a set of images that capture the scene along with the corresponding cameras calibrated by Structure-from-Motion [25]. SfM not only estimates the camera parameters but also generates a sparse point cloud of the scene. This sparse model serves as input for the Gaussian training process.
Typically, Gaussian Splatting workflows utilize softwares like COLMAP or similar camera calibration systems to perform SfM. These systems estimate the positions of each camera based on feature points detected in the input images, resulting in both camera parameters and a sparse point cloud of the scene.
As part of the mesh-generation process employed by *Volucap*, a sparse set of points

is also created. After feature point matching, these points are triangulated, forming a sparse point cloud. Since this cloud is generated from externally calibrated, high-precision camera data, it proves to be significantly more accurate than those obtained through conventional SfM-based calibration methods. *Volucap*'s workflow, as a matter of fact, incorporates a highly precise camera calibration system: before each shooting, the cameras undergo a dedicated calibration process using a checkerboard pattern, which is captured from multiple angles by all cameras. This method allows for structured and reliable feature point matching, surpassing the standard approach used in SfM-based tools like COLMAP, where feature points are extracted from arbitrary scene images that may lack sufficient detail.
Given this, we opted to bypass the default camera calibration step typically included in Gaussian Splatting pipelines and instead directly import *Volucap*'s precomputed camera parameters. This modification allows us to fully exploit the superior accuracy of *Volucap*'s calibration system while seamlessly integrating it into the Gaussian Splatting framework.

To summarize, the initial stages of the pipeline, up to the creation of the sparse point cloud, are shared between both workflows. These include Camera Calibration, Data Conversion, Masking & Rigging, and Sparse Cloud Creation.
At this point, the two workflows diverge:

- In the mesh-based rendering workflow, the process proceeds with depth estimation and the reconstruction of a dense 3D geometry;
- In the Gaussian-based rendering workflow, the pipeline transitions into the training phase, where 3D Gaussians are optimized for novel-view synthesis.

In this chapter, the complete pipeline for Gaussian-based rendering within *Volucap*'s volumetric capture workflow will be detailed. Since the initial steps were already described in the context of mesh-based rendering [Section 2.1.3], they will not be reiterated.

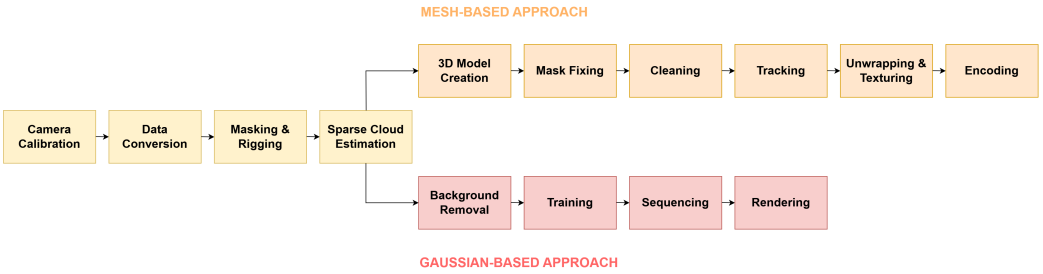

Figure 3.1: Mesh-based and Gaussian-based pipelines

3.1 Input data

Standard Gaussian Splatting workflows require undistorted images of the scene along with sparse point cloud data as input. These images are exported as JPG files

undistorted, accounting for camera calibration parameters, through the company's proprietary meshing tool. Once processed, they are stored in a dedicated folder, *images* [Figure 3.2].

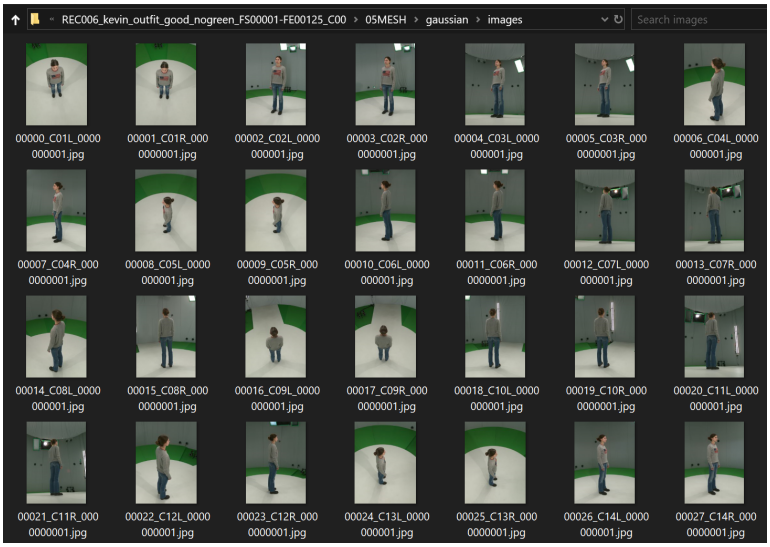

Figure 3.2: *images* folder, containing undistorted footage

As shown in Figure 3.2, the footage stored in the *images* folder includes the background. If training were performed using this raw input, not only would the human avatar be reconstructed, but also the surrounding studio environment. To ensure a cleaner, ready-to-use output, it's possible to apply the masks generated earlier in the pipeline during Masking, a step shared with the meshing process. These masks, exported in an undistorted state from the meshing tool to perfectly align with the images, are applied using a custom external script. This results in a new folder, *masked_images*, containing the masked images, which can now be used as input for the training process.

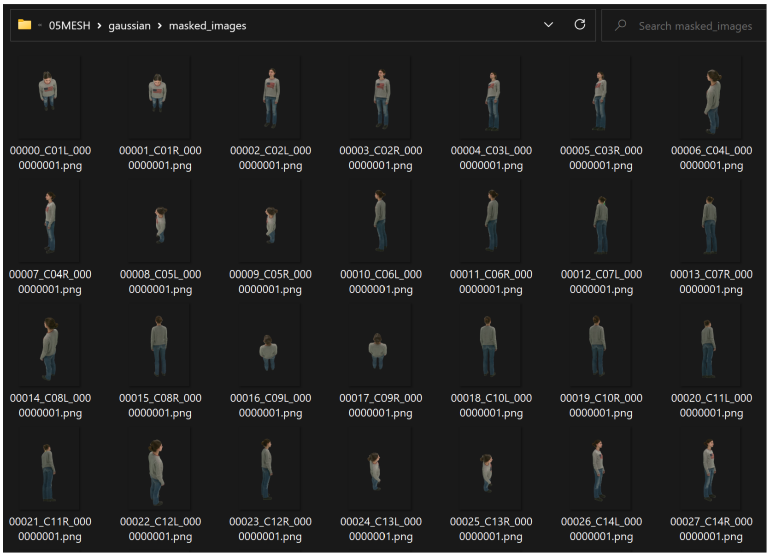

Figure 3.3: *masked_images* folder, containing masked footage

Another essential input is the sparse model data. Gaussian Splatting workflows often require the sparse model in COLMAP format. This data is extracted directly from *Volucap*'s meshing tool, which allows to export the sparse model (with undistorted camera parameters) in both Binary File Format (.bin) and Text File Format (.txt).
Once exported, the sparse model data is stored inside a specific folder, *sparse*, which contains three .bin or .txt files that constitute the sparse model: *cameras*, *images*, and *points3D*. They are structured as follows:

- *cameras*: this file contains the intrinsic parameters of all reconstructed cameras in the dataset;
- *images*: this file contains the pose and keypoints of all reconstructed images in the dataset;
- *points3D*: this file contains the information of all reconstructed 3D points in the dataset.[41]

Since the .bin format proves to be more efficient, it is preferred for training the Gaussian-based scene [Figure 3.4].

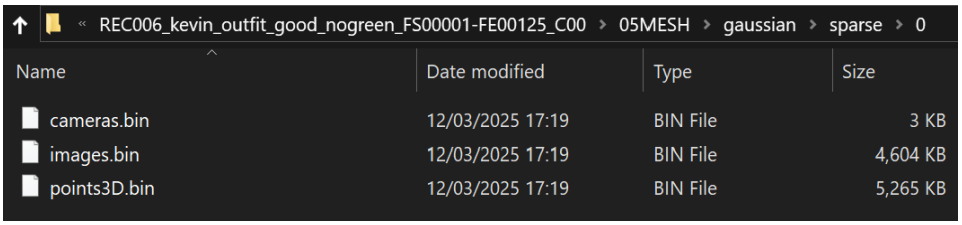

Figure 3.4: *sparse* folder, containing the sparse model data

3.2 Frame-by-frame production pipeline

Frame-by-frame Gaussian Splatting reconstructs dynamic scenes by training and rendering each frame independently and then sequencing them chronologically to reconstruct the complete motion.

To reconstruct static frames and sequence them effectively, we utilized *Jawset Postshot*, a comprehensive software solution for radiance field processing. *Postshot* offers a user-friendly graphical interface that facilitates the tracking, training, editing, and rendering of radiance fields. Notably, it integrates Gaussian Splatting techniques to achieve fast and memory-efficient training within a seamless workflow. Users can import images or videos, along with camera poses and sparse points. The live preview feature allows users to observe the scene's progression during training. Additionally, *Postshot* supports exporting trained scenes as PLY files, enabling compatibility with various applications that support Gaussian Splatting.

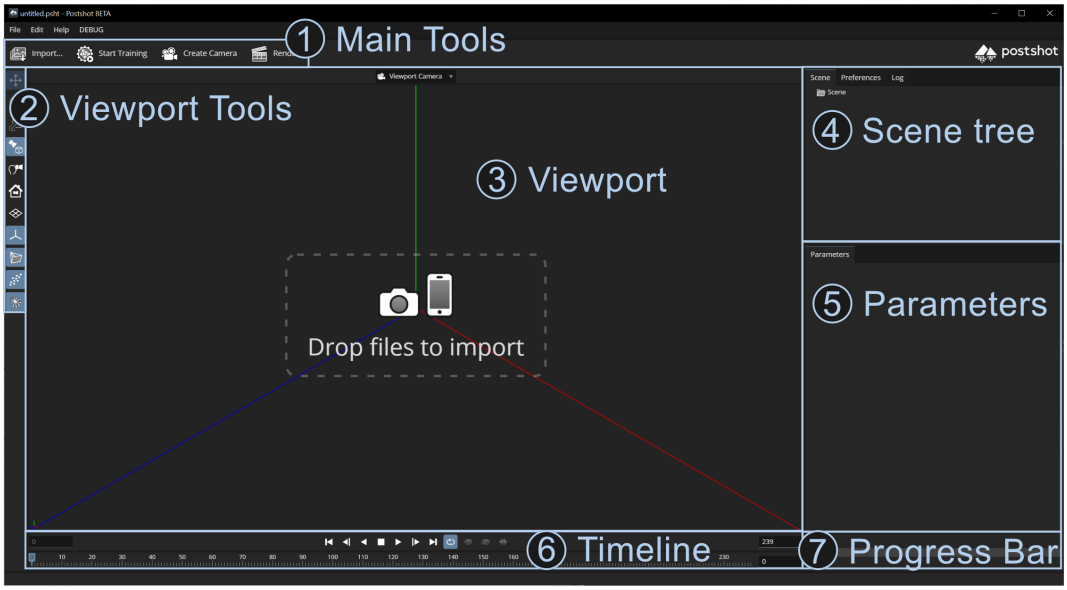

Figure 3.5: *Postshot* interface
Source: `http://bit.ly/4kNNnFL`

Postshot can independently determine camera poses through a process known as Camera Tracking. However, due to the previously discussed reasons, we opt to bypass this step by importing externally calibrated camera data.
Considering this approach, *Postshot*, like other Gaussian Splatting workflows, requires undistorted images of the scene along with sparse point cloud data as input. Since Frame-by-frame Gaussian Splatting treats each frame independently, only the images corresponding to a single frame are used in this workflow. Given that *Volucap*'s volumetric capture system consists of 42 cameras, each static scene is reconstructed using 42 input images capturing the subject from all angles. These images are stored inside of the folder *images*.
Postshot also requires as an input the sparse model in COLMAP format.

In order to start training a scene through *Postshot*, both of these folders need to be imported inside of the software. When a *images* folder gets imported, *Postshot* will recognized the content of this folder as the training footage; when a *sparse* folder gets imported, *Postshot* will automatically recognize the camera poses.
The training menu contains a series of different parameters that can be changed based on the training configuration we want to set and the result we wish to obtain:

- Image Selection:
  - Use Best Images: with this setting, *Postshot* will select, out of all the input images, only the best ones that are well distributed across the scene for camera tracking and radiance field training;
  - Use All Images: All imported images will be used for tracking and training. When using externally calibrated cameras, this option is set as default;
- Max Image Count: When the Use Best Images setting is enabled, the Max

Image Count determines how many images are selected from the imported sequence, typically between 100 and 300. While fewer than 100 images can be used, at least 100 are generally needed for reasonable results. Adding many more images, instead, won't degrade quality but may not improve it either, especially if they come from similar viewpoints. When using externally calibrated cameras, this number is set as default as the total number of imported images;

- Camera poses: When importing images or videos, *Postshot* will execute the Camera Tracking process, computing the camera poses from the images. This process will be executed before the training step. If the imported shot uses externally calibrated cameras, the camera poses are imported and the tracking step is avoided;

- Single Lens & Focal Length: This advanced setting can be used when all of the imported images were shot with the same lens and focal length setting and it's useful to create more stable camera poses. In case of imported camera poses, this setting cannot be used;

- Max Features/Frame: This advanced setting defines the maximum number of feature points that will be extracted per frame during Camera Tracking. However, since the imported sparse model already contains feature points, this setting is not applicable in our specific case;

- Radiance Field Profile: *Postshot* supports different profiles to create radiance fields. Both Splat profiles allow for very fast rendering and quickly reconstruct fine detail in well-covered regions of the scene. The two possible profiles are:

  - Splat MCMC: this profile allows limiting the number of Splat primitives (see Max Splat Count) and therefore the amount of memory and disk space the resulting model requires;
  - Splat ADC: this profile is very similar to the previous one, but differs in the way it produces detail in the scene during training. It's possible to control the amount of detail it creates during training through the Splat Density parameter;

- Downsample Images: To reduce training time, *Postshot* allows the use of downscaled images. Reducing the image resolution could lead to a loss of detail in the trained model. However, the extent of this impact depends on the level of detail present in the original images and the degree to which they are downscaled;

- Max Splat Count: This parameter is exclusive to the Splat MCMC profile and defines the maximum number of Splat primitives that will be generated during training. This directly impacts the memory and disk space required for the radiance field model, as well as the level of fine detail that can be represented;

- Create Sky Model: Outdoor shots often lack sufficient texture in the sky to guide reconstruction, leading to large floating artifacts appearing lower than they should. To mitigate this, *Postshot* can generate a specialized sky model that projects the environment onto a large surrounding sphere, reducing such artifacts;
- Splat Density: this parameter is available only for the Splat ADC profile and it controls the level of detail that will be created in the scene. Values larger that 1.0 increase sensitivity, adding splats even for minor inaccuracies; values smaller than 1.0 reduce sensitivity, resulting in fewer splats and a less detailed representation;
- Start Training: When selected, this option automatically starts tracking (if no sparse model is imported) and training after the images are imported;
- Stop Training After: If enabled, *Postshot* will stop training after reaching the specified number of steps. A recommended starting point for many scenes is 30k steps. While most convergence happens in the early stages, image quality can continue to improve significantly with double or even more training steps.[42]

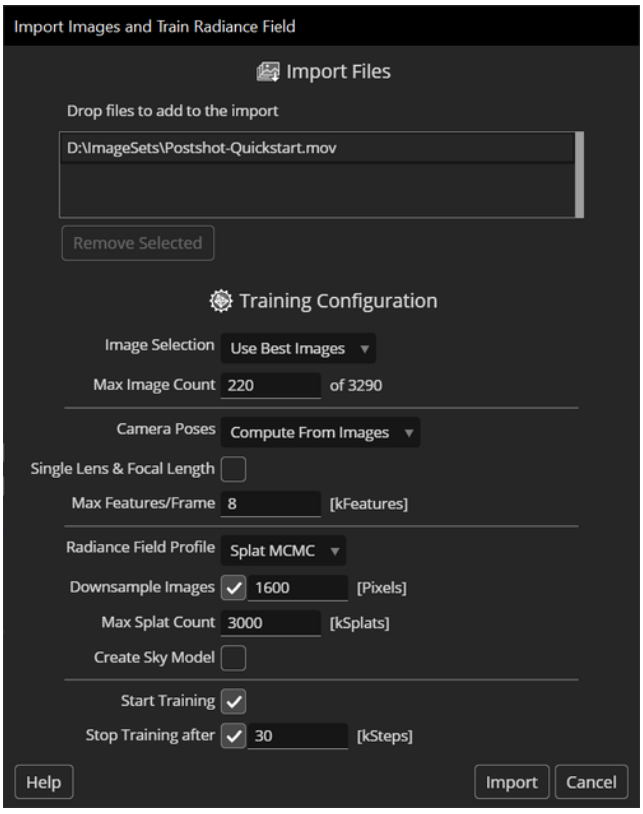

Figure 3.6: *Postshot* Training Configuration
Source: `https://bit.ly/4irMK2O`

With the necessary parameters set, the training process can begin. Since this workflow uses imported camera poses, *Postshot* bypasses the image selection and camera tracking stages, directly initiating the training of the radiance field.

During training, users can monitor progress in real time through the live preview feature. This allows them to observe how the radiance field is gradually formed, not only from a fixed perspective but also by freely moving around the scene to inspect the evolving reconstruction. The total training duration depends on the resolution of the input images and the selected training parameters.
Once the training process is complete, the set of Gaussians defining the scene stabilizes, enabling full free-view exploration. *Postshot* provides a Viewport Camera for smooth navigation, offering intuitive controls such as zoom, tilt, and pan. While this camera is intended for previewing and cannot be animated or rendered, other viewing options are available. If the scene contains an Image Set with tracked camera poses, users can select these cameras in the Scene Tree, cycling through the original input images and matching camera viewpoints to compare the trained radiance field with the captured footage. Additionally, custom cameras can be created within the scene, allowing to set its properties such as focal length, but also animate camera trajectories, and eventually render video sequences.
After training, a range of post-processing actions can refine the scene. The entire radiance field can be translated, rotated, or scaled. To further clean the output, *Postshot* includes a Splat Selection tool, which enables manual selection and removal of specific Gaussians. Additionally, the Cropping tool, available immediately after training, creates a bounding box around the subject, eliminating unwanted Gaussians outside of it. These tools are particularly useful for removing background artifacts or misplaced floating splats.

At this stage, the finalized scene can be saved either as a *Postshot* project file (.psth) or as a PLY file (.ply) for compatibility with other Gaussian Splatting-supported applications and viewers. *Postshot* also offers the capability to render a video of the scene. To achieve this, users must create and animate a virtual camera, defining smooth camera movements around the subject before exporting the final rendered sequence. For further compositing and editing, *Postshot* scenes can be loaded into Adobe After Effects using the dedicated *Postshot* plugin. Once imported, the software's camera controls and animation tools can be leveraged to refine and render the scene as part of a larger composition.

Once the .ply file is saved, the process can be repeated for each subsequent frame, applying the same training method to reconstruct the entire sequence frame by frame. Once all frames in the desired sequence have been individually trained, assembling them into an animation becomes a straightforward process. Each .ply file corresponding to a trained frame must be imported into the scene. At this stage, the Timeline tool integrated within *Postshot* proves particularly useful. This tool allows users to import multiple .ply files simultaneously, sequence them in chronological order, and play them back as a continuous animation. The Timeline interface provides standard playback controls, enabling users to preview the animation and make adjustments as needed. Additionally, keyframes can be added to control specific parameters, allowing

for smooth transitions and refined motion within the sequence.

As with the reconstruction of static frames, a virtual camera can be introduced into the scene. This camera can be animated using keyframes, enabling dynamic perspectives and smooth motion paths throughout the animation. Once the scene is finalized, *Postshot* provides the option to render the animation as a video file, capturing both the reconstructed subject and any camera movements applied.
For additional refinements, the exported sequence can be further edited in post-processing software such as Adobe After Effects, where additional visual effects, color grading, and compositing can be applied.

3.3 4D Gaussian Splatting production pipeline

4D Gaussian Splatting [39] is a dynamic scene rendering technique that models Gaussian motion and shape deformation using a Gaussian Deformation Field, maintaining a single set of 3D Gaussians that are dynamically transformed over time.

The implementation of 4D Gaussian Splatting is available on the authors' GitHub page. This codebase facilitates data preparation, training, and rendering through command-line operations and includes a viewer for real-time monitoring of training progress. To set up the environment, users need to clone the repository and utilize an Anaconda environment to install the necessary Python packages, including PyTorch.

The data preparation process involves calibrating input images to generate camera poses and a sparse point cloud required for training. For multi-view scene datasets, this can be achieved by:

- Using the *multipleviewprogress* script included in the repository to generate camera poses and point cloud data;
- Employing *nerfstudio*, an external repository that allows to process the input data through COLMAP and FFMPEG, providing camera poses and a sparse cloud as an output;

After data preparation, the training phase begins, utilizing the processed data and a configurable training setup defined in a configuration file.
In this configuration, the main parameters we can intervene on are:

- *iterations*: determines the total number of training iterations;
- *batch_size*: determines the number of samples processed in one iteration of training;
- *coarse_iterations*: specifies the number of iterations in the coarse, warm-up training phase;

- *densify_until_iter*: indicates the iteration until which the model continues to densify the Gaussians;
- *opacity_reset_interval*: defines how frequently the opacity values are reset during training.

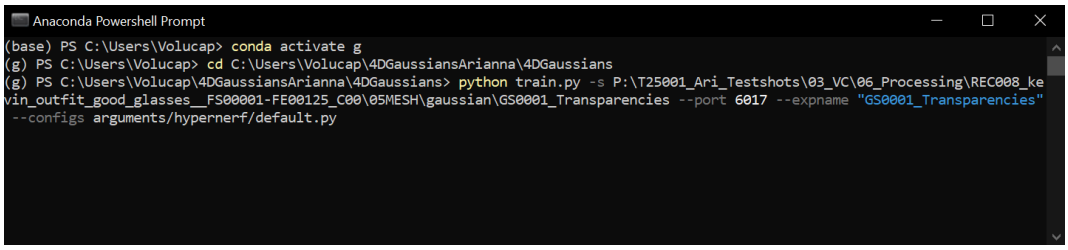

Figure 3.7: 4D-GS training command

When training is completed, the resulting radiance field, exported in .ply format along with some deformation files, can be viewed using external viewers such as the one included in the *Gaussian Splatting PyTorch Lightning Implementation*.
The implementation also includes a rendering step that outputs a series of still images and an .mp4 file, creating a rendered video of the animated scene from a specified camera perspective.

3.4 Evaluation of the approaches

Both methods present a number of advantages and disadvantages, that have been analyzed in order to choose which of these two methods we want to adopt as the preferred one.

3.4.1 Frame-by-frame approach

The Frame-by-frame approach, implemented using *Postshot* offers several key advantages. The primary benefit is its ease of use. The software installation is extremely straightforward, requiring no additional external packages or libraries. Users can simply download an executable file from the official *Jawset* website, complete a quick installation, and immediately begin training radiance fields. The inclusion of a graphical user interface (GUI) further simplifies the training and rendering process, making Gaussian Splatting accessible even to those with limited technical expertise. This user-friendly interface significantly lowers the barrier to entry for dynamic scene reconstruction with Gaussian Splatting.
Additionally, *Postshot* provides a variety of training options and post-processing tools that allow for fine-tuning and editing of reconstructed scenes. The software enables users to generate rendered videos where they can not only choose specific viewpoints but also animate complex camera movements using keyframes. This flexibility grants significant creative control over the final output.

Another major advantage of the Frame-by-frame approach is the ease of integrating external camera calibration data into the training process. Since each frame is trained independently, it is simple to bypass the automatic camera calibration included in standard Gaussian Splatting implementations and instead utilize the highly accurate camera poses and sparse cloud generated by *Volucap*'s processing pipeline. This ensures a more precise reconstruction of the scene.

Despite these benefits, the Frame-by-frame approach also presents several notable drawbacks. The most significant limitation is its high memory consumption. Since each frame is stored as an independent file with its own set of Gaussians, no temporal compression is applied. Consequently, storage requirements increase dramatically for longer sequences, quickly becoming impractical for animations exceeding just a few seconds.
Another major issue is the lack of temporal consistency. Because each frame is computed separately, the set of 3D Gaussians fluctuates in size and shape between frames, leading to noticeable visual artifacts such as flickering during playback. This inconsistency diminishes the realism of the reconstructed motion.
Moreover, training each frame individually and then manually sequencing them into an animation demands significant time and effort. This high level of manual intervention is undesirable, particularly for production pipelines where efficiency and automation are key priorities.

3.4.2 4D Gaussian Splatting approach

The primary advantage of the 4D Gaussian Splatting approach is its ability to model correspondences over time, capturing Gaussian motion and shape deformations instead of merely sequencing independently computed frames. This is achieved through a single set of Gaussians that dynamically shift and deform throughout the sequence, eliminating flickering artifacts present in the Frame-by-frame approach. The seamless transformation of Gaussians ensures smooth transitions between frames, significantly improving temporal coherence.
Another key benefit of 4D-GS is its inherent temporal compression, which drastically reduces output file sizes and minimizes storage requirements. Additionally, since multiple frames are trained simultaneously, the overall workflow demands far less manual intervention compared to the Frame-by-frame method, as it eliminates the need for training single frames and sequencing them. Given these advantages, 4D-GS appears to effectively resolve the primary limitations of the Frame-by-frame approach.

However, a deeper analysis of 4D-GS has revealed several significant challenges that are difficult to overcome. The first major obstacle is the complexity of installation. The repository presents compatibility issues between CUDA, PyTorch, and required package versions, making it difficult to set up. A review of reported issues on the official repository confirms that many users face similar struggles, indicating

that this implementation requires technical expertise and is not easily accessible to a wider audience.
Another considerable limitation is the difficulty of integrating external camera calibration data. *Volucap*'s proprietary meshing tool, like many others on the market, exports a separate sparse model for each frame. In contrast, 4D-GS requires a cumulative sparse model that incorporates all input images across the entire sequence, rather than per-frame data. This discrepancy makes direct integration of *Volucap*'s sparse model into 4D-GS highly complex. While integration is not impossible, it would demand extensive modifications to either *Volucap*'s meshing tool (to generate a cumulative sparse model) or to 4D-GS itself (to support multiple per-frame sparse models as input). Either approach would require significant development effort and code adaptation. An alternative would be generating the sparse model using the built-in tools provided within the 4D-GS implementation. However, this would mean foregoing *Volucap*'s high-precision camera calibration data and instead relying on a less accurate reconstruction, which is not ideal for the company's objectives.
A further drawback of 4D-GS refers to rendering performance. According to the official paper [39], 4D-GS achieves real-time rendering at 30 FPS with a resolution of 1352×1014 when using real-world datasets. Applying this approach to *Volucap*'s custom dataset would necessitate a significant reduction in image resolution, potentially affecting the level of detail captured in the trained Gaussian model. Higher-resolution input images contribute to more accurate reconstructions of intricate textures and complex environments, which is a crucial aspect of *Volucap*'s testing and development efforts.

3.4.3 Method of Choice

After careful evaluation and extensive testing conducted during the initial phase of the collaboration with *Volucap*, the Frame-by-frame approach was selected as the preferred method for implementing Dynamic Gaussian Splatting. Several key factors contributed to this decision.

One of the main reasons for selecting the Frame-by-frame approach was the superior reconstruction quality it enabled, particularly when leveraging *Volucap*'s highly precise camera calibration data. The ability to integrate externally calibrated camera poses and sparse point clouds seamlessly into the training process ensured a more accurate and detailed scene representation, which is essential for high-quality volumetric capture. Additionally, this approach provided a much smoother integration within *Volucap*'s existing pipeline, avoiding the complexities associated with 4D-GS. The latter presented considerable installation challenges and required extensive code modifications to incorporate external calibration data, making it far less practical in this context. The ability to bypass the standard automatic camera calibration in favor of a more precise, precomputed calibration workflow eliminated a major bottleneck in achieving high-fidelity results.

Furthermore, despite the inherent memory limitations of storing each frame separately, the Frame-by-frame approach remains a practical solution for handling short video sequences. While temporal coherence is not inherently maintained across frames, the stability of the reconstruction process and the reliability of the output compensate for this limitation.

Chapter 4

Meshes VS Gaussian Splatting

The primary goal of this study is to evaluate the effectiveness of Gaussian Splatting as a dynamic scene rendering technique for human avatars, particularly in the context of XR experiences. Specifically, this research investigates whether Gaussian-based rendering techniques offer improvements over traditional mesh-based reconstruction, particularly in challenging scenarios previously discussed in Section 2.2.
Since *Volucap*'s establishment in 2018, meshes have been the company's standard method for representing 3D scenes in volumetric capture workflows. While mesh-based rendering provides well-established advantages, it struggles with complex scene reconstruction . These limitations arise primarily due to the reliance on Multi-View Stereo (MVS) techniques, which can lead to incomplete geometry and visual artifacts. To address these challenges, Gaussian Splatting was explored as a potential alternative for volumetric reconstruction. This method introduces an unstructured, differentiable representation that does not depend on predefined connectivity, making it particularly appealing for capturing intricate scene details.

In this chapter, we present a conducted test designed to systematically evaluate, through practical examples, the complex scenarios that typically challenge mesh-based rendering. The results of this evaluation will be demonstrated through a comparative analysis of scenes reconstructed using both mesh-based and Gaussian-based rendering techniques. In the image captions, Ground Truth images will be labeled as GT, Mesh-based results as M and Gaussian-based results as GS.

4.1 Experimental Setup

The test was conducted over two separate days, totaling approximately 2.5 hours of shooting, at *Volucap*'s volumetric capture studio, located in *Studio Babelsberg*, Potsdam. The studio features 42 ultra-high-resolution cameras, all of which were utilized for the experiment. For further details on the studio setup, refer to Section 2.1.1.
Each take was recorded at a resolution of 6K (4508×6016) per camera, as the tested

scenarios involved limited movement, making the full 9K resolution unnecessary. Prior to the start of each shooting session, a standard calibration and clean plate procedure was performed.
To ensure the test closely mirrored a typical shooting scenario, all standard capture and lighting settings were applied. Each take consisted of 125 frames, recorded at 25 frames per second (5 seconds per take).

The test focused on capturing a human subject in various complex scenarios, using props and outfits that could realistically be included in standard volumetric capture shoots. These scenarios were carefully selected based on past reconstruction difficulties observed by *Volucap*. The tested objects and outfits were planned beforehand in collaboration with the team to comprehensively cover all complex cases. The specific scenarios evaluated include:

- A standard capture scenario;
- Transparent objects;
- Thin objects;
- Reflective objects;
- Loose hair;
- Dark and uniformly-textured outfits.

A more detailed discussion of these scenarios will be provided in the next section. After the shooting was completed, all takes were processed on a Windows machine equipped with an NVIDIA GeForce RTX 3090 GPU (24 GB VRAM), using both reconstruction workflows.

For the mesh-based workflow, the standard pipeline detailed in Section 2.1.3 was followed. All of the 3D models were built with medium quality and textured with 4K textures. Specifically, all 125 frames per take were processed using the mesh-based approach, utilizing all 42 cameras in the studio. The workflow and settings remained consistent to ensure that the comparison was based on a typical output that *Volucap* produces for XR experiences.
For the mesh-based workflow, processing time can vary significantly depending on the workload of the network and workstations at the time the process begins. To ensure consistency, we established an average processing time based on measurements taken when both the network and workstations were free. This average processing time is used uniformly across all tested scenarios and was determined for both a single frame and an entire take (125 frames):

- Single frame: 12 minutes
- Whole take (125 frames): 180 minutes

Notably, the relationship between the processing time for a single frame and that of a full take is non-linear.
These processing times account for all stages from stereo depth estimation onward, marking the point where the two workflows (mesh-based and Gaussian-based) diverge and follow their respective paths. The earlier phases, which are shared between both workflows, are excluded from these estimates.

For the Gaussian-based workflow, the Frame-by-frame approach described in Section 3.2 was used. In this case, each frame was reconstructed independently, incorporating all 42 cameras as input, and later sequenced to form the final dynamic scene. Externally calibrated camera poses and sparse point cloud data were imported. The chosen Radiance Field Profile was *Splat MCMC*, with a fixed number of 30k training steps per frame, a value considered an optimal baseline by the *Postshot* guidelines. The input image resolution used to obtain the results presented in this chapter was set to 4K. Additionally, the Max Splat Count, which determines the number of Gaussian primitives generated during training, was fixed at 3000 kSplats. In the following chapter, however, both parameters will be varied to conduct a quantitative analysis of their impact on key aspects such as training time and output file size.

For these tests, we chose to use masked images as input, capturing only the subject without the background. This decision was made to align the process as closely as possible with the mesh-based workflow, which also relies on masks for mesh creation. Additionally, training time is significantly reduced when the background does not need to be reconstructed.
However, an important consideration must be noted: when training is performed on the full images (background included), the subsequent removal of the background using *Postshot*'s Cropping tool is highly precise. In contrast, when using pre-masked images, any background residuals included in the original masks will also be reconstructed. These residuals are much more difficult to remove once the radiance field has been fully trained, as they appear in close proximity to the Gaussians that should be preserved, making complete removal challenging.
To illustrate this point, we compare two outputs of the same scene:

- Figure 4.1: The model was trained using the full images (background included), and the background was removed afterwards. The input images had a resolution of 6K, and the training took approximately 2 hours and 26 minutes;
- Figure 4.2: The model was trained using pre-masked images. The input images had a resolution of 6K, and the training took approximately 50 minutes.

This particular scenario presents challenges for masked images. Loose hair often allows glimpses of the background through strands, as seen in Figure 4.2. Additionally, the bottle in Figure 4.2 appears more opaque compared to Figure 4.1. This occurs because, in the masked-image approach, the background color is partially baked into the transparent areas of the bottle. Conversely, when the background is removed

after training, the depth information is correctly computed, allowing for a cleaner separation of background and foreground elements.
In this specific test, we prioritized reduced training time over slight losses in accuracy, acknowledging that the masked-image approach may introduce minor artifacts but significantly accelerates the reconstruction process.

Figure 4.1: Bg removed after training

Figure 4.2: Bg removed before training

This experiment aims to assess how Gaussian-based rendering performs in scenarios that typically pose challenges for mesh-based reconstruction. Furthermore, the goal is to determine the optimal input resolution and training parameters that produce Gaussian-based results comparable in quality to the high-fidelity mesh-based human avatars *Volucap* regularly creates for commercial use.

4.2 Tested Scenarios

4.2.1 Standard capture scenario (S)

For the first scenario, we opted for a standard capture setup, one that aligns well with mesh-based reconstruction. The chosen outfit fully complies with the company's wardrobe guidelines. The subject wore a textured grey knitted sweater and a pair of medium-wash blue jeans with an uneven texture. Both pieces feature highly distinguishable details that facilitate feature point detection and depth computation: the sweater's knitted pattern provides rich surface variation, while the jeans include pockets and strategically placed rips, adding well-defined structural elements.
Additionally, to ensure an optimal reconstruction process, the subject's hair was styled in a tight bun, eliminating flyaways or loose strands that could introduce complexity into the capture.

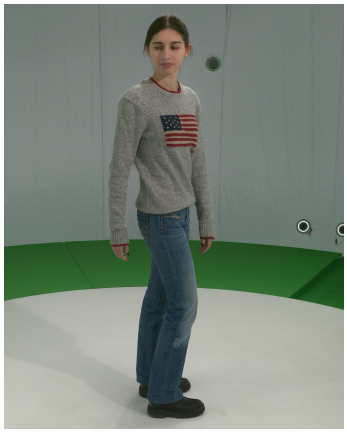

Figure 4.3: S - GT

Figure 4.4: S - M

Figure 4.5: S - GS

The Gaussian-based scene was trained with 4K resolution input images, and the training process was completed in 29 minutes.

4.2.2 Transparent objects (TR)

Next, we tested the reconstruction of transparent objects, which are notoriously difficult, if not impossible, to accurately capture using mesh-based rendering.
For this experiment, we selected three distinct transparent objects to evaluate how each would be reconstructed using both techniques. The chosen objects included a pair of reading glasses, a glass water jug filled halfway, and an empty glass for pouring water. This selection allowed us to analyze how the different objects were handled by the reconstruction process.

Figure 4.6: TR - GT

Figure 4.7: TR - M

Figure 4.8: TR - GS

The Gaussian-based scene was trained with 4K resolution input images, and the training process was completed in 32 minutes.

4.2.3 Thin objects (TH)

The next test focused on thin objects, a category that is typically avoided in volumetric capture due to its challenging reconstruction properties. For this experiment, we

selected two distinct objects: a paper brochure and a baseball cap. These objects differ significantly in thickness, the cap being notably thicker than the brochure. Additionally, the brochure, being made of paper, is far more translucent than the cap's visor. The goal of this test was to assess how the reconstruction quality varies with different object thickness.

Figure 4.9: TH - GT

Figure 4.10: TH - M

Figure 4.11: TH - GS

The Gaussian-based scene was trained with 4K resolution input images, and the training process was completed in 35 minutes.

4.2.4 Reflective objects (R)

Another challenging scenario involved testing reflective objects. For this, we selected a chrome ball, an extremely reflective surface that behaves almost like a mirror, posing significant challenges for reconstruction due to its highly dynamic reflections.

Figure 4.12: R - GT

Figure 4.13: R - M

Figure 4.14: R - GS

The Gaussian-based scene was trained with 4K resolution input images, and the training process was completed in 33 minutes.

4.2.5 Loose hair (H)

The next complex scenario we tested was loose hair, a particularly challenging element to reconstruct due to its intricate geometry and fine details. Hair strands are difficult to capture accurately, often leading to inconsistencies in reconstruction. For this reason, loose hairstyles are typically avoided in volumetric capture, with tighter styles, such as the one used in our Standard Capture scenario, being preferred.
In this case, however, we deliberately tested how loose, wavy hair performs in both mesh-based and Gaussian-based reconstruction, evaluating their ability to handle the complexities associated with fine hair structures.

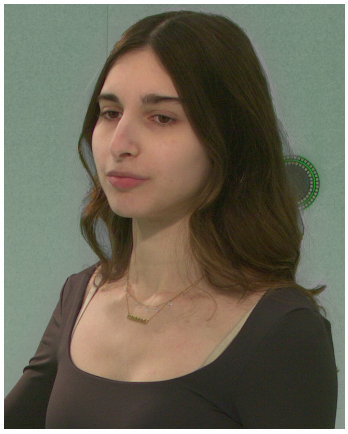

Figure 4.15: H - GT

Figure 4.16: H - M

Figure 4.17: H - GS

The Gaussian-based scene was trained with 4K resolution input images and the training process was completed in 22 minutes.

4.2.6 Dark and Uniformly-textured outfits (DU)

The final scenario focused on capturing an outfit with minimal visible texture and uniform color. The chosen clothes consisted of a brown, slightly shiny top, tightly fitted to the body, and a pair of black, wide-legged pants with a completely uniform color, lacking any noticeable variations or patterns.

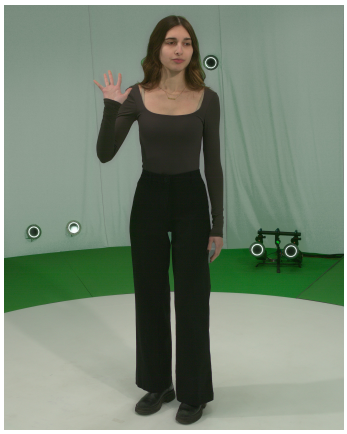

Figure 4.18: DU - GT

Figure 4.19: DU - M

Figure 4.20: DU - GS

The Gaussian-based scene was trained with 4K resolution input images, and the training process was completed in 22 minutes.

Chapter 5

Evaluation of the Results

In this chapter, the outcomes of the conducted tests are presented and analyzed, comparing the mesh-based and Gaussian-based reconstruction approaches. The evaluation is divided into two parts: a quantitative analysis, which considers objective metrics such as training time and file size, and a qualitative evaluation, which examines the visual fidelity of the reconstructed scenes. This comparison aims to provide a comprehensive understanding of the strengths and limitations of each method within the context of volumetric capture.

5.1 Quantitative Evaluation

This section presents a quantitative evaluation of the Gaussian-based rendering approach. The analysis examines how individual training parameters influence both training time and output file size, then comparing these findings with the results obtained from the mesh-based rendering workflow. The objective is to determine, through measurable data, whether the Gaussian-based pipeline can be a viable alternative to the mesh-based approach within *Volucap*'s volumetric capture workflow.

5.1.1 Training Time

As a first step, we investigated how training time is influenced by the resolution of the input images. This test was conducted using a fixed set of input images, which were downsampled, along with their corresponding masks, to different resolutions prior to training. The masks were then applied onto the downscaled images and imported into *Postshot*. This ensures greater control over the entire process.
Each image set, representing a single frame, was trained individually using its corresponding sparse model data. To isolate the effect of input resolution, all other training parameters were kept constant across the different runs. The goal was to observe how changes in image resolution alone affect training time. The following training settings were chosen for each resolution level:

- Radiance Field Profile: *Splat MCMC*;

- Max Splat Count: 3000 kSplats;
- Stop Training After: 30K steps.

For the resolution of the input images, we selected four commonly used formats: 1K, 2K, 4K, and 6K, the latter being the original resolution of the captured frames. A Gaussian-based scene was trained separately for each resolution, in order to observe how input resolution influences the training time.

The results were visualized using three graphs, each illustrating how training time varies with image resolution across three distinct capture scenarios. These graphs help highlight the relationship between image resolution (in pixels) and training time (in minutes). The detailed data used to generate the graphs is provided in three corresponding tables, one for each scenario, which can be found in Appendix A.

Standard capture scenario [Section 4.2.1]:

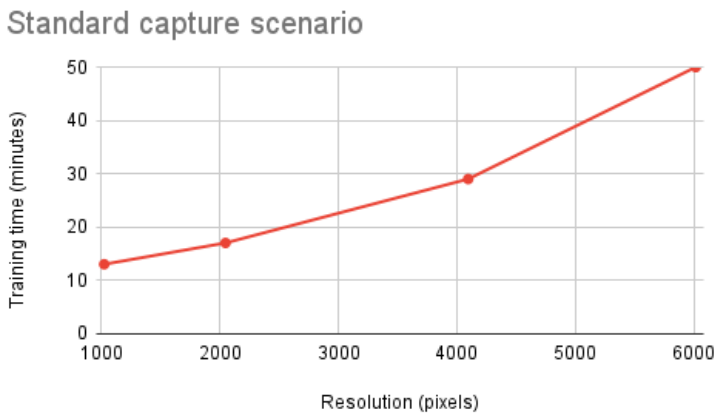

Figure 5.1: Training times for Standard capture scenario

Transparent objects [Section 4.2.2]:

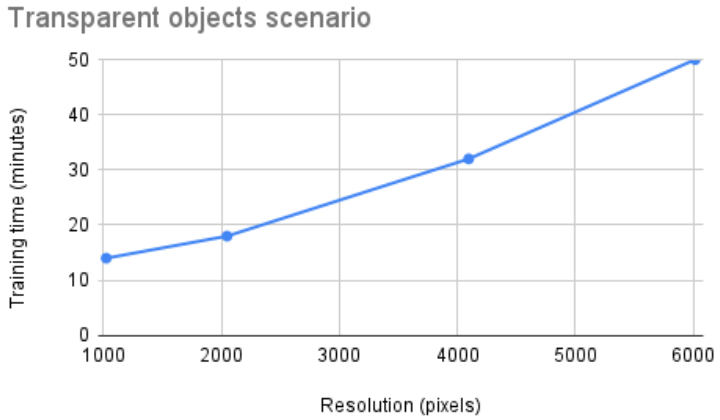

Figure 5.2: Training times for Transparent objects scenario

Dark and Uniformly-textured outfits [Section 4.2.6]:

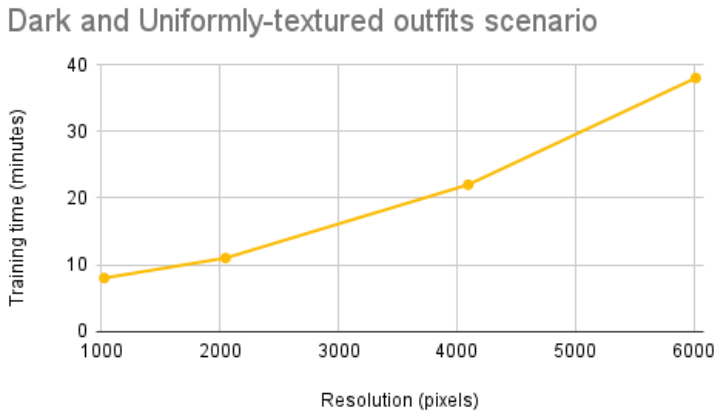

Figure 5.3: Training times for Dark and Uniformly-textured outfits scenario

As illustrated in the three graphs above, the trend between input image resolution and training time is rather linear.
For the scenes displayed in Sections [4.2] and [5.2], we chose to use 4K input resolution for training. This choice provided a balanced compromise: 6K input images significantly increased training time without showing substantial improvements in visual quality, while 2K images began to noticeably degrade reconstruction quality. Moreover, the mesh-based workflow typically employs textures in 4K resolution, making 4K a more consistent and comparable benchmark for the two approaches.

Training time for Gaussian Splatting is directly comparable to Processing time data for mesh-based rendering. As previously discussed in Section 4.1, the estimated average mesh-based processing time is:

- Single frame: 12 minutes
- Whole take (125 frames): 180 minutes

This average was calculated under optimal network and workstation conditions and is used consistently across all scenarios. These processing times account for all steps from stereo depth estimation onward, marking the point where the mesh and Gaussian-based workflows diverge. When comparing these two workflows, the time required to reconstruct a single frame is a fair metric since both methods result in a ready-to-use 3D avatar. For Gaussian Splatting, training time for a single frame using 4K images and 30K training steps ranged from 22 to 35 minutes depending on scene complexity. We determined 29 minutes to be a suitable average value.

This average training time is more than double compared to that of the mesh-based workflow (29 minutes vs. 12 minutes per frame). The discrepancy becomes even more significant when scaling to a full take: mesh-based processing benefits from optimizations across frames, resulting in a non-linear increase in processing

time. In other words, the entire take does not require 12 minutes multiplied by 125 frames. Instead, inter-frame efficiencies bring the total to around 180 minutes.
In contrast, the Gaussian-based frame-by-frame approach requires each frame to be trained independently. Even when only considering training time and excluding setup or sequencing, the total training time for 125 frames (29 minutes $\times$ 125) reaches 3625 minutes, an unfeasible duration for commercial applications. Although training time can be reduced by lowering input resolution and decreasing the number of training steps, this inevitably compromises visual quality, which is critical for professional use.

From this perspective, it becomes clear that the frame-by-frame Gaussian-based approach is not yet practical for full dynamic scene reconstruction due to its high computational demands. However, it remains a viable option for static scenes, where extended training times are more acceptable. In particular, the method demonstrates outstanding performance in complex scenarios that typically challenge mesh-based reconstruction, as will be shown in the next section [5.2].
Depending on the nature of the scene to be reconstructed, especially when dealing with difficult elements such as translucency, fine geometry, or reflectivity, it may be worthwhile to consider the Gaussian-based approach. In such cases, the superior reconstruction quality may outweigh the higher computational cost, making this method a valuable alternative in specific high-complexity use cases.

5.1.2 Output File Size

The next data point we aim to analyze is the output file size. By this, we refer to the size of the `.ply` file that is generated after completing the training process for a single frame using the Gaussian-based approach.
To begin, we analyze how file size scales with changes in the *Max Splat Count* training parameter. This parameter defines the maximum number of splat primitives generated during training and directly influences both disk space requirements and the level of scene detail. For this test, a fixed set of previously masked 4K images was used, maintaining the same input image set throughout all of the different trainings. The goal was to observe how changes in *Max Splat Count* alone affect the file size. To do this, we isolated the effect of this parameter by keeping all other training settings constant:

- Input images resolution: 4K;
- Radiance Field Profile: *Splat MCMC*;
- Stop Training After: 30K steps.

We tested four values for *Max Splat Count*: 500, 1000, 2000, and 3000 kSplats. A separate scene was trained for each configuration, and the resulting file sizes were visualized in the graphs below for two different scenarios. The corresponding numeri-

cal data is presented in Appendix B.

Standard capture scenario [Section 4.2.1]:

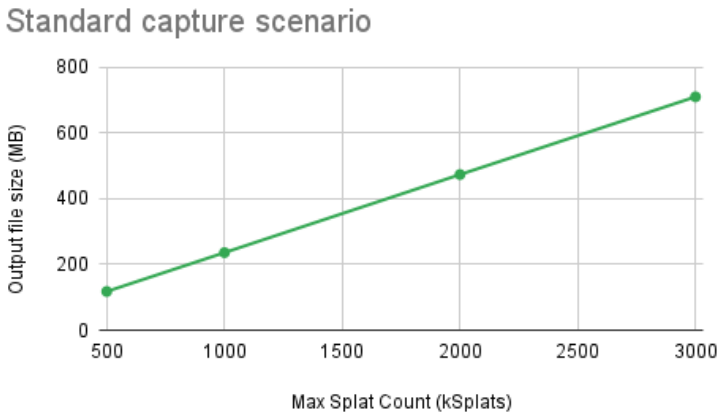

Figure 5.4: Output file sizes for Standard capture scenario

Thin objects [Section 4.2.3]:

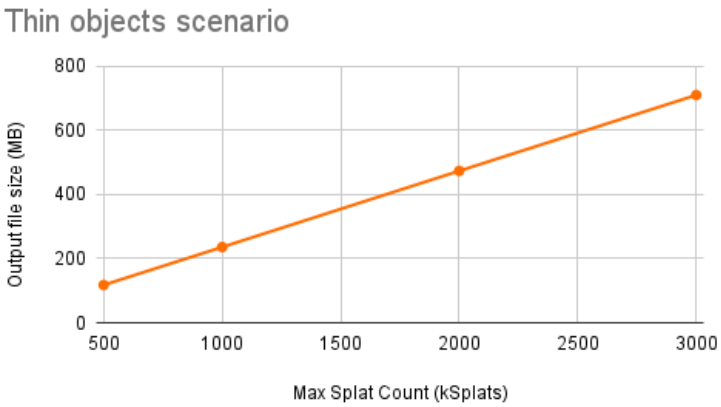

Figure 5.5: Output file sizes for Thin objects scenario

As seen in the graphs, the relationship between *Max Splat Count* and file size is highly linear; moreover, both tested scenarios produced identical numerical results. For Sections [4.2] and [5.2], we adopted the default *Postshot* value of 3000 kSplats. However, a reduction in *Max Splat Count* to 1000 kSplats still produced results with satisfactory visual detail. This makes it a valuable parameter to consider for optimization: a moderate reduction in image quality and detail can lead to significant file size savings.

However, even when reducing the Max Splat Count down to 1000 kSplats, the output file size remains significantly larger compared to that of the mesh-based rendering. For the mesh-based data, we also consider an estimate and present the average file size for both a single frame and a complete take of 125 frames:

- Single frame: 1.85 MB
- Whole take (125 frames): 108 MB

Even when using 1000 kSplats for the Gaussian-based scene, the resulting file size per frame is still far from comparable: while the mesh-based approach averages around 1.85 MB per frame, the Gaussian-based method produces approximately 236 MB per frame. The contrast becomes stark when comparing entire takes: mesh-based rendering benefits from inter-frame efficiencies and shared geometry, while Gaussian Splatting stores each frame independently, causing linear growth in storage requirements, which quickly become unmanageable.

In conclusion, while frame-by-frame Gaussian Splatting shows promising reconstruction capabilities, especially in challenging scenarios, this implementation is not yet scalable for dynamic scenes. Both the extended training time and substantial storage demands pose serious constraints. Although it is technically possible to reduce training time and output size by adjusting input resolution, training steps, and Max Splat Count, these reductions come at the cost of visual fidelity, ultimately compromising the suitability of the results for commercial use.

5.2 Qualitative Evaluation

This section presents a qualitative evaluation of the conducted tests, providing a visual comparison between the two reconstruction approaches. The focus lies particularly on how the complex scenarios described in Sections [2.2] and [4.2] are handled by both the mesh-based and Gaussian-based methods. The aim is to highlight the strengths and weaknesses of each technique and assess which performs better depending on the scenario. The same capture cases introduced in Section [4.2] are used here as the basis for evaluation across both approaches.

The main issues related to mesh-based reconstruction will be discussed in the following sections, together with practical examples. Before getting into those details, it's worth adding a few observations about Gaussian-based reconstruction.
Gaussian Splatting is capable of reconstructing scenes with extremely fine detail and high visual fidelity. However, it is inherently a view-dependent rendering method. This means that the appearance of the reconstructed scene, including its lighting, color accuracy, and level of detail, varies depending on the viewing angle. When the point of view is close to one or more of the input cameras, the reconstruction quality tends to be very high. On the other hand, as the point of view moves away from the original camera perspectives, the quality of the reconstruction can degrade. Therefore, the greater the number of input cameras distributed around the subject, the more complete and accurate the reconstruction becomes. In this test, each frame was captured using 42 cameras, providing broad coverage from multiple angles. However, some areas, such as low-angle views, were not as well covered. These

less-represented viewpoints can result in visible quality drops in the reconstructed scene, often showing up as blurriness or aliasing artifacts [Figure 5.7].
In contrast, mesh-based scenes tend to preserve a more consistent appearance across all viewpoints [Figure 5.6]. This is due to the inherent properties of meshes: they rely on predefined surface connectivity and are not view-dependent, therefore maintaining a solid and stable structure regardless of the viewing direction.

Figure 5.6: Low angle - M

Figure 5.7: Low angle - GS

5.2.1 Standard capture scenario

For the standard capture scenario, both of the workflows ended up creating a result that's quite close to the ground truth image [Figure 4.3], as we can notice from Figures 4.4 and 4.5. Even the finer details ended up being quite well preserved, such as the sweater pattern or the jeans' texture:

Figure 5.8: Sweater - GT

Figure 5.9: Sweater - M

Figure 5.10: Sweater - GS

5.2.2 Transparent objects

This test, when evaluated considering the mesh-based rendering workflow, delivered largely expected results: both the glass and the water jug failed to reconstruct, while

the reading glasses were partially captured but appeared flattened and fused onto the subject's face, lacking a proper 3D structure.

Figure 5.11: Reading glasses - M

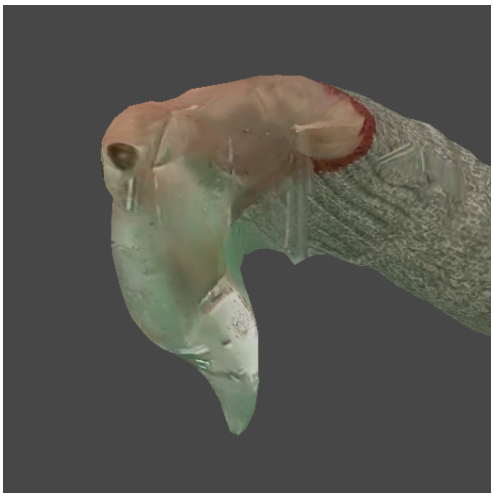

Figure 5.12: Water jug - M

In contrast, the Gaussian-based rendering handled the translucent objects considerably better. All three items appear well-defined when viewed from a moderate distance, with shapes that closely resemble the ground truth. However, when observed at closer range, particularly the jug and the glass, some blurriness becomes noticeable: this is due to the visible splats composing their structure. This effect is more pronounced depending on the viewing direction, and varies based on how close the point of view is to one of the original input cameras.

Figure 5.13: Water jug, near - GS

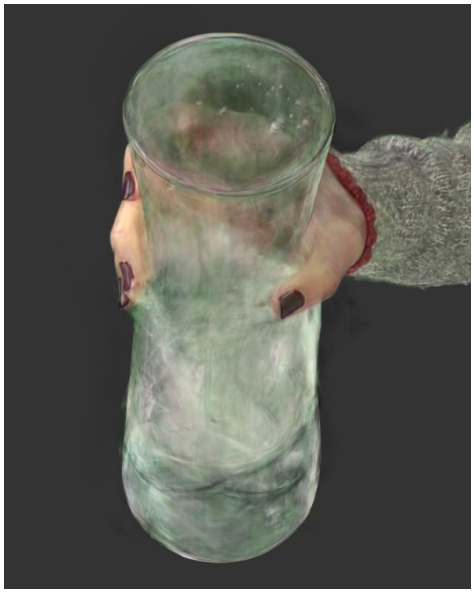

Figure 5.14: Water jug, far - GS

The jug also appears more opaque than expected. This is because the scene shown in Figures 5.13 and 5.14 was trained using masked input images, which included some background information in the transparent areas. This background color becomes embedded in the reconstruction, especially in transparent regions such as the jug.

To achieve a more realistic translucent effect, closer to the ground truth, the full images can be used during training, removing the background afterwards. While this approach increases training time significantly, it improves the transparency of the reconstructed objects. Nonetheless, the inherent blurriness introduced by the splats remains visible, and some minor color artifacts can be observed in the output.

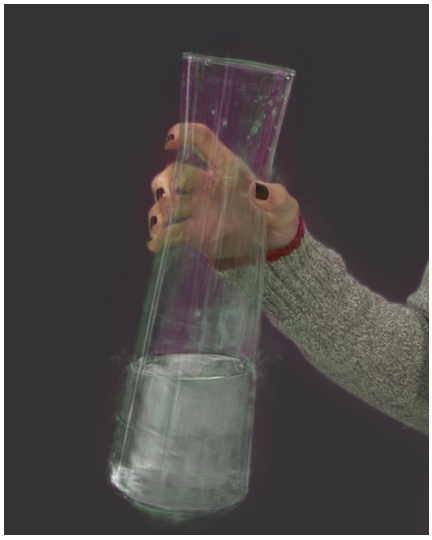

Figure 5.15: Water jug, background removed after training - GS

5.2.3 Thin objects

The results of this test were particularly interesting. The baseball cap was consistently reconstructed with high accuracy in each frame using the mesh-based approach, an outcome that was not entirely anticipated. This is likely due to the visor's rigidity and sufficient thickness, which allowed for effective feature point detection and depth estimation, enabling an accurate reconstruction. The paper brochure, on the other hand, was only partially reconstructed, excluding angles and other parts of the thin paper structure.

Figure 5.16: Cap - M

Figure 5.17: Paper - M

An analysis of the Gaussian-based output, on the other hand, demonstrated a complete and accurate reconstruction of both objects:

Figure 5.18: Cap - GS

Figure 5.19: Paper - GS

The paper brochure [Figure 5.19] is fully reconstructed, even in areas where the mesh-based method struggled significantly. The same applies to the baseball cap [Figure 5.18], which provided good results with both reconstruction techniques. However, when viewed from certain angles, the cap exhibits slight translucency issues. This is likely due to insufficient splat density in specific regions or opacity values that are too low, making parts of the cap partially see-through. This effect is particularly noticeable when observing the cap's visor from above, where the underlying paper brochure becomes visible through the semi-translucent splats:

Figure 5.20: Translucency issues - GS

5.2.4 Reflective objects

In the case of mesh-based rendering [Figure 5.21], the chrome ball is reconstructed with poor accuracy, showing several missing regions and an uneven, distorted shape. This is because mesh-based reconstruction struggles to handle reflective surfaces, whose appearance varies significantly depending on the viewing angle, making consistent

geometry estimation particularly challenging.

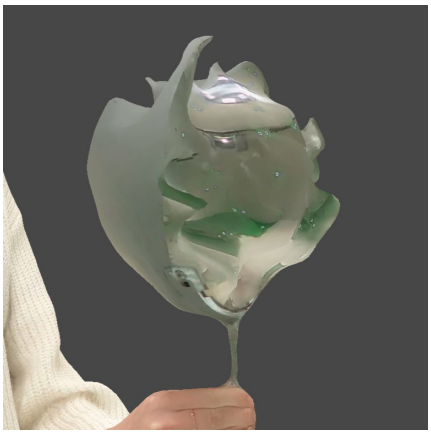

Figure 5.21: Chrome ball - M

Conversely, Gaussian Splatting demonstrated a greater ability to handle reflections, successfully reconstructing not only the chrome ball itself but also the reflected studio within it [Figure 5.22]. However, as previously noted, Gaussian-based scenes are inherently view-dependent, meaning the quality of the reconstruction varies based on the proximity of the viewing angle to one of the input cameras. This limitation is particularly evident in this scenario, where the clarity and accuracy of the reflection inside the chrome ball fluctuate significantly between viewpoints, even with minimal shifts in camera position.

Figure 5.22: Chrome ball, near - GS

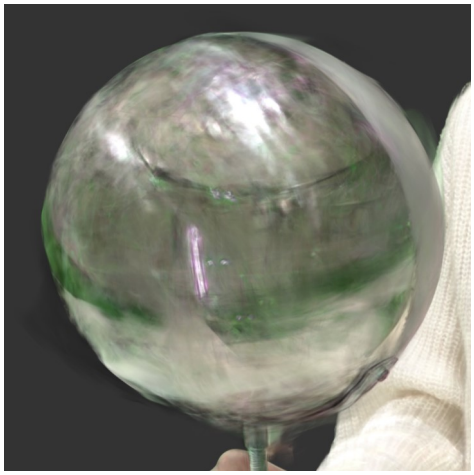

Figure 5.23: Chrome ball, far - GS

5.2.5 Loose Hair

In the mesh-based result [Figure 5.24], the reconstructed hair appears, as expected, quite unrealistic: the loose strands are rendered in a blocky manner, while the finer ones fail to appear at all, having been lost during processing. Much of the intricate hair detail is simply not captured by the mesh. In contrast, the Gaussian-based result [Figure 5.25] preserves these finer details significantly better. This is due to

the nature of Gaussians, which can represent even subtle structures and support semi-transparent areas, allowing for loose strands to be more accurately reconstructed.

Figure 5.24: Hair detail - M

Figure 5.25: Hair detail - GS

Using masked images, however, can easily lead to background artifacts close to the hair, which result hard to remove manually. For better results, it would be preferable to train on full images and remove the background afterwards. Despite the higher processing cost, this approach provides considerably more precise and artifact-free results.

5.2.6 Dark and Uniformly-Textured Outfits

Out of all the complex scenarios tested, the dark and uniformly-textured outfit produced the most visually successful result with mesh-based rendering. The subject was almost entirely reconstructed in every frame, with the exception of a few instances where parts of the left arm were missing [Figure 5.26].

Figure 5.26: Arm - M

However, the main limitation of this outfit emerges when viewing the full mesh-reconstructed take, which suffers from significant flickering artifacts, especially on the pants and bodysuit. This is primarily due to depth estimation errors in low-texture areas, where feature matching becomes unreliable, leading to instability in the reconstructed geometry over time. While the Cleaning and Tracking steps in *Volucap*'s pipeline typically help reduce such issues, in this case they were insufficient to fully correct the errors, indicating that depth information in those regions was notably less accurate than usual.

Figure 5.27: Pants flicker, f63 - M

Figure 5.28: Pants flicker, f64 - M

The Gaussian-based result, by contrast, avoids these artifacts, offering more accurate depth estimation and scene reconstruction. However, upon closer inspection, some imperfections can be seen on the clothing:

Figure 5.29: Bodysuit detail - GS

Figure 5.30: Pants detail - GS

This highlights that, although Gaussian Splatting demonstrates a clear advantage over mesh-based reconstruction in many challenging scenarios, it is still not immune to limitations. In particular, clothing with low texture or uniform color, such as the tested outfit, can still present issues. These materials often lack distinct visual features and even Gaussian Splatting may struggle to reconstruct such regions with perfect fidelity. This is especially true when viewing angles deviate significantly from the original input cameras, where the quality of the reconstruction may visibly drop due to view-dependency and sparse angular coverage.

In conclusion, the visual tests conducted have clearly demonstrated that Gaussian-based reconstruction offers a significant visual improvement over traditional mesh-based methods, particularly when it comes to handling complex and problematic scenarios. While meshes are limited by their reliance on explicit surface connectivity and accurate depth maps, which frequently break down when dealing with transparent, reflective, or finely detailed structures, Gaussians provide a more adaptable alternative. Their unstructured and differentiable nature, combined with properties such as view-dependent opacity and color, allows them to effectively reconstruct intricate geometry and subtle details that meshes often fail to capture. Although not free of challenges, Gaussian Splatting significantly broadens the range of scenarios that can be accurately reconstructed, offering a promising alternative for advancing volumetric capture technology.

Chapter 6

Conclusion

This thesis aimed to explore the potential of Dynamic Gaussian Splatting as an alternative rendering technique to traditional mesh-based reconstruction within *Volucap*'s volumetric capture workflow. The main goal was to investigate whether Gaussian Splatting could resolve long-standing limitations of mesh-based reconstruction, particularly in challenging scenarios that commonly affect volumetric productions involving human avatars for XR experiences.
To achieve this, a frame-by-frame Gaussian Splatting pipeline was developed and successfully integrated into *Volucap*'s existing mesh-based production workflow. This was a significant technical achievement, requiring the adaptation of external camera calibration data and compatibility with high-resolution input. All experiments were conducted using real footage captured at *Volucap*'s studio, strictly following the company's internal standards and leveraging their calibration system and image processing tools.

The primary focus of the study was on complex scenarios where mesh-based reconstruction typically fails or underperforms, such as transparent objects, reflective surfaces, thin items, loose hair, and dark or uniformly textured clothing. To evaluate the performance and capabilities of a Gaussian-based approach in complex scenarios, a dedicated test was conducted. In this test, all the difficult conditions that usually pose challenges for standard mesh-based methods were captured, processed, and assessed using both reconstruction approaches.

The results of the practical tests conducted demonstrated that Gaussian-based reconstruction offers significant visual improvements over traditional mesh-based methods, especially when it comes to handling complex and problematic scenarios. While meshes remain limited by their dependency on explicit surface connectivity and accurate depth maps, which often fail to accurately reproduce problematic scenarios, Gaussian Splatting provides a more versatile alternative. Thanks to its unstructured nature and view-dependent properties, along with the ability to encode opacity and color per Gaussian primitive, Gaussian Splatting managed to preserve finer details and produce more complete reconstructions, delivering higher visual fidelity. This tech-

nique significantly broadens the range of scenarios that can be accurately reproduced, offering a promising alternative for the advancement of volumetric capture technology.

However, this research also revealed major limitations. Firstly, it's necessary to consider the very extended training times per frame, which remain considerably longer than the processing times of mesh-based reconstruction, especially when aiming to meet a certain quality threshold. Moreover, the need to train each frame independently, without any temporal continuity, results in a linear increase in training time that quickly becomes unmanageable for longer sequences. Secondly, another critical limitation is the output file size: when trained at quality levels acceptable for commercial use, Gaussian-based scenes result significantly larger than their mesh-based counterparts, leading to unsustainable storage demands for full-length volumetric videos.
Due to these limitations, the frame-by-frame Gaussian Splatting approach is currently not applicable in a real-world production pipeline.

To sum up, while frame-by-frame Gaussian Splatting demonstrates strong reconstruction capabilities, particularly in challenging scenarios, it becomes clear that this implementation is not yet scalable for full dynamic scene reconstruction. Although training time and output size can technically be reduced by adjusting training parameters, this comes at the cost of visual fidelity, making the results unsuitable for commercial use. However, for static scenes, where longer training times and larger files are more acceptable, it remains a viable solution. Despite these drawbacks, the study highlights Dynamic Gaussian Splatting as a promising and effective alternative to overcome key mesh-based reconstruction issues. Its integration into *Volucap*'s pipeline confirms its potential as a valuable direction for future research and development.

We identified a promising approach for future advancements in a paper proposed in the SIGGRAPH Asia 2024, *Representing Long Volumetric Video with Temporal Gaussian Hierarchy* [43]. This research introduces a novel approach for efficiently reconstructing and rendering longer dynamic scenes, addressing one of the main limitations of recent dynamic view synthesis methods, which are typically constrained to short sequences of about 1–2 seconds. This new approach is able to successfully model and render minutes-long volumetric videos, while maintaining state-of-the-art quality, low storage requirements and efficient training. To efficiently model long volumetric videos, this method introduces the Temporal Gaussian Hierarchy, a multi-level structure of 4D Gaussian primitives. Each level delineates different regions of the scene based on how much of their content changes over time, and Gaussian primitives are adaptively shared across temporal segments for areas that remain static. This strategy significantly reduces the overall number of primitives required. One of the main strengths of this representation is its efficiency during training and rendering, which remains consistent regardless of the video length. Moreover, thanks

to its tree-like structure, the representation can efficiently identify the relevant segments for any given moment in time, loading only the necessary segments into GPU memory during training or rendering. This strategy ensures near-constant runtime memory usage and prevents scalability issues when working with longer sequences. To further minimize model size, a Compact Appearance Model is introduced: this combines diffuse and view-dependent Gaussians, which helps to minimize the model size while maintaining the rendering quality.[43]

In this context, integrating *Long Volumetric Video* [43] or similar methods into *Volucap*'s pipeline could provide the solution to combine the visual advantages of Gaussian Splatting with the scalability and efficiency needed for commercial-level dynamic volumetric productions. Unfortunately, this implementation is not yet publicly available, which prevented us from testing it during the course of this research.

In conclusion, this thesis demonstrates that Gaussian Splatting is capable of solving critical challenges in mesh-based reconstruction, offering a higher level of detail and realism in complex capture scenarios. While the tested frame-by-frame method is not yet ready for production use, emerging technologies offer a promising path forward. With further research and continued innovation, these new approaches may soon make it possible to implement Gaussian-based rendering techniques at scale, reshaping the future of volumetric video production.

Appendix A

Single-frame training times

The following section contains the tables used for the quantitative analysis. Specifically, they present the training times associated with different input image resolutions. This data was subsequently mapped into the graphs shown in [Section 5.1.1]. Each table reports the time required to train a single frame, along with the corresponding input resolution. Image resolution is expressed in pixels, and Training time is given in minutes. Each table refers to a distinct capture scenario.

Standard capture scenario [Section 4.2.1]:

Input resolution (pixels)	**Training time (minutes)**
1K	13
2K	17
4K	29
6K	50

Table A.1: Training times for Standard capture scenario

Transparent objects [Section 4.2.2]:

Input resolution (pixels)	**Training time (minutes)**
1K	14
2K	18
4K	32
6K	50

Table A.2: Training times for Transparent objects scenario

Dark and Uniformly-textured outfits [Section 4.2.6]:

Input resolution (pixels)	**Training time (minutes)**
1K	8
2K	11
4K	22
6K	38

Table A.3: Training times for Dark and Uniformly-textured outfits scenario

Appendix B

Single-frame output file sizes

The following section contains the tables used for the quantitative analysis. Specifically, they present the output file size associated with different values of the *Max Splat Count* parameter, which represents the number of Splat primitives that training will at most create in the scene. These tables were subsequently mapped into the graphs shown in [Section 5.1.2]. Each table shows how the output file size varies in regards to the maximum number of splats present in a scene. As we can observe, the two presented tables are identical, indicating that this parameter alone determines the output file size, regardless of the specific scene being reconstructed.

Standard capture scenario [Section 4.2.1]:

Max Splat Count (kSplats)	Output file size (MB)
500	118
1000	236
2000	473
3000	709

Table B.1: Output file sizes for Standard capture scenario

Thin objects scenario [Section 4.2.3]:

Max Splat Count (kSplats)	Output file size (MB)
500	118
1000	236
2000	473
3000	709

Table B.2: Output file sizes for Thin objects scenario

List of Figures

List of Tables

Acronyms

DoF	Degrees of Freedom.
AR	Augmented Reality.
VR	Virtual Reality.
HMD	Head-Mounted Display.
IMU	Inertial Measurement Unit.
XR	Extended Reality.
3DHBR	3D Human Body Reconstruction.
SfM	Structure-from-Motion.
MVS	Multi-View Stereo.
SPSR	Screened Poisson Surface Reconstruction.
NeRFs	Neural Radiance Fields.
MLP	Multi-Layer Perceptron.
3D-GS	3D Gaussian Splatting.
SH	Spherical Harmonics.
CNN	Convolutional Neural Networks.
SGD	Stochastic Gradient Descent.
PDF	Probability Density Function.

4D-GS	4D Gaussian Splatting.
FPS	Frames Per Second.
GT	Ground Truth.
M	Mesh.
GS	Gaussian Splatting.
S	Standard capture scenario.
TR	Transparent objects.
R	Reflective objects.
TH	Thin objects.
H	Loose Hair.
DU	Dark and Uniformly-textured outfits.

Bibliography

[1] Yili Jin, Kaiyuan Hu, Junhua Liu, Fangxin Wang, and Xue Liu. *From Capture to Display: A Survey on Volumetric Video*. 2024. arXiv: 2309.05658 [cs.MM]. URL: https://arxiv.org/abs/2309.05658 (cit. on p. 1).

[2] Jeroen van der Hooft, Hadi Amirpour, Maria Torres Vega, Yago Sanchez, Raimund Schatz, Thomas Schierl, and Christian Timmerer. "A Tutorial on Immersive Video Delivery: From Omnidirectional Video to Holography". In: *IEEE Communications Surveys & Tutorials* 25.2 (2023), pp. 1336–1375. DOI: 10.1109/COMST.2023.3263252 (cit. on pp. 1, 2).

[3] Arcturus. *What is Volumetric Video: A Beginner's Guide — arcturus.studio*. https://arcturus.studio/blog/what-is-volumetric-video/. [Accessed 15-02-2025] (cit. on p. 1).

[4] Zhengren Wang. *3D Representation Methods: A Survey*. 2024. arXiv: 2410.06475 [cs.CV]. URL: https://arxiv.org/abs/2410.06475 (cit. on pp. 3, 19).

[5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. *3D Gaussian Splatting for Real-Time Radiance Field Rendering*. 2023. arXiv: 2308.04079 [cs.GR]. URL: https://arxiv.org/abs/2308.04079 (cit. on pp. 4, 23–28, 30, 34).

[6] Oliver Schreer, Ingo Feldmann, Peter Kauff, Peter Eisert, Danny Tatzelt, Cornelius Hellge, Karsten Muller, Sven Bliedung, and Thomas Ebner. "Lessons Learned During One year of Commercial Volumetric Video Production". In: *SMPTE Motion Imaging Journal* 129 (Oct. 2020), pp. 31–37. DOI: 10.5594/JMI.2020.3010399 (cit. on pp. 7–9, 17, 18).

[7] Basler AG. *CMOS Global Shutter Cameras — baslerweb.com*. https://www.baslerweb.com/en/learning/cmos-global-shutter-cameras/. [Accessed 15-02-2025] (cit. on p. 8).

[8] Oliver Schreer, Ingo Feldmann, Sylvain Renault, Marcus Zepp, Markus Worchel, Peter Eisert, and Peter Kauff. "Capture and 3D Video Processing of Volumetric Video". In: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 4310–4314. DOI: 10.1109/ICIP.2019.8803576 (cit. on pp. 8, 14, 15).

[9] Basler AG. *Flat-Field Correction | Basler Product Documentation — docs.baslerweb.com*. https://docs.baslerweb.com/flat-field-correction. [Accessed 15-02-2025] (cit. on p. 8).

[10] Inc. The MathWorks. *What Is Camera Calibration?* https://de.mathworks.com/help/vision/ug/camera-calibration.html. [Accessed 15-02-2025] (cit. on pp. 9, 10).

[11] Fraunhofer Heinrich-Hertz-Institut. *Foreground/Background Separation — hhi.fraunhofer.de*. https://www.hhi.fraunhofer.de/en/vit-imc/research-topics/foreground-background-separation.html. [Accessed 15-02-2025] (cit. on p. 10).

[12] Johannes L. Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4104–4113. DOI: 10.1109/CVPR.2016.445 (cit. on p. 14).

[13] M. Goesele, B. Curless, and S.M. Seitz. "Multi-View Stereo Revisited". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 2402–2409. DOI: 10.1109/CVPR.2006.199 (cit. on pp. 14, 23).

[14] Wolfgang Waizenegger, Ingo Feldmann, and Oliver Schreer. "Real-time Patch Sweeping for High-Quality Depth Estimation in 3D Videoconferencing Applications". In: *Proceedings of SPIE - The International Society for Optical Engineering* 7871 (Feb. 2011). DOI: 10.1117/12.872868 (cit. on p. 14).

[15] W. Waizenegger, I. Feldmann, O. Schreer, P. Kauff, and P. Eisert. "Real-time 3D body reconstruction for immersive TV". In: *2016 IEEE International Conference on Image Processing (ICIP)*. 2016, pp. 360–364. DOI: 10.1109/ICIP.2016.7532379 (cit. on p. 14).

[16] S. Ebel, W. Waizenegger, M. Reinhardt, O. Schreer, and I. Feldmann. "Visibility-driven patch group generation". In: *2014 International Conference on 3D Imaging (IC3D)*. 2014, pp. 1–8. DOI: 10.1109/IC3D.2014.7032597 (cit. on p. 14).

[17] Michael Kazhdan and Hugues Hoppe. "Screened poisson surface reconstruction". In: *ACM Trans. Graph.* 32.3 (July 2013). ISSN: 0730-0301. DOI: 10.1145/2487228.2487237. URL: https://doi.org/10.1145/2487228.2487237 (cit. on p. 15).

[18] Michael Garland and Paul S. Heckbert. "Surface Simplification Using Quadric Error Metrics". In: *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 1st ed. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9798400708978. URL: https://doi.org/10.1145/3596711.3596727 (cit. on p. 15).

[19] Ziyan Wang, Giljoo Nam, Aljaz Bozic, Chen Cao, Jason Saragih, Michael Zollhoefer, and Jessica Hodgins. *A Local Appearance Model for Volumetric Capture of Diverse Hairstyle*. 2023. arXiv: 2312.08679 [cs.CV]. URL: https://arxiv.org/abs/2312.08679 (cit. on p. 19).

[20] Yiming Qian, Minglun Gong, and Yee-Hong Yang. "3D Reconstruction of Transparent Objects with Position-Normal Consistency". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4369–4377. DOI: 10.1109/CVPR.2016.473 (cit. on p. 20).

[21] Benjamin Ummenhofer and Thomas Brox. "Point-Based 3D Reconstruction of Thin Objects". In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 969–976. DOI: 10.1109/ICCV.2013.124 (cit. on p. 20).

[22] Xi Deng, Lifan Wu, Bruce Walter, Ravi Ramamoorthi, Eugene d'Eon, Steve Marschner, and Andrea Weidlich. "Reconstructing translucent thin objects from photos". In: *SIGGRAPH Asia 2024 Conference Papers*. SA '24. Association for Computing Machinery, 2024. ISBN: 9798400711312. DOI: 10.1145/3680528.3687572. URL: https://doi.org/10.1145/3680528.3687572 (cit. on p. 21).

[23] Ali Hosseininaveh Ahmadabadian, R. Yazdan, A. Karami, M. Moradi, and F. Ghorbani. "Clustering and selecting vantage images in a low-cost system for 3D reconstruction of texture-less objects". In: *Measurement* 99 (2017), pp. 185–191. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2016.12.026. URL: https://www.sciencedirect.com/science/article/pii/S0263224116307242 (cit. on p. 21).

[24] Fangjinhua Wang, Marie-Julie Rakotosaona, Michael Niemeyer, Richard Szeliski, Marc Pollefeys, and Federico Tombari. *UniSDF: Unifying Neural Representations for High-Fidelity 3D Reconstruction of Complex Scenes with Reflections*. 2024. arXiv: 2312.13285 [cs.CV]. URL: https://arxiv.org/abs/2312.13285 (cit. on p. 21).

[25] Noah Snavely, Steven M. Seitz, and Richard Szeliski. "Photo tourism: exploring photo collections in 3D". In: *ACM Trans. Graph.* 25.3 (July 2006), pp. 835–846. ISSN: 0730-0301. DOI: 10.1145/1141911.1141964. URL: https://doi.org/10.1145/1141911.1141964 (cit. on pp. 23, 34).

[26] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: 2003.08934 [cs.CV]. URL: https://arxiv.org/abs/2003.08934 (cit. on p. 23).

[27] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. "Plenoxels: Radiance Fields without Neural Networks". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 5491–5500. DOI: 10.1109/CVPR52688.2022.00542 (cit. on p. 23).

[28] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. "Instant neural graphics primitives with a multiresolution hash encoding". In: *ACM Transactions on Graphics* 41.4 (July 2022), pp. 1–15. ISSN: 1557-7368. DOI: 10.1145/3528223.3530127. URL: http://dx.doi.org/10.1145/3528223.3530127 (cit. on p. 23).

[29] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. *Point-NeRF: Point-based Neural Radiance Fields*. 2023. arXiv: 2201.08845 [cs.CV]. URL: https://arxiv.org/abs/2201.08845 (cit. on p. 23).

[30] M. Botsch, A. Hornung, M. Zwicker, and L. Kobbelt. "High-quality surface splatting on today's GPUs". In: *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.* 2005, pp. 17–141. DOI: 10.1109/PBG.2005.194059 (cit. on p. 24).

[31] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. "Surfels: surface elements as rendering primitives". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 335–342. ISBN: 1581132085. DOI: 10.1145/344779.344936. URL: https://doi.org/10.1145/344779.344936 (cit. on p. 24).

[32] Liu Ren, Hanspeter Pfister, and Matthias Zwicker. "Object Space EWA Surface Splatting: A Hardware Accelerated Approach to High Quality Point Rendering". In: *Computer Graphics Forum* 21.3 (2002), pp. 461–470. DOI: https://doi.org/10.1111/1467-8659.00606. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00606. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00606 (cit. on p. 24).

[33] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. "Surface splatting". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: Association for Computing Machinery, 2001, pp. 371–378. ISBN: 158113374X. DOI: 10.1145/383259.383300. URL: https://doi.org/10.1145/383259.383300 (cit. on p. 24).

[34] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. "Neural Point-Based Graphics". In: *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII*. Glasgow, United Kingdom: Springer-Verlag, 2020, pp. 696–712. ISBN: 978-3-030-58541-9. DOI: 10.1007/978-3-030-58542-6_42. URL: https://doi.org/10.1007/978-3-030-58542-6_42 (cit. on p. 24).

[35] Darius Rückert, Linus Franke, and Marc Stamminger. "ADOP: approximate differentiable one-pixel point rendering". In: *ACM Trans. Graph.* 41.4 (July 2022). ISSN: 0730-0301. DOI: 10.1145/3528223.3530122. URL: https://doi.org/10.1145/3528223.3530122 (cit. on p. 24).

[36] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. "EWA volume splatting". In: *Proceedings Visualization, 2001. VIS '01.* 2001, pp. 29–538. DOI: 10.1109/VISUAL.2001.964490 (cit. on p. 25).

[37] Christoph Lassner. "Fast Differentiable Raycasting for Neural Rendering using Sphere-based Representations". In: *CoRR* abs/2004.07484 (2020). arXiv: 2004.07484. URL: https://arxiv.org/abs/2004.07484 (cit. on pp. 27, 28).

[38] Duane G. Merrill and Andrew S. Grimshaw. "Revisiting sorting for GPGPU stream architectures". In: *Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques*. PACT '10. Vienna, Austria: Association for Computing Machinery, 2010, pp. 545–546. ISBN: 9781450301787. DOI: 10.1145/1854273.1854344. URL: https://doi.org/10.1145/1854273.1854344 (cit. on p. 28).

[39] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. *4D Gaussian Splatting for Real-Time Dynamic Scene Rendering*. 2024. arXiv: 2310.08528 [cs.CV]. URL: https://arxiv.org/abs/2310.08528 (cit. on pp. 29–33, 42, 45).

[40] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. *K-Planes: Explicit Radiance Fields in Space, Time, and Appearance*. 2023. arXiv: 2301.10241 [cs.CV]. URL: https://arxiv.org/abs/2301.10241 (cit. on p. 31).

[41] *Output Format; COLMAP 3.12.0.dev0 documentation — colmap.github.io*. [Accessed 12-03-2025]. URL: https://colmap.github.io/format.html#text-format (cit. on p. 37).

[42] *Training Configuration — jawset.com*. https://www.jawset.com/docs/d/Postshot+User+Guide/Interface/Training+Configuration. [Accessed 15-03-2025] (cit. on p. 40).

[43] Zhen Xu, Yinghao Xu, Zhiyuan Yu, Sida Peng, Jiaming Sun, Hujun Bao, and Xiaowei Zhou. "Representing Long Volumetric Video with Temporal Gaussian Hierarchy". In: *ACM Transactions on Graphics* 43.6 (Nov. 2024), pp. 1–18. ISSN: 1557-7368. DOI: 10.1145/3687919. URL: http://dx.doi.org/10.1145/3687919 (cit. on pp. 70, 71).

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to my academic supervisors, Andrea Bottino and Francesco Strada, for their continuous guidance and support throughout the course of this research. Their expertise and encouragement have been essential to the development and completion of this thesis.

My sincere appreciation goes to my company supervisor, Sven Bliedung von der Heide, and to the entire *Volucap* team, for welcoming me into their innovative environment and providing me with the tools to grow both professionally and personally during my months abroad. Their support and collaboration were pivotal in shaping the practical aspects of this work.

Finally, I would like to extend my deepest thanks to my family and friends. Their steady support, patience, and belief in me have been a constant source of strength. Without their love and encouragement, this journey would not have been possible.