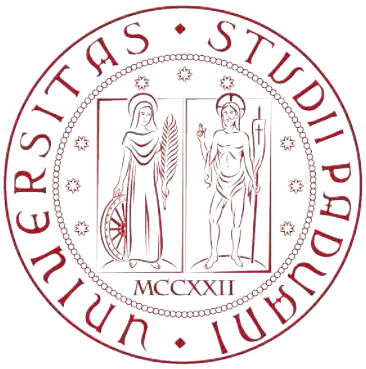

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Degree in ICT for Internet and Multimedia

Title

Gaussian Splatting for Enhanced Scene Representation in 5G-Enabled Virtual Reality for Remote Operations

Supervisor:
Prof.ssa Federica Battisti

Candidate:
Japheth Bidayan Duut
2080652

Academic Year 2023-2024

”I just wondered how things were put together.”

Claude Shannon

Abstract

Scene reconstruction in virtual reality (VR) applications presents numerous challenges, particularly in scenarios requiring remote communication, such as with remotely operated vehicles (ROVs). As a result, there have been a number of proposed solutions for addressing these concerns more effectively and efficiently. Most of these techniques for representing 3D environments often struggle with training and rendering speed, accuracy, and the ability to handle complex, dynamic environments, which limits their effectiveness in time-sensitive fields which require remote operations. This thesis focuses on the application of the 3D Gaussian splatting technique for real-time radiance field rendering aiming to improve the efficiency and fidelity of 3D scene representation for remote operations. The ROV is equipped with high-resolution cameras to capture detailed visual data, facilitating faster and more accurate reconstruction of remote environments. Gaussian splatting allows for smoother transitions in scene representation, making it ideal for real-time applications where both speed and visual quality are crucial. This work explores the deployment of Gaussian splatting to enhance scene reconstruction in remote settings. Additionally, a framework for integrating Gaussian splatting with 5G technology, leveraging high-speed, low-latency communication to improve overall operational effectiveness and user experience in remote and critical scenarios, is proposed.

Keywords: Gaussian Splatting, Scene Reconstruction, 3D Rendering, Virtual Reality, 5G Technology

Astratto

La ricostruzione della scena nelle applicazioni di realtà virtuale (VR) presenta numerose sfide, in particolare negli scenari che richiedono una comunicazione a distanza, come nel caso dei veicoli a comando remoto (ROV). Di conseguenza, sono state proposte diverse soluzioni per affrontare questi problemi in modo più efficace ed efficiente. La maggior parte di queste tecniche per la rappresentazione di ambienti 3D spesso si scontra con la velocità di formazione e rendering, l'accuratezza e la capacità di gestire ambienti complessi e dinamici, il che limita la loro efficacia in campi sensibili al tempo che richiedono operazioni a distanza. Questa tesi si concentra sull'applicazione della tecnica di splatting gaussiano 3D per il rendering in tempo reale del campo di radianza, con l'obiettivo di migliorare l'efficienza e la fedeltà della rappresentazione della scena 3D per le operazioni a distanza. Il ROV è dotato di telecamere ad alta risoluzione per acquisire dati visivi dettagliati, facilitando una ricostruzione più rapida e accurata degli ambienti remoti. Lo splatting gaussiano consente transizioni più fluide nella rappresentazione della scena, rendendolo ideale per le applicazioni in tempo reale in cui sono fondamentali sia la velocità che la qualità visiva. Questo lavoro esplora l'impiego dello splatting gaussiano per migliorare la ricostruzione della scena in ambienti remoti. Inoltre, viene proposto un framework per integrare lo splatting gaussiano con la tecnologia 5G, sfruttando la comunicazione ad alta velocità e bassa latenza per migliorare l'efficacia operativa complessiva e l'esperienza dell'utente in scenari remoti e critici.

Parole chiave: Gaussian Splatting, Ricostruzione della scena, Rendering 3D, Realtà Virtuale, Tecnologia 5G

Acknowledgements

I would like to express my sincere gratitude to everyone who has supported me throughout the journey of completing this thesis. Their guidance, encouragement, and patience have been invaluable in bringing this work to fruition.

First and foremost, my deepest gratitude goes to Professor Federica Battisti for her invaluable guidance, patience, and support. Her insightful perspectives, constructive feedback, and expertise have been instrumental in shaping this work, from the early stages of research to the final completion. Her commitment to excellence and attention to detail have inspired me throughout this project, and I am truly fortunate to have had her as my advisor.

Additionally, I would like to extend my heartfelt thanks to my family and friends, who have been a constant source of support and encouragement. Their unwavering belief in me, even during challenging moments, has been a source of strength and motivation. I am grateful for their understanding and patience, as they have been with me every step of the way, offering reassurance and inspiration.

I am also grateful to my colleagues and fellow students at the University of Padua, whose camaraderie, advice, and encouragement have made this journey both enjoyable and fulfilling. Their willingness to share ideas, discuss challenges, and celebrate milestones together has enriched my academic experience, and I am fortunate to have been part of such a supportive community.

Finally, to everyone who has contributed in any way to the successful completion of this thesis, whether through guidance, friendship, or inspiration, I extend my sincere thanks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Virtual reality (VR) has proven, in the past few years, to be transformative in providing immersive experiences especially in the field of remote operations. From industrial automation to medical procedures, VR has been integrated into many real world sectors and industries to provide users with visual feedback, spatial awareness and precise control of systems in remote or rather hazardous environments. Hence, one of the most promising applications of VR lies in the control and management of remotely operated vehicles (ROV) such as ground rovers or unmanned aerial vehicles (UAVs) and their use in tasks such as infrastructure inspection, search and rescue and environmental monitoring.

The progress in the field of virtual reality has not eliminated all the challenges that persist in the field. In spite of these advancements in VR and remote operations technologies, the main challenge of accurately reconstructing and rendering dynamic 3D environments in real time still remains. Most traditional 3D scene representation techniques, such as mesh-based or point cloud representations, often struggle with the complexities of scene reconstruction when the environment is dynamic and fast-changing, as required in real-time, critical applications. This reality presents an important issue related to training speed, rendering accuracy and computational overhead, which is then worsened by the latency and bandwidth limitations of traditional communication systems. In operations that are particularly time sensitive, such as those requiring the control of ROVs in hazardous environments, these issues pose a major constraint on the effectiveness of remote operations.

To tackle these challenges, the use of 3D Gaussian splatting for real-time rendering can be employed in these VR systems. This will enable smoother transitions and greater efficiency in

handling complex and dynamic environments. Due to the demanding requirements of VR applications in terms of speed, accuracy and visual fidelity requirement, this rendering technique excels. Alongside this, the 5G network provides a great opportunity for enabling high-speed, low-latency communication between remote environments and operators, thus greatly improving the feasibility of real-time 3D scene reconstruction and visualization.

1.2 Problem Statement

To be effective, VR systems for remote systems, especially those involving ROVs depend on the ability to capture, process and render complex scenes. Traditional scene representation techniques often struggle in three main domains: slow training and rendering speeds which impede performance, poor accuracy particularly in dynamic environments. This thesis addresses the above mentioned issues by focusing on the integration of 3D gaussian splatting as the scene representation model with 5G technology to enhance the performance of VR applications in remote applications. The main focus of this is implement a pipeline that uses Gaussian splatting to recontruct a scene and prepare it for rendering in applications developed for head-mounted displays (HMD) in order to improve the fidelity and efficiency of scene reconstruction for ROVs equipped with a number of data capture tools like a high resolution camera.

1.3 Scope

This thesis focuses on developing a pipeline for scene reconstruction in VR applications designed for remotely operated vehicles (ROVs). Specifically, the project integrates the use of 3D Gaussian splatting as a scene representation technique to address the challenges of rendering speed, accuracy, and fidelity in complex environments.

The scope of this work includes:

1. **Data Acquisition and Processing**: Using high-resolution cameras and other data capture tools to collect images of scenes for ROV applications, with an emphasis on effective data preprocessing techniques to prepare for 3D Gaussian splatting.

2. **Model Training and Scene Reconstruction**: Implementing and optimizing the Gaussian splatting model for scene reconstruction, focusing on achieving high visual fidelity and efficient training suitable for real-time VR use.

3. **Pipeline Integration with HMDs**: Developing a pipeline that prepares the reconstructed scene for rendering on head-mounted displays (HMDs), particularly the Meta Quest 2, using the Gracia VR application. This stage also examines the integration with 5G technology to enhance data transmission efficiency.

This thesis focuses on leveraging existing tools and technologies to create an optimized pipeline.

1.4 Proposed Approach

The proposed approach to this work is done by training an implementation of the 3D Gaussian Splatting system and rendering in a head-mounted display (HMD). Since the focus of this work is on enhancing 3D scene representation using 3D Gaussian splatting, a more immersive and interactive experience for remote operators which will enable them to make more accurate and informed decisions in real-time is sought. And as a result, a framework for integrating the rendering of Gaussian splats in virtual realilty applications, 5G technology and ROVs to improve the efficiency and the fidelity of 3D scene representation is proposed

3D gaussian splatting in VR for remote operations has several significant benefits. Due to its ability to adjust to different scenes, Gaussian splatting is great for dynamic environments. Not only that, there are benefits in better rendering efficiency presented. And the overall visual quality of the rendered scene is better than other methods.

In addition to all of this, this thesis proposes the integration of Gaussian splatting with 5G technology leading to high speed, low-latency communication while allowing faster high resolution visual data transmission captured by the ROVs and overall operational effectiveness resulting from lower latency and minimized delay in VR rendering.

1.5 Thesis Objectives

To achieve the main aim of this thesis which is to investigate the effectiveness of Gaussian splatting for 3D scene representation in remote operations, the following are the key objectives of this work:

1. To develop a streamlined pipeline that trains the 3D Gaussian splatting system for virtual reality applications.

2. To render the trained model in a virtual reality application for a head-mounted display (HMD)

3. To propose a framework for the integration of Gaussian splatting and 5G technology with an eye on operational effectiveness and user experience.

1.6 Thesis Contributions

This thesis presents a number of contributions key of which is proposing a framework for the integration of Gaussian splatting in VR and 5G technology in the broader system for remote operations. It achieves this through the following:

1. Providing a detailed analysis of the existing techniques currently used for 3D visual scene representation and comparing and contrasting them with the novel 3D Gaussian splatting technique.

2. An implementation of a streamlined pipeline to train and render scenes using 3D Gaussian splatting

3. Insights into visual quality of rendered scenes.

4. Providing a comprehensive framework for the integration of Gaussian splatting with a private 5G network for use in the ROV in remote operations. The components of this part includes the following:

    - Edge computing nodes and a local 5G transceiver for on-site data processing.

    - High resolution scene capture systems including high resolution cameras.

    - A telco cloud for offline VR model reconstruction, further enhancing the capabilities of remote operators within mission critical operations.

1.7 Thesis Structure

The remainder of this thesis is organized as follows:

Chapter 2: Related Works provides a thorough review of past works on existing 3D scene representation techniques. This includes traditional methods such as mesh-based and point cloud representations, and modern approaches like neural radiance fields (NeRF). Additionally, virtual reality technology is explored. This chapter also sheds light on the 3D Gaussian Splatting technique, provides a comparison with other techniques, and examines the role of 5G technology and ROVs in remote VR operations.

Chapter 3: Methodology outlines the developement environment setup, including data collection , data processing pipeline development, model training, and evaluation metrics. Additionally, this chapter also describes the process followed to render the trained scenes in VR applications in a head-mounted display.

Chapter 4: Results and Discussion presents a discussion of the results of the development process, comparing the displayed scenes with ground truth data and discussing the improvements brought about by the system.

Chapter 5: Conclusion concludes this work, summarizes the key findings of the work and provides recommendations for future work.

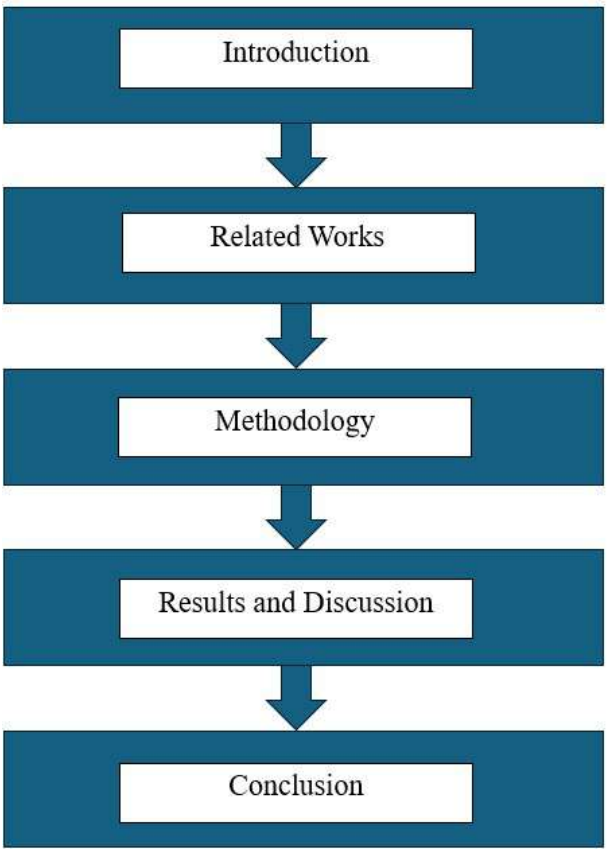

Figure 1.1: Organization of the thesis

Chapter 2

Related Works

2.1 3D Scene Representation

2.1.1 Introduction

Over the years, 3D scene representation has experienced tremendous evolution leading to an era of advanced technological advancements that have made applications such as virtual reality and augmented reality possible. 3D scene representation is a fundamental challenge in the field of computer vision and graphics. At its core, it aims to capture, model and reproduce the geometric and photometric properties of real-world environments in a digital format. Many applications rely on the efficient and accurate building of 3D representations which are crucial to ensuring great user experiences. Fields like remote sensing, autonomous navigation, and medical imaging benefit from these applications. Virtual reality applications often require accurate 3D scene representations to deliver realistic experiences for users [43]. Achieving these stringent goals require techniques that operate on input data to produce excellent reconstructed scenes captured by the input image data. While traditional methods such as photogrammetry, LiDAR scanning, and mesh-based modeling have been employed to achieve this, these techniques often face limitations in terms of processing speed, scalability and the ability to handle complex or dynamic environments.

3D scene representation is directly related to the primitives that is used to store scene information and hence the evolution of the field has followed that of the ways in which 3D data is stored [6]. That is to say, the early three-dimensional scene representation techniques used points, which led to a number of methods developed in subsequent decades that used meshes for data representation [44]. Voxels, which are volumetric analogous to pixels, were then developed which became crucial in scene representation. Multilayer perceptron (MLPs),

also known as neural networks became the state of the art a few years ago. And more recently gaussian splats have become the latest development in 3D data representation. These data representation primitives map directly to the methods and algorithms that are used to produce them and hence 3D scene representations.

Point clouds and meshes had trouble addressing dynamic changes in complicated situations and capturing minute details. Considering this, voxels were created to provide more detailed volumetric representations; nonetheless, their computing complexity remained a concern, particularly in high-resolution scenes. For continuous scene representation, smoother representations and novel view synthesis, Neural networks became the trend. Then came gaussian splats.

Gaussian splats offer a compact, efficient, and scalable approach to representing 3D scenes. This has led to faster computations and much better scene representations in terms of fidelity and smoothness. By approximating surfaces and radiance fields with Gaussian distributions, this technique allows for faster real-time rendering and better handling of both geometric and photometric properties. There are four main 3D scene representations and reconstruction techniques that we will cover in this work. Photogrammetry and structure from motion which maps to point clouds and meshes, Neural Radiance Fields (NeRFs) which maps to multilayer perceptrons and Gaussian splatting which maps to gaussian splats.

Traditional methods, such as polygonal meshes and voxel grids, often struggle with rendering speed, accuracy, and the ability to handle complex, dynamic scenes due to the explicit scene representation in voxels or grid points that are not continuous.

This drawback of traditional methods is what Neural methods such as NeRFs sought to address. NeRFs are great for novel view synthesis because they build on continuous scene representations leading to improved visual quality from traditional methods but face the major problem of requiring huge amounts of data and the long training times. These challenges in particular is what motivated the development of the 3D gaussian splatting technique for real-time rendering in [21].

2.2 Traditional Geometric Methods

2.2.1 Voxel Grid

Voxel Grids are a mesh of primitives called voxels that contain 3D information akin to pixels in 2D. Volumetric or voxel-based modeling represents a 3D object by dividing it into small,

uniform volume elements, or voxels. This modeling technique is becoming increasingly popular due to its capacity to capture highly complex, free-form shapes and accurately represent objects with varied internal structures or directional properties, offering a more realistic simulation of real-world objects. Voxel-based models are particularly well-suited for data obtained from natural acquisition techniques like CT or MRI scanning and easily accommodate Boolean operations for modeling. However, the main drawbacks include the need for substantial memory to reduce discretization errors and considerable processing power for rendering and analysis [32].

As volumetric pixels, voxels can capture and store detailed 3D information about objects and environments. This allows for more accurate rendering of complex geometries, realistic lighting and shading effects, and improved collision detection compared to traditional polygon-based 3D models. Voxel-based rendering is particularly well-suited for VR as it enables smooth, continuous transitions between views, reduces visual artifacts, and supports real-time interactions with virtual objects. While voxel data can require more storage space than polygon meshes, advances in hardware and compression techniques have made voxel-based VR experiences more practical and performant.

In computer graphics and digital media, voxels are increasingly used to create realistic visual effects and complex environments, especially in gaming and virtual simulations. Voxel-based models also enable Boolean operations like union, intersection, and subtraction, making it easier to modify objects, merge shapes, or create holes and other complex structures without complex recalculations. Voxel models come with challenges, primarily due to the high memory and computational requirements needed to store and process detailed voxel data. To accurately represent large or high-resolution models, voxel grids need to be highly dense, which can demand significant storage and processing power.

A voxel model typically represents data as a uniform 3D grid, where each cell's occupancy status is recorded. For each voxel, a binary value may be used to indicate whether material is present or absent, or a numeric value may be recorded to represent a specific physical property, such as density. This 3D array (also known as a volume buffer, cubic frame buffer, or 3D raster) aligns with the coordinate axes to store these values consistently across the grid [20].

2.2.2 Point Clouds

3D scene reconstruction techniques are marked by the primitive data structures that they use to represent aspects of a scene. In the case of point cloud, this is represented by 3D points in space. Point clouds are essential to the representation of external surfaces of objects and by extension the scenes in which they sit. And as the field of 3D graphics advances it has proven very useful in our understanding of three-dimensional scenes and objects. Point clouds can be described as a set of points arranged in space that represent information about the external surfaces of objects and scenes. Each point in the set contains information pertaining to the 3D spatial coordinates as well as information on color, intensity and normal vectors [43]. As technology has advanced, it has become increasingly easy to acquire point cloud data and has made them a ubiquitous form of three-dimensional data representation. They are mainly acquired through scanning technologies like LiDAR (Light Detection and Ranging) and 3D reconstruction from 2D images which make them easy to generate and work with than other methods of scene representation. Their inherent simplicity is an attractive point for work in scene representation. However, point clouds are often sparse, unstructured, and prone to noise, which complicates processing tasks such as segmentation, recognition, and reconstruction.

Acquisition Methods

A fundamental aspect of scene representation using point clouds is the acquisition of the point cloud data itself. This involves capturing the 3D data and includes spatial information about the objects or environments the points represent. There are a number of techniques and methods for acquiring point clouds, each of which is suited for different applications, depending on several factors. Light Detection and Ranging (LiDAR), photogrammetry, structured light scanning, and stereo vision are just a few of them.

Light Detection and Ranging (LiDAR)

LiDAR remains one of the most popular ways of capturing point cloud data in the 3D data representation field. In order to get surface information in the form of points, LiDAR works by shooting pulses of laser within its field of view and takes measurements at intervals of the time it takes for those pulses to reflect off the surfaces they hit onto the sensor. This measurement is then converted into distance data and then used to generate the 3D point cloud. The output of the LiDAR scans consists of two components. Firstly, the 3D point clouds which represents to the scanned environments and secondly, the reflected laser energy intensities [26].

Most LiDAR systems can be broken down into two major components: the laser rangefinder and the scanning system. The former is responsible for shooting the lasers onto surfaces and the later is more concerned with steering the laser beams at different azimuths [26]. While this technology is associated with superior ranging accuracy, it still struggles with capturing semantic information as well as sensors like high-resolution cameras do. However, in applications where distance estimation is essential, such as collecting point cloud, it is superior to other sensing technologies.

LiDAR output data, thus point clouds, is usually stored in a binary file format which in most cases is the las file format. This file format consists of three parts namely: the header, length record area and point set record area. The las format takes into account the characteristics of LIDAR data, the structure is reasonable, and it is easy to expand [46]. Efficient storage and retrieval of point cloud data are essential for processing. Common data structures include a simple list of points, which is straightforward but uses a lot of memory. Octrees are a hierarchical structure that partitions space for better organization, while K-d trees use binary partitioning to enable fast nearest neighbor searches. Voxel grids divide the data into regular 3D grids, making it easier to handle and process point cloud information.

Point clouds are useful in numerous applications including 3D modeling, self-driving cars and virtual and augmented reality. Virtual and Augmented Reality applications use point clouds to create a merging of the digital and physical worlds, creating immersive experiences. Additionally, in the medical field, point clouds have made significant inroads, enhancing medical imaging and analysis by providing detailed 3D visualizations of anatomical structures.

Despite all its usefulness and advantages, LiDAR equipment remains expensive and the data gathering process remains susceptible to environmental and weather conditions like rain, wind, and fog that can affect the accuracy of the data collected. Finally, the data collected requires sophisticated software for processing and handling large datasets.

Photogrammetry

Photogrammetry involves creating 3D point clouds from 2D stereo image pairs. It was pioneered in the 1980s and introduced the idea of identifying corresponding points in stereo image pairs by using stereo vision techniques [9]. This technique makes use of the principle of triangulation, where corresponding points on overlapping images are identified to compute their spatial coordinates. An important aspect of this technique is the acquisition of images by capturing pictures of a scene from different perspectives and using sophisticated processing software to match corresponding points, estimate the camera positions and ultimately gener-

ate the 3D point cloud data which in this case represents the model of the captured scene [6]. This enables the creation of photorealistic scenes.

The photogrammetry process typically starts with a camera, which captures multiple overlapping images of the object or scene from different angles. Then calibration tools are used to ensure that the camera's intrinsic parameters (like focal length and distortion) are known for accurate image alignment. After this capture process is complete, photogrammetry processing software is then used to process the images using algorithms that match common points in overlapping images and then reconstruct the 3D geometry. This essentially takes the process from capturing, common points matching, triangulation, and finally point cloud generation.

The accuracy of points generated by photogrammetry has led it to be applied in many fields. In architecture and construction, it is used for creating accurate 3D representations of buildings and infrastructure, aiding in design, documentation, and inspection. In aerial surveying and mapping, photogrammetry from drones or aircraft helps in terrain modeling, urban planning, and land-use analysis. Archaeology benefits from photogrammetry by documenting historical sites and artifacts in high detail without physical contact. In film and gaming, the technology is employed to create realistic 3D environments and characters.

Some of the attractive points of using photogrammetry for point cloud generation is its cost-effectiveness due to compatibility with the use of standard cameras. In this sense there is no need to acquire expensive equipment. Additionally, the output of the process is usually high-resolution imagery. The process is also very versatile, allowing for the capturing of both small objects and large landscapes from multiple angles. Despite all these positive points, photogrammetry has a few challenges, namely, the time-consuming processing process, poor outputs for bad lighting conditions and the dependence on a high degree of overlap between images which can be challenging in large complex environments.

Structured Light Scanning

Structured light scanning is another method of collecting 3D information about objects, surfaces and scenes. A structured 3D scanner shoots a regular pattern of light onto an object or surface. Cameras are then used to collect the resulting informationa and post-processing software is then used to generate the 3D information by triangulation [39]. Also, one or more cameras are used to capture how the pattern of light deforms when it hits the surface. The deformation of the light pattern allows the system to calculate the geometry of the object by analyzing the distortions. This technique is highly effective for capturing complex surfaces with high accuracy and is commonly used in industries such as manufacturing, robotics, and

medical imaging.

One of the key advantages of structured light scanning is its speed and precision. Unlike contact-based methods, which require physical interaction with the object, structured light scanning is non-contact and can quickly capture detailed 3D data. It is especially useful for scanning objects with intricate geometries or textures that might be difficult to capture with other methods. Additionally, structured light systems can be relatively compact and portable, making them adaptable to different environments and use cases.

A specific type of structured light system is the fringe projection system, which projects a series of sine wave variations with multiple fringes onto the surface of an object. The camera captures these projected patterns, which can include various combinations of coded patterns or fringes with different phases. This setup allows for the use of phase-shifting algorithms, improving accuracy and providing dense sampling. A fringe projection system offers a detailed 3D reading for each pixel and can capture fine details with exceptional precision. By projecting multiple patterns, the system helps distinguish the scene's contrast and ambient light from the object, offering better control over background lighting. However, data acquisition with fringe projection systems is slower compared to single-shot systems [39].

2.2.3 3D Mesh

3D meshes are an essential part of VR applications as they provide a way to model objects within scenes [31]. The idea of 3D meshes extends the concept of 3D points in space with points, vertices and complex geometric structures. 3D meshes, which are explicit structural representations in 3D, are foundational in representing spatial details within virtual environments, consisting of vertices, edges, and faces that form the structure of 3D objects. Complex geometric structures, like wire-frames and meshes, are formed by linking multiple vertices together through edges [4]. This is where the process of creating meshes in 3D usually begins. After this initial phase the final output mesh is produced by using shapes such as polygons to refine it. Meshes are processed by using polygons, such triangles and quadrilaterals, to create representations of objects and scenes that look realistic [41]. Most graphic editing workflows rely on triangle meshes, as they are essential for digital content creation (DCC) pipelines due to their widespread compatibility and integration with industry tools [25]. A mesh surface can contain a large number of triangles, making it highly complex. Before creating planar patterns for unfolding, it is often necessary to simplify these meshes while retaining key geometric details [14].

Meshes are versatile and widely used in fields such as computer graphics, animation, virtual reality, and video games. They allow for efficient rendering and realistic representations of intricate forms by adjusting the density and arrangement of polygons. Meshes can vary in complexity, from low-poly models for efficient real-time rendering to highly detailed, dense meshes for photorealistic visualizations. Advanced mesh processing techniques, such as mesh simplification, smoothing, and subdivision, are often applied to optimize these structures while retaining important visual details. Rendering a 3D mesh into a 2D image from a specific viewpoint primarily demands computational resources. Generally, complex 3D meshes require more processing power compared to simpler ones due to the additional detail and structure that need to be displayed accurately [31].

3D meshes are usually made up of collections of triangles; however, many can be represented more efficiently using polygons of various sizes. Meshes that contain polygons with variable side lengths are known as n-gon meshes [29]. Meshes also serve as the basis for many modeling and animation workflows, allowing artists and designers to modify shapes, add textures, and apply lighting for realistic appearances. In engineering and scientific fields, 3D meshes are crucial for simulations and analyses, where they model real-world objects for structural, environmental, and other types of studies.

The use of 3D meshes in virtual reality (VR) offers several advantages. One of the key benefits is high visual fidelity, as 3D meshes can capture intricate details and complex shapes, allowing virtual objects to closely resemble their real-world counterparts. This contributes to more immersive and realistic experiences for users. Additionally, well-designed 3D meshes facilitate smooth interactivity, enabling users to easily manipulate, grasp, and collide with virtual objects. Efficient rendering is another advantage, as optimized 3D meshes help maintain fluid frame rates and responsive performance in VR environments. The scalability of 3D meshes also plays a crucial role, as they can be adapted to create virtual worlds of various sizes, ranging from small objects to expansive landscapes.

However, there are challenges and considerations associated with 3D meshes in VR. One of the main concerns is mesh complexity, as highly detailed meshes can become computationally expensive, which may impact overall performance. Accurate texture and material mapping is also essential for achieving photorealistic appearances, as improper mapping can detract from the realism of the virtual environment. Furthermore, integrating 3D meshes with realistic physics simulations and collision detection is necessary to create believable interactions within the virtual world. Lastly, asset creation and management can be time-consuming and require specialized skills, as developing high-quality 3D meshes and handling the associated

asset pipelines is a complex process.

2.3 Machine Learning Methods

2.3.1 Neural Radiance Fields (NeRF)

Neural Radiance Fields (NeRF) is a method in 3D scene representation that leverages neural networks to generate highly realistic 3D renderings from 2D images. NeRFs use a neural network to encode the scene as a continuous volumetric field, where each point in 3D space is associated with color and density information. This approach enables photorealistic renderings with remarkable detail and realism, especially in scenes with complex lighting, reflections, and transparency. This makes NeRFs particarly powerful for novel view synthesis. NeRF models are self-supervised, allowing them to be trained solely from multi-view images of a scene. Unlike many other neural representations for 3D scenes, NeRF models need only the images and their poses to learn the scene, without requiring any 3D or depth supervision [13].
Traditional 3D representations, such as meshes or point clouds, are often limited in their ability to capture complex lighting effects, reflections, and fine details within a scene. Neural Radiance Fields, or NeRFs, address these limitations by using deep learning to encode the intricate details of a 3D environment. With only a set of 2D images from various angles, a NeRF can construct a continuous, volumetric representation of the scene, making it possible to synthesize new views with high fidelity. This approach leverages the power of neural networks to understand and recreate both the geometry and visual appearance of complex environments. A basic NeRF model represents 3D scenes as a radiance field approximated by a neural network, where the radiance field defines the color and volumetric density at each point and for every viewing direction in the scene [13].
The original NeRF paper introduces a method for representing 3D scenes as a neural volume, encoded in the weights of a simple, fully connected neural network architecture called a Multilayer Perceptron (MLP). In this approach, the MLP takes in five-dimensional inputs: a 3D position in space, given as x, y, and z coordinates, and a 2D viewing direction, specified by two angles. The output of the network provides color values (red, green, and blue) and a measure of volume density, which together correspond to the appearance of a pixel on the 2D image plane from that viewing angle. Unlike traditional 3D models that rely on explicit voxel grids, NeRF represents the scene entirely through the learned weights of the neural network, which maps spatial positions and directions to color and density information. The

MLP's parameters are optimized using a process called differentiable volume rendering, and the network is trained on actual images along with their known viewing directions. The error in color prediction is minimized using a basic mean squared error loss between the real and predicted pixel colors [8]. The approach presented in this paper builds on the strengths of volumetric representations, which are capable of accurately capturing the complex geometry and appearance of real-world scenes. One of the key advantages of this method is its ability to avoid the high storage costs associated with discretized voxel grids, which often become prohibitively expensive when modeling high-resolution scenes. Instead, the authors introduce a 5D neural radiance field parameterized by a simple Multilayer Perceptron (MLP) network, offering a more efficient representation of complex 3D scenes.

A significant technical contribution of the original NeRF paper [28] is the development of a differentiable rendering procedure based on classical volume rendering techniques, which allows the optimization of the neural radiance field representations using standard RGB images. This method also incorporates a hierarchical sampling strategy, which ensures that the MLP's capacity is effectively utilized by focusing on regions of the scene that are visible. Another important innovation is the use of positional encoding to map the 5D coordinates into a higher-dimensional space, enabling the representation of high-frequency scene content.
The results demonstrated that this approach not only outperforms existing state-of-the-art view synthesis techniques, including methods that fit neural 3D representations and those that train deep convolutional networks for volumetric predictions, but also sets a new benchmark in terms of photorealistic rendering. This was the first continuous neural scene representation capable of generating high-resolution, photorealistic novel views of real objects and environments from RGB images captured in natural settings. The paper's contributions represent a significant leap forward in the field of 3D scene representation, offering a more efficient and scalable approach to rendering realistic 3D scenes.

2.3.2 3D Gausssian Splatting

3D Gaussian Splatting is one of the most recent techniques for 3D reconstuction and novel view synthesis. It achieves this by approximating complex shapes using a dense collection of Gaussian ellipsoids rather than traditional polygons or voxels. 3D Gaussian Splatting is a machine learning based technique which means it has an optimizer that takes in source images and regulates the parameters of the model in order to minimize loss. It is made up of three major ideas. The first being the represention of 3D scenes as a cloud of primitives called 3D

Gaussians which are high-quality, unstructured representation of radiance fields. The second idea is a two step optimization: the optimization of the properties of each of the 3D Gaussians in space and the adaptive addition and removal of 3D Gaussians. The properties to be optimized are 3D position, opacity, anisotropic covariance, and spherical harmonic (SH) coefficients [21]. Thirdly, a real-time renderer which according to the authors of the original paper was inspired the work done on tile-based rasterization by [24].

According to [21] selecting a 3D Gaussian primitive for scene representation combines the benefits of volumetric rendering with the speed of splat-based rasterization, making it particularly effective for efficient and high-quality 3D scene optimization. This choice allows for a simpler yet powerful method to approximate complex lighting and geometry without relying on continuous volumetric fields. By leveraging discrete Gaussians, this approach facilitates rapid projection of scene elements onto a 2D plane, maintaining the visual accuracy and depth cues expected from traditional volumetric rendering but with a much lower computational overhead.

Additionally, this method is computationally efficient. Using a compact format to store Gaussian splats allows for the efficient representation of numerous splats while keeping data requirements relatively low [6]. This streamlined storage approach is beneficial because it enables a dense and detailed scene representation without demanding substantial memory resources. Each Gaussian splat captures essential spatial, color, and density information, and by representing these attributes efficiently, it becomes possible to handle a large number of splats that collectively approximate the intricate details of complex scenes. As a result, Gaussian splatting provides a way to store and manage high-detail 3D content with reduced data overhead compared to traditional methods. This compactness makes Gaussian splats especially well-suited for rendering high-resolution scenes that require intricate detail and photorealistic fidelity. Unlike voxel-based or mesh-based approaches, which often struggle with scalability as scene complexity increases, Gaussian splatting maintains both visual quality and performance. The compact data footprint of Gaussian splats allows for real-time rendering of complex scenes with high fidelity on devices with limited memory resources. This efficiency is particularly advantageous for applications in virtual reality, gaming, and augmented reality, where maintaining high resolution and realism at interactive frame rates is crucial. Consequently, Gaussian splats offer a robust solution for next-generation rendering challenges, providing a balance between detailed representation and computational efficiency.

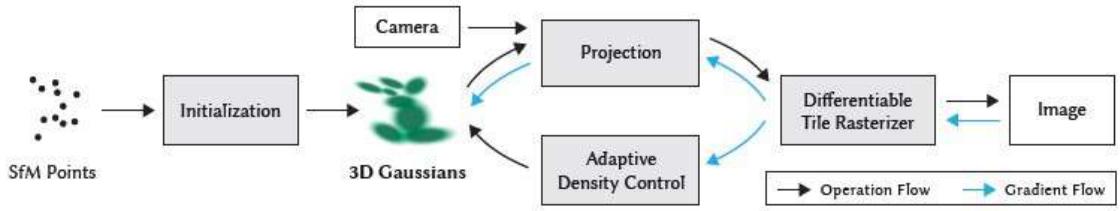

Figure 2.1: Gaussian Splatting System [21]

3D Gaussians

3D Gaussians are volumetric representations in 3D space that resemble anisotropic ellipsoids, with each one modeled as a 3D Gaussian distribution [11]. Unlike standard point representations, which lack volume, or spheres, which have uniform spread in all directions, these 3D Gaussians are anisotropic, meaning they can vary in shape and orientation, taking on elongated or compressed forms in different spatial directions. This flexibility allows each 3D Gaussian to capture unique variations in density and spread, providing a more nuanced representation of spatial detail.

3D Gaussians represent points with a probabilistic distribution in three dimensions, where each ”point” in the space has a Gaussian (bell curve) falloff. This falloff describes the distribution of density and color around a central position, typically diminishing as you move away from the center. In visual applications, each Gaussian can represent a soft-edged, volumetric ”blob” that collectively contributes to the shape, color, and shading of a scene or object. Each anisotropic Gaussian ”ball” in a 3D scene has a set of defining parameters that enable it to accurately represent both position and visual characteristics. The center, or mean, of the Gaussian specifies the spatial position of the Gaussian in 3D coordinates. To capture the shape and spread of each Gaussian, a covariance matrix is used, which describes how the Gaussian is stretched or compressed in different directions. Additionally, each Gaussian includes an opacity parameter, which determines how transparent or solid the Gaussian appears, as well as spherical harmonics parameters. These harmonics allow the Gaussian to model view-dependent color variations, meaning the color can change depending on the viewer’s perspective. In this scene representation, view rendering is achieved using point splatting technique. The Gausssian ”ball” in the scene is initially projected onto the 2D image plane, where its color is then calculated based on its spherical harmonic parameters [45].

Optimization

Gaussian splatting optimization and adaptive density control work together to enhance the

efficiency and quality of 3D scene rendering by dynamically adjusting the distribution and density of 3D Gaussians based on scene content and viewer perspective.
The optimization process relies on iterative cycles of rendering and comparing the resulting image with training views from the captured dataset. This process is essential for refining the 3D scene representation, but it is prone to inaccuracies due to the inherent ambiguities in the 3D-to-2D projection. As a result, the optimization not only generates geometry but must also be capable of removing or repositioning incorrectly placed geometry. A key factor in achieving an efficient and compact representation is the quality of the covariance parameters of the 3D Gaussians. If the parameters are tuned correctly, large, homogeneous regions of the scene can be represented by fewer, larger anisotropic Gaussians, significantly improving memory usage and computational efficiency.

The optimization is typically carried out using Stochastic Gradient Descent (SGD) methods, which take advantage of GPU-accelerated frameworks, as well as custom CUDA kernels for specific operations, aligning with recent advancements in the field [12] [42]. The efficiency of this optimization process heavily depends on fast rasterization, which is the primary computational bottleneck. To ensure smooth optimization, a sigmoid activation function is used for opacity parameters to constrain them within a range of $[0, 1)$, while an exponential activation function is applied to the covariance scale, providing smooth gradients for both parameters. The initial covariance matrix for each Gaussian is typically estimated as an isotropic Gaussian, where the axes correspond to the mean distance to the three closest points in the scene [21].

Regarding adaptive Gaussian control, the process generally starts with a sparse set of points derived from Structure-from-Motion (SfM), and the method is progressively applied to refine the number and density of Gaussians within the scene. The goal is to transform the initial sparse set of Gaussians into a more densely populated set that better captures the scene's details. After a warm-up phase for optimization, every 100 iterations, the Gaussians are densified, and any Gaussians that have become effectively transparent—indicated by their opacity being below a predefined threshold—are removed. This adaptive approach ensures that the representation fills in areas of the scene that lack sufficient geometry, particularly where the reconstruction is incomplete or sparse. Furthermore, the method targets areas with excessively large Gaussian coverage, which often results from over-reconstruction. These areas are identified by large view-space positional gradients, signaling regions where the optimization process has not yet adequately adjusted the Gaussians. By focusing on these regions, the op-

timization refines the scene representation, adjusting the Gaussians to better match the true structure and appearance of the scene[21].

Differentiable Rasterizer

The optimization process is designed to ensure both rapid rendering and efficient sorting, facilitating approximate alpha-blending, including for anisotropic splats. One key challenge in previous approaches was the limitation on the number of splats that could be incorporated into gradient computations. To address this, a tile-based rasterization approach was introduced in [21], inspired by recent advancements in software rasterization. Unlike traditional methods, which require sorting at the per-pixel level, this approach allows the sorting of Gaussian primitives across the entire image at once. By doing so, the method avoids the computational overhead of sorting for each pixel, a significant bottleneck in earlier alpha-blending solutions. The rasterizer enables efficient backpropagation over an arbitrary number of blended Gaussians while maintaining low memory overhead—requiring only constant space for each pixel. Moreover, the rasterization process is fully differentiable and can handle anisotropic splats similar to previous 2D splatting methods [23].

To begin, the screen is divided into 16×16 tiles. Each tile undergoes the process of culling the 3D Gaussians against the view frustum, keeping only those with a 99% confidence interval within the frustum's boundaries. Gaussians that are outside the frustum, particularly those near the near and far planes, are discarded to avoid unstable computations. After this culling, the Gaussians are instantiated based on the number of tiles they intersect and assigned a key combining their view space depth and tile ID. This enables the sorting of the Gaussians using a GPU-accelerated Radix sort, eliminating the need for per-pixel sorting. The result is a fast, approximate alpha-blending process that provides high-quality rendering without visible artifacts as the splats approach the pixel size [22]. After the Gaussians are sorted, a list for each tile is created by identifying the first and last depth-sorted Gaussian that splats to a given tile. For efficient rasterization, each tile is processed by a thread block that loads the Gaussians into shared memory, and pixels are processed front-to-back, accumulating color and alpha values. The rasterization stops when the target saturation of alpha is reached for a pixel, improving the parallelism of the computation.

A distinctive feature of this method is its ability to perform gradient updates on all blended primitives without limitations. This avoids the need for hyperparameter tuning based on scene

complexity and allows the method to handle varying depth complexities accurately. During the backward pass, the full sequence of blended points per pixel from the forward pass is recovered. To minimize memory overhead, the system reuses the sorted Gaussian array from the forward pass and performs the back-to-front traversal for gradient computation. The gradients are computed by dividing the final accumulated opacity by each point's opacity during the backward pass, thus efficiently calculating the necessary coefficients for training[21].

2.4 Neural Radiance Fields vs Gaussian Splatting

Neural Radiance Fields (NeRFs) and Gaussian Splatting are both methods used for 3D scene representation and view synthesis, but they differ in their underlying approaches and computational efficiency. NeRFs represent a scene using a neural network that models the radiance field of the scene by learning to map spatial coordinates and viewing directions to color and density values. While NeRFs are capable of generating highly photorealistic images, their training process can be slow, and the required computation for rendering can be intensive, particularly for high-resolution scenes. On the other hand, Gaussian Splatting represents a scene through a set of 3D Gaussian primitives, allowing for faster and more efficient scene representation and rendering. By using a compact set of Gaussians, this approach reduces memory usage and can handle high-resolution scenes more efficiently than NeRFs. Additionally, Gaussian Splatting optimizes for both geometry and density control, making it more adaptive and flexible for real-time applications. In summary, while NeRFs excel in generating high-quality images, Gaussian Splatting offers advantages in computational efficiency and scalability for large-scale or real-time rendering tasks.

Feature	NeRF	Gaussian Splatting
Representation	Uses a neural network to model the radiance field.	Uses a set of 3D Gaussian primitives for scene representation.
Rendering Speed	Slow rendering due to computational complexity.	Faster rendering, especially for high-resolution scenes.
Memory Efficiency	Requires large memory for high-resolution scenes.	More memory efficient, compact representation.
Real-time Performance	Not suitable for real-time rendering without optimization.	Suitable for real-time rendering with optimization.
Quality	High-quality, photorealistic images.	Good quality with efficient handling of complex geometry.
Optimization	Uses volume rendering for optimization.	Uses adaptive control to optimize density and geometry.
Training Data	Requires ground truth images and their corresponding camera views for training.	Requires sparse points from SfM (Structure from Motion) for initial optimization.

Table 2.1: Comparison between NeRF and Gaussian Splatting

2.5 Rendering

Rendering is the process of creating images from a 3D model using computer software, converting a three-dimensional scene into a two-dimensional image while considering factors like lighting, textures, and shadows. This technique is commonly used in fields such as animation, visual effects, architectural visualization, product design, and video games to produce both realistic and stylized images or animations. There are various rendering techniques, such as rasterization, ray tracing, and global illumination, each with its own strengths and specific use cases. The choice of technique depends on factors like the desired visual quality, available computational power, and project specifications. In summary, rendering plays a crucial role in modern computer graphics and visual communication, helping artists and designers transform 3D models into engaging, high-quality images and animations for diverse purposes [27]. The rendering process involves several key stages, such as geometry processing, lighting calculations, texture mapping, and projection. Initially, in the geometry processing stage, the vertices of the 3D model are transformed and clipped to determine their screen positions. Then, during lighting calculations, the system considers the position, intensity, and color of light sources to compute the brightness and color for each pixel. In the texture mapping stage, textures are applied to the object surfaces to enhance detail and realism. Lastly, the projection stage converts the 3D scene onto a 2D plane, resulting in the final image or animation [27].

Ray tracing is one very common group of rendering algorithms. These algorithms simulate the transport of light using the principles of geometric optics, assuming that light travels instantaneously in straight lines through a medium [10]. By tracing the paths of rays of light as they reflect, refract, and transmit through surfaces, ray tracing can accurately simulate complex lighting effects such as shadows, reflections, and refractions. This method produces highly realistic images, as it closely mimics the way light interacts with materials in the real world. However, ray tracing is computationally expensive, often requiring significant processing power to produce high-quality results, especially in real-time applications.

Another group of rendering algorithms is Real-time Rendering. Real-time rendering involves the generation and display of computer-generated graphics instantaneously, typically at frame rates suitable for interactive applications [5]. In the context of real-time shading, techniques such as free-form planar area lights are used, with a linear transform cosine function applied to achieve efficient rendering in real-time. This method allows for realistic lighting and shading effects while maintaining the high frame rates necessary for interactive environments, such as video games or virtual reality applications. Real-time rendering is crucial in fields where immediate feedback and smooth visual experiences are essential. By optimizing algorithms and hardware performance, real-time rendering ensures that complex scenes with dynamic lighting, shadows, and textures can be displayed without noticeable delays, offering an immersive and responsive user experience.

In recent years, advancements such as real-time rendering have made it possible to achieve near-photo-realistic graphics at interactive frame rates. This has been particularly transformative in video games and virtual reality, where players or users expect smooth and responsive visual experiences. Techniques such as real-time shading with free-form planar area lights and global illumination have enhanced the quality of real-time rendering, making it possible to achieve more lifelike environments in real-time applications.

Rendering also plays a pivotal role in visual communication, helping to convey design concepts, architectural visualizations, and simulations. The development of more powerful hardware, along with sophisticated software algorithms, continues to push the boundaries of what is possible in rendering, making it a critical tool in creating realistic, immersive, and engaging visual content.

2.6 5G in Remote Operations

5G technology is transforming remote operations by providing faster, more reliable, and low-latency connectivity, which is essential for industries that rely on real-time data and seamless communication. The next generation of wireless networks offers significant improvements in bandwidth, speed, and reliability, enabling remote control and monitoring systems to function more efficiently than ever before. This capability is especially important in fields such as manufacturing, healthcare, logistics, agriculture, and energy, where remote operations are becoming increasingly vital for improving productivity, safety, and operational flexibility.
There are a number of parameters and metrics that play an important role for communications in remote operations. in effect, these metrics also translate to the requirements for the use of 5G in remote operations. These metrics are: Delay and Reliabily [19]. The latency, or delay, within mobile networks directly impacts the responsiveness of remote-controlled systems. In the case of current 4G mobile networks, the delay at the data link layer ranges from 50 to 300 milliseconds (ms), which is often sufficient for many applications but becomes a limiting factor for more demanding use cases. For instance, real-time gaming requires a delay of 50 ms with a packet loss rate of 10^-3 to maintain a smooth experience, while interactive gaming can tolerate a delay of up to 100 ms with the same packet loss rate. For applications like streaming and file downloading, the delay can extend to 300 ms, but it must be paired with a significantly lower packet loss rate of 10^-6 to ensure quality [19]. The limitations of 4G networks become more evident when considering the strict demands of emerging remote operation technologies. As industries increasingly adopt autonomous systems, robotics, and telemedicine, the need for low-latency communication becomes even more critical. Delays in these systems can result in poor performance, with actions being executed too late or with diminished accuracy, potentially compromising safety and efficiency.
Reliability, closely tied to delay, is another crucial factor in remote operations. While delay impacts how quickly commands are transmitted and executed, reliability measures the consistency and availability of the network. A highly reliable system ensures that packets of data are successfully transmitted to their destination within the required time frame, which is essential for real-time control and monitoring. In scenarios where low reliability is present, the consequences can be similar to those caused by high delay, particularly in applications requiring tactile or immediate feedback, such as robotic surgeries or remote industrial machinery control.

The shift from 4G to 5G technology offers promising solutions to these challenges. With

ultra-low latency capabilities, 5G reduces the delay to as low as 10-20 ms due to its shorter sub-frame of just 1 ms, making it ideal for applications requiring high precision and real-time responsiveness. Moreover, 5G's improved reliability ensures that data packets are delivered with greater consistency, further supporting the dependability of remote operations.

2.7 Regulatory Framework

2.7.1 Public Protection and Disaster Relief (PPDR)

Whenever there are disasters or emergency situations like natural disaster, fires and so on, there usually is the need to operate in sites that have been affected by these phenomena in a safe, secure and harmless way. In this light, remote communications come into play. Public Protection and Disaster Relief (PPDR) provides the framework within which to operate in such situations. Public Protection and Disaster Relief (PPDR) refers to the coordinated efforts of various public safety and emergency response agencies to protect citizens, manage crises, and provide essential services during natural or human-made disasters. PPDR involves activities such as emergency medical response, firefighting, law enforcement, search and rescue, and civil defense. In times of crisis, effective PPDR ensures that lives are saved, property is safeguarded, and communities are supported during recovery. The PPDR is concerned with two aspects namely the Public Protection which is a preventative actions taken in anticipation of disasters, and Disaster Relief which is more concerned with the activities after a disaster has actually occurred [3].

PPDR services address a wide range of natural and human-made threats. A key responsibility is managing emergency and surveillance operations across land, sea, and air. Since much of this work occurs directly in the field, the tools and equipment used must be well-suited to these requirements. In field operations, vehicles equipped with specialized devices, systems, and services are essential, and factors such as safety, efficiency, and ergonomics are critical considerations. The vehicles and equipment must be durable, secure, and capable of functioning under highly demanding and diverse conditions, as they often rely on numerous technical devices to support their tasks effectively [16]. Agencies that use the PPDR in various application areas such as law and order, protection of life and properties and responding to emergencies play a vital role in society. Hence, the objective is often to respond to catastrophes that put human lives, society and the environment in danger in disaster relief situations [30]. PPDR agencies often collaborate internationally, sharing resources, strategies, and ex-

pertise to strengthen global resilience against large-scale emergencies. This collaborative and technology-driven approach makes PPDR a cornerstone of modern public safety and disaster management efforts.

Field operations in PPDR are growing more dependent on information technology, with a particular focus on wireless communication systems [16]. The ability to exchange information is crucial for enhancing coordination among PPDR officers during operations. This capability is particularly important for effective response and crisis mitigation, supporting the mobility and efficiency of first responders. A critical component of PPDR is reliable communication systems that allow for swift information sharing and coordination among agencies. In previous years organizations and agencies concerned with providing public saftey relied on using Land Mobile Radio (LMR) systems to support mission-critical voice communications [30]. However, with the advent of 5G systems there is a growing trend of implementing systems that use this generation of cellular technology. Ultra-reliable communications are essential for services that demand a high level of reliability and low latency to ensure a satisfactory user experience or to activate critical services. The introduction of 5G, which supports mission-critical systems, brings additional security challenges. As data volumes grow due to higher transmission speeds and a surge in connected devices, ensuring privacy and reliability becomes increasingly important. While 5G incorporates some security features, further research on advanced security mechanisms remains necessary [30]. 5G systems are highly flexible and support applications that have stringent latency requirements. One of the main usage scenarios of 5G systems as recommended by ITU-R M.2083 is Ultra-reliable and Low Latency Communications (URLLC) which is a desireable feature of situations covered by PPDR as it is essential to to cater for safety-critical and mission critical applications [18]. Consequently, PPDR has been listed within the category of emerging services for 5G [1]. 5G aims to provide flexible systems to support different The primary focus in developing ICT systems for PPDR is on standardizing interoperability frameworks for both applications (such as command and control systems) and infrastructure (such as interface gateways and mobile units). Usability is also a key concern, as many current solutions lack ergonomic design and are challenging to integrate with existing vehicles and infrastructure[16]. This has led to many efforts to design systems that allows for seemless user experiences for remote operators using system that operate within the framework of PPDR. The emergence of 5G offers transformative potential for PPDR by delivering higher speeds, lower latency, and enhanced connectivity. With its advanced capabilities, 5G can support massive device connectivity, enabling a multitude of sensors, drones, and IoT devices in real-time on the same network, providing a fuller operational picture. Fur-

thermore, 5G's network slicing allows for dedicated and secure channels for PPDR services, ensuring high priority and uninterrupted connectivity during peak times or disasters. This level of connectivity enhances response times, situational awareness, and decision-making, revolutionizing how public safety operations are conducted in critical, time-sensitive situations.

Chapter 3

Methodology

3.1 Overview

To achieve the stated goal of this thesis, a system is designed and developed as shown below. The idea of this system is to capture scenes for remotely operated vehicles, model these scenes using Gaussian splatting, and render them on the Meta Quest 2 headset. Doing this allows a remote operator to experience an immersive view of the remote scene. This methodology uses a range of tools to support data acquisition, model training, and VR rendering processes. The primary hardware includes a high-resolution camera and the Meta Quest 2 headset. Software components, such as Python scripts and COLMAP for data preprocessing and the original gaussian splatting implementation, were used to ensure efficient processing and compatibility with the Gracia VR application for immersive rendering.

3.2 Software and Hardware Tools Used

3.2.1 COLMAP

COLMAP is a powerful photogrammetry tool used in the preprocessing stage of this project's system pipeline. COLMAP is used to perform Structure-from-Motion (SfM) and Multi-View Stereo (MVS) operations, which help in reconstructing the 3D structure of the scene from the 2D images [40]. It aligns the images, estimates camera poses, and creates dense 3D point clouds or meshes that serve as input for the subsequent model training phase. After capturing the original images of the scene, COLMAP is used to extract key features and match them across multiple images, enabling it to estimate the camera positions and orientations. This process, known as Structure-from-Motion (SfM), generates accurate 3D points and camera calibrations

that are critical for the next stage of model training. COLMAP works by doing two main things: feature matching and 3D reconstruction. It detects and matches key features (such as edges, corners, and textures) across the set of images. By analyzing the relative positions of these features in the images, COLMAP estimates the camera poses. The matched features are then converted into 3D coordinates that represent the scene's geometry. COLMAP can generate camera calibration data, including intrinsic parameters (such as focal length and lens distortion) and extrinsic parameters (such as the position and orientation of the camera). These calibrations are essential for ensuring the accuracy of the 3D reconstruction and serve as input to the Gaussian splatting model training process. The output 3D points and camera calibrations from COLMAP is then what serves as the input to the Gaussian splatting algorithm which can be trained to produce realistic, high-quality scene renderings.

3.2.2 Google Colab

Google Colab provides the main computational environment for the development phase of this project. It provides a free and fully configured runtime for deep learning [33]. It allows access to the GPUs necessary for the training stage of the gaussian splatting algorithm. It is utilized for running the Python scripts that process the raw data, transforming it into a format suitable for training the Gaussian splatting model. These scripts handle tasks such as image preprocessing, and the conversion of 2D image data into the appropriate 3D model inputs. After the raw data has been acquired, including the high-definition images and the 3D points and camera calibrations generated by COLMAP, Google Colab is used to run the Python scripts that transform these inputs into a suitable format for training the Gaussian splatting model. Using Colab for the model training also allowed for the leveraging of powerful GPUs to accelerate the model training, which can be computationally demanding given the complexity of the 3D data and the machine learning algorithms involved. By running the training process in Colab, it was possible to speed up the process and iterate more quickly, which was essential for optimizing the model's performance and ensuring high-quality rendering results in the later stages of the pipeline.

3.2.3 SuperSplat

SuperSplat is a Gaussian Splatting editor developed by PlayCanvas, designed to provide an extensive suite of editing tools for point cloud manipulation and scene cleaning. This is a tool that is suited for 3D scene editing and cleaning especially for gaussian splatting. It is essential

for refining and managing Gaussian splat data, making it an ideal tool for post-processing in 3D visualization and editing pipelines.

In this project, SuperSplat served as the primary scene cleaning technology in the system pipeline, employed after model training to refine and optimize the captured 3D data for improved accuracy and usability [35].

SuperSplat supports several editing features essential for point cloud refinement, such as Gaussian removal, compression, and cropping. These tools allow for selective management of Gaussian points, which can be visualized in two distinct viewing modes: Centers and Rings. The Centers mode presents a sparse point cloud that can be adjusted by increasing the splat size to facilitate selection, while the Rings mode visualizes each Gaussian with easy-to-select rings, simplifying the selection and modification processes. Additionally, SuperSplat offers three selection tools—Rect, Brush, and Picker—that enable flexible and precise edits. Rect allows for large-scale cropping of Gaussian splats by drawing a selection rectangle, Brush offers refined control similar to a paintbrush, and Picker provides precise, granular selection through single-point clicks [37]. SuperSplat was particularly chosen for the project because it served not only as an excellent editing tool but also a way to view and explore the modelled scenes after training. And because files can be exported in various formats, including .ply, a compressed version, or .splat it integrated well into the system pipeline. With recent updates, SuperSplat also supports Progressive Web App (PWA) functionality, allowing for a native desktop experience. a snapshot of the SuperSplat platform is shown below.

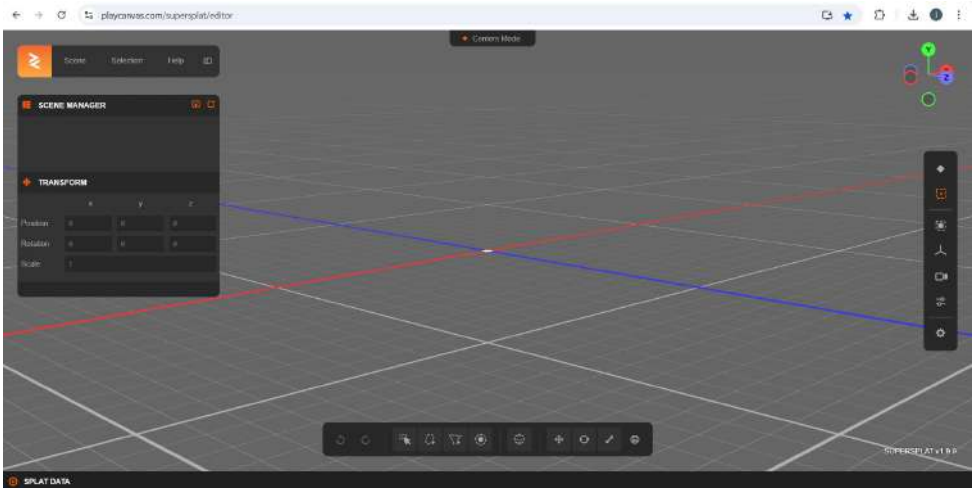

Figure 3.1: SuperSplat Platform [36]

3.2.4 Meta Quest 2

In the virtual reality space, the Meta Quest series of head mounted displays is one of the most popular series of hardware equipment enabling the realization of innovation in this field. De-

veloped by Meta Platforms Inc., the headsets fully immerses users in a virtual environment. Some VR environments can also be shared online, allowing users to interact, communicate, and experience the space collectively [38]. VR headsets create an immersive experience by placing the user into a virtual environment where they can interact in real time through physical movements and gestures. This virtual space can offer novel experiences, allowing users to explore new worlds or engage in interactive activities. Beyond exploration, VR headsets are widely used for entertainment, including gaming, browsing the web, watching videos, and socializing in shared virtual spaces. These headsets also support practical applications like virtual training, educational simulations, and even remote collaboration, making them versatile tools for both entertainment and functional experiences [38].

The specific headset from this series that was used for this project is the Meta Quest 2. It provided the hardware platform for the running of the virtual reality application that enabled the viewing of the reconstructed scenes. One of the defining features of this headset is the high resolution display, motion tracking and ease of use of users especially in immersive applications. Being equipped with projection and sensor tracking allows users to experience environments that they otherwise would not have the chance to experience in real life. The Quest 2 features a powerful Qualcomm Snapdragon XR2 processor, 6GB of RAM, and a high-resolution display, which makes it suitable for a range of VR applications—from gaming and social interaction to immersive storytelling and visualization. Using the Gaussian splatting method to reconstruct scenes that can be rendered in the Quest 2 is especially useful for rendering realistic and data-heavy scenes in VR, as it balances quality with computational efficiency, making it a suitable approach for VR.

Figure 3.2: Meta Quest 2 HMD and Controllers [7]

3.2.5 Gracia Application

Gracia is a new spatial computing application and platform that allows users to visualize and experience 3D gaussian spaltting based scenes in the Meta Quest 2. It also allows the creation of Gaussian splatting-based volumetric videos for the Meta Quest 2. In VR, this approach allows for lifelike, volumetric 3D scenes that users can explore interactively, creating a sense of true immersion. Gracia uses this technology to allow users to navigate within 360° scenes as if they were physically present in those scenes. This helps in breaking away from the traditional constraints of pre-recorded 2D video or static 3D imagery. Therefore, users are able to experience these scenes in a seemless and more natural way compared to traditional methods [15].

On the Gracia platform, users can use specific tools for content creation and distribution which allows creators to craft and share immersive experiences in ways previously impossible on traditional platforms. These tools streamline the process of converting real-world scenes into volumetric representations suitable for VR. The platform's approach to spatial computing enables the immersion of users as well as scalability, as it allows creators to distribute their content across VR devices efficiently. As a result, Gracia is poised to set a new standard in VR experiences by combining innovative volumetric capture with accessible and user-driven storytelling, marking a significant leap forward for the Meta Quest 2 and spatial computing as a whole.

One of Gracia's standout features is its ability to offer users innovative 2D fly-throughs within these immersive 3D environments. These fly-throughs are not merely passive; they are dynamic sequences that guide the viewer through different perspectives, revealing nuances within a scene that might otherwise go unnoticed. This feature allows for flexible interaction, as users can choose to view specific parts of a scene from various angles, giving a unique balance of guided storytelling and interactive exploration. By leveraging Gaussian splatting for volumetric data, Gracia can produce these smooth fly-throughs with minimal computational overhead, which is ideal for a consumer-grade VR headset like the Meta Quest 2. This design consideration expands access to advanced spatial computing, allowing users to experience high-fidelity scenes without requiring enterprise-level hardware [15].

In this project, Gracia was downloaded onto a desktop computer as well as the Meta Quest 2 via the sidequest platform and served as the OpenXR runtime for the streaming of the rendered gaussian splatting scenes from the windows application. The following figure illustrates the setup.

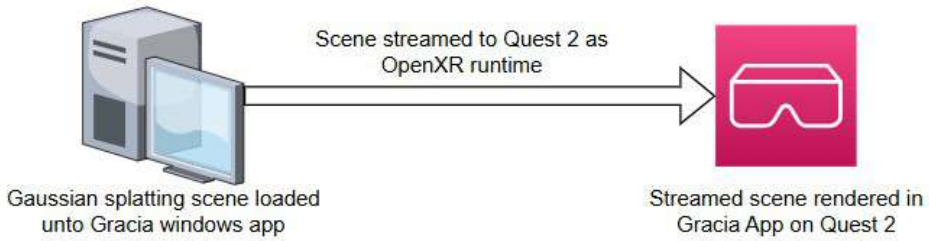

Figure 3.3: Setup for VR Rendering

3.3 System Pipeline

To achieve this, the system is designed around the following stages:

1. **Data Acquisition**: High-definition images of the scene are captured using a high-resolution camera.

2. **Data Preprocessing**: Three Python scripts process the raw data, transforming it into a suitable format for model training. Then COLMAP used to generate input files for model training.

3. **Model Training**: The preprocessed data is used to train a Gaussian splatting model.

4. **Model Cleaning**: This optional step cleans artifacts from the modeled scenes to improve rendering quality.

5. **VR Rendering**: Finally, the model is rendered on the Meta Quest 2 using the Gracia VR application.

The system pipeline is illustrated below:

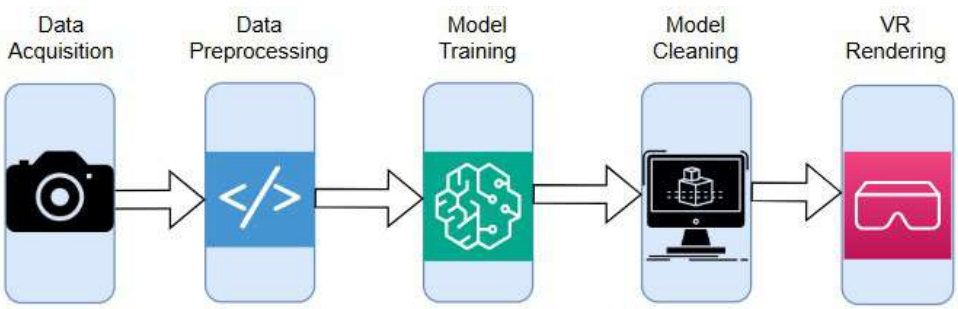

Figure 3.4: System Pipeline

The remaining parts of this chapter go deep into each phase of the system, the design considerations and the development process at each phase.

3.3.1 Data Acquisition

To acquire the data, a high-resolution camera was used to capture 4K images and video of the scene. This approach was used to evaluate how effectively the model learns from raw images compared to frames extracted from video. All captures were taken in landscape mode to provide a wide field of view, enhancing the immersive quality of the reconstructed scene.

Images were captured in well-lit conditions, ideally during the day, to avoid shadows that could affect reconstruction accuracy. The camera was moved around the stage to capture as many viewpoints as possible, as this improves the model's performance. Each image was taken with significant overlapping with adjacent images to ensure accurate feature matching during reconstruction.

For video capture, gradual camera movement was essential to avoid blurred transitions between frames. This careful approach helps maintain clarity and consistency, which are crucial for high-quality scene reconstruction.

This process of capturing images and videos of the target scene using a high-resolution camera to ensured that sufficient detail of the scene is available for the Gaussian splatting model. As a result, Several key factors were considered to optimize the quality of data collected:

1. Setup of Camera: A high-definition camera capable of capturing 4K images and videos was used to ensure a high level of detail. The camera settings were configured to maintain consistent exposure, white balance, and focus throughout the capture session, reducing variability in lighting or color that could interfere with model training.

2. Lighting Conditions: To avoid shadows and inconsistencies, data was captured under uniform lighting conditions, preferably during the day or with diffused artificial lighting. Because shadows can lead to the appearance of artifacts in the captured images and hence the reconstructed scenes, the capturing process was carried out in conditions that ensure that details are accurately represented across all images.

3. Capture Angles and Coverage: To ensure a complete representation of the scene, images were captured from multiple angles. The camera was moved systematically around the scene to capture it from a variety of perspectives, with an emphasis on achieving 60-80% overlap be-

tween consecutive images. This overlap is crucial for accurate feature matching, which allows the reconstruction model to stitch together different views seamlessly. By capturing the scene from different elevations, angles, and distances, the data set provides comprehensive coverage for both static and dynamic elements. Additionally, for scenes without objects present in the environment, the camera was moved through the scene as well as around it to capture as much information from the scene as possible.

4. Image and Video Capture Orientation: Both still images and videos were captured in landscape orientation to maximize the field of view. The inclusion of video allowed for a comparison between using individual frames from video and standalone images for model training. Videos were captured with gradual camera movement to minimize motion blur and ensure smooth transitions between frames, especially for sequences where the camera moves around the scene in real-time.

5. Data Preprocessing: Following data acquisition, preliminary preprocessing steps were applied to prepare the images and video frames for model training. This included resizing, normalization, and, if necessary, adjusting brightness and contrast to achieve consistency across the dataset.

The development of the system relied on a number of scenes that were captured from diverse settings and locations in order to prove the effectiveness of the system to reliably model these diverse scenes. a few pictures from the data set of images of the scenes is shown below. After the collection of these images and videos, additional images and videos were taken of other scenes to build a larger well prepared data set that can be used to train this model or future models to assess their effectiveness at modeling diverse, dynamic scenes.

Figure 3.5: Sample Scene 1 from the Dataset

Figure 3.6: Sample Scene 2 from the Dataset

Figure 3.7: Sample Scene 3 from the Dataset

3.3.2 Data Preprocessing

TODO: outline of scripts used for preprocesssing with scrreenshots with exp of what they do.

The original Gaussian splatting implementation is designed to work with images up to 1200 pixels in width. This limitation of the Gaussian splatting implementation stems from several technical considerations such as:

Memory constraints during training Computational efficiency requirements Historical benchmarking data availability Original model architecture design decisions

Since the images and videos taken in the data acquisition phase were captured in 4K resolution, resizing was necessary to ensure compatibility with the model training. Using a custom Python script, all images were resized to the required dimensions. For videos, the script also split each video into individual frames, allowing them to be processed as image sequences compatible with the Gaussian splatting pipeline.

Image Processing Pipeline **Image Resizing**

Input: 4K resolution images (3840 x 2160 pixels) Output: Processed images (1200 pixels width, maintaining aspect ratio) Processing steps:

Lanczos resampling for optimal quality Aspect ratio preservation to maintain scene geometry Metadata preservation for camera parameters Quality validation checks post-resize

The video preprocessing pipeline involves multiple stages:

Frame Extraction

Videos are decomposed into individual frames Frame rate is preserved in metadata Timestamps are recorded for temporal consistency

Frame Processing

Each extracted frame undergoes the same resizing process Temporal consistency checks are implemented Frame sequence validation is performed

Screenshots of these preprocessing scripts are included for reference.

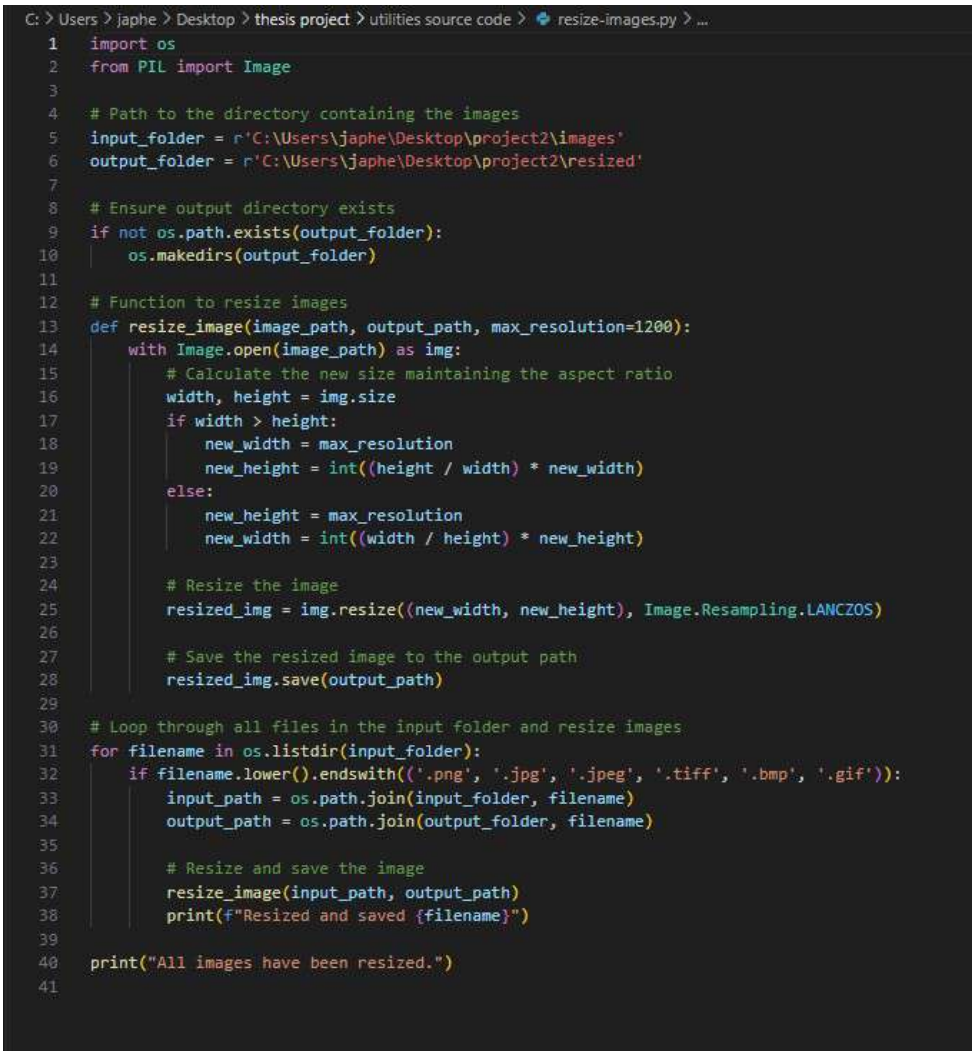

Figure 3.8: Image Resizing Script

The above image of the image resizing script is designed to resize images from a specified input directory and save the resized images into an output directory. This script is especially useful for batch resizing, making it suitable for tasks where all images need to be downscaled to a consistent maximum resolution while preserving their aspect ratio. The libraries used are: **os**: Used for file path manipulation and directory operations, allowing the script to locate, iterate over, and create directories if needed. **PIL (Pillow)**: The Image module from the Pillow library is used to handle image processing tasks. It provides functionalities for opening, resizing, and saving images.

The core function, resize_image, adjusts each image's dimensions based on a maximum resolution parameter (defaulting to 1200 pixels) while preserving its aspect ratio. The function calculates new dimensions based on the original image size, ensuring the longest side matches the specified resolution, then uses the Image.Resampling.LANCZOS filter for high-quality resizing. The script iterates over all files in the input folder, checks if each file is an image, and processes it accordingly, printing a message each time an image is resized and saved. It checks if the file extension matches a list of common image formats (such as PNG, JPG, JPEG, TIFF, BMP, and GIF). For each qualifying image, it constructs the full input and output paths and calls resize_image to resize and save the image.

Figure 3.9: Video Frame Extractor Script

The video frame extractor code snippet above extracts frames from a video file at specified intervals and saves them as images in an output directory. This script is useful for video analysis, generating keyframes, or converting a video into a sequence of images. It utilizes several libraries: cv2 (OpenCV) for handling video files, tqdm for displaying a progress bar, and PIL (Pillow) for image resizing and saving. Additionally, the script uses the argparse library, which could potentially allow command-line argument parsing, although there is no evidence of its usage in the visible portion of the code. The script defines a function called extract_frames with parameters for the video path, output folder, frame interval, and maximum resolution. It first creates the output folder if it doesn't already exist using os.makedirs with exist_ok=True, allowing it to proceed without errors if the folder is already there. Next, the script opens the video file using OpenCV's cv2.VideoCapture and retrieves properties like frames per second

(FPS), frame count, and video duration. This information is printed to the console to give users context about the video's length and frame rate. A frame counter (frame_number) is initialized at zero, and a progress bar (pbar) from tqdm is set up based on the total number of frames to provide a visual indication of the extraction progress. The script enters a while loop that processes each frame in the video. It reads a frame using video.read() and checks if the frame was successfully captured. If not, the loop breaks, indicating the end of the video. For frames at intervals matching the frame_interval parameter (e.g., every nth frame), the script converts the frame from an OpenCV format (BGR) to a format compatible with Pillow (RGB). It then resizes the frame to the specified max_resolution while preserving the aspect ratio. The aspect ratio calculation adjusts the width or height based on which dimension is greater, using the Image.Resampling.LANCZOS filter to ensure high-quality downscaling. The resized frame is saved as a JPEG in the output folder with a filename that includes the frame number, padded to six digits.

3.3.3 Model Training

The goal of the Gaussian splatting model training process is to produce 3D point cloud representations of the scenes that it is being trained on at the end of the training iterations using a combination of source images and 3D structural data. The inputs to the model include the source images along with three essential output files from the COLMAP structure-from-motion (SfM) pipeline: points3D.bin, images.bin, and cameras.bin. All three files from the SfM process which provide crucial information about the 3D geometry and camera parameters are fed as input into the training algorithm as a starting point. The data contained in points3D.bin are the 3D points in space and associated feature descriptors, images.bin includes the camera poses and orientation data for each input image, and cameras.bin provides intrinsic camera parameters such as focal length and sensor size. All these inputs give the model a well-defined understanding of the spatial configuration of the scene and camera characteristics, which are critical for accurate 3D reconstruction.

The training process begins once the algorithm has been fed and initialized with these input data. The training progresses in two main stages: an initial stage of 7,000 iterations followed by a prolonged training phase up to 30,000 iterations. The training usually takes about one (1) hour for the full 30,000 iterations. During the first 7,000 iterations, the model works to establish a coarse representation of the scene by optimizing the placement, size, and density of Gaussian splats in 3D space. The output that is saved after the 7000 iterations is

essential because it helps in tracking the early training progress of the model and helps in capturing the basic structure and layout of the scene. The idea at each iteration is for the model to adjust parameters in such a way to optimize the error between the projection of its internal 3D representations and the 2D source image inputs by leveraging feedback from these discrepancies to make incremental improvements in spatial accuracy and detail.

Reaching 30,000 iterations after the intial stages of training then allows the model to properly optimize and represent each gaussian splat properly. This extended phase focuses on refining the splatting representation, allowing the model to capture finer details, textures, and more subtle lighting variations present in the source images. By adjusting the Gaussian splats' parameters (e.g., mean positions, variances, colors, and opacities), the model iteratively improves the quality of the 3D representation. This aims to create a more lifelike and realistic reconstruction. Each point carries additional information about colour, transparency, and spread which helps smooth out the reconstruction and minimizes the appearance of artefacts in the scene. And these are issues typically present in point clouds.

The final output of the model is a file named splat.ply, stored in PLY format, which can be viewed in standard 3D visualization software. This splat.ply file encapsulates the entire learned 3D scene as a high-quality point cloud of Gaussian splats, complete with color and transparency data. In a 3D viewer, users can explore the scene interactively, navigating around the reconstructed environment to inspect different viewpoints. The advantage of using Gaussian splats is that it allows for smoother transitions between points and a more continuous visual effect compared to traditional point clouds, which often appear discrete or fragmented. This detailed 3D representation is ideal for applications such as virtual reality, where immersive and photorealistic experiences are essential, as well as for computer graphics and architectural visualization, where accuracy and visual fidelity are critical.

As shown in the following screenshots of the training process outputs in Colab, the Gaussian Splatting model transforms 2D image inputs and 3D geometry data into a good and interactive 3D scene. It does this by optimizing the placement and properties of Gaussian splats over 30,000 iterations, the model achieves a high-fidelity point cloud that captures the spatial and visual characteristics of the original environment. The output spalt.ply file then serves as an output that can easily be integrated into various platforms for visualization both in 2D and 3D.

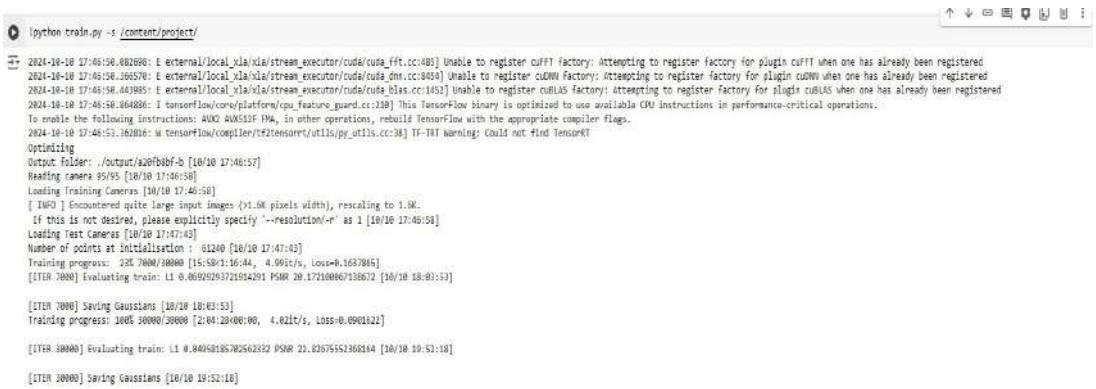

Figure 3.10: Model Training in Google Colab

3.3.4 Model Cleaning

Model cleaning in this project is done using a web application called SuperSplat. SuperSplat is an open-source web application that is designed for editing and inspecting 3D Gaussian splats. This helps clean the initial Gaussian splat points in order to create photorealistic scenes from point clouds. SuperSplat is an application that allows the manipulation of Gaussian splats directly in web browsers without the need for downloads or installations. It comes with a user-friendly and interactive interface that also features drag-and-drop functionality. This essentially allows the dragging of splat files and dropping them unto the workspace for editing. Through its Scene Manager panel, users can hide or remove splats, adjust their orientation, and perform various editing tasks efficiently. The application leverages the principles of 3D Gaussian Splatting, which involves converting images into 3D scenes by representing them as millions of particles or Gaussians. Each particle is defined by parameters such as position, scale, opacity, and color, allowing for a rich visual representation of the captured environment. SuperSplat not only facilitates the editing of these splats but also aids in optimizing them for better performance during rendering. It has become an essential tool for the development and creation of Gaussian splats for real-time rendering use cases. But this utility does not end with Gaussian splats but other spheres of 3D graphics as well [2][34][35].

Cleaning

Post-processing in the form of model cleaning using SuperSplat is conducted after the initial training period. This is to improve upon the quality of the generated 3D point cloud. This cleaning process is designed to refine the raw output of the model, which may include noise, unwanted artifacts, and outliers that reduce the visual quality of the 3D reconstruction.

Gaussian splatting, by its nature, sometimes produces splats that are not perfectly aligned with the real-world structure of the scene, especially in areas where the input images have less detail or coverage. SuperSplat addresses these issues by performing a series of filtering, smoothing, and optimization operations on the point cloud.

SuperSplat is very useful for refining point clouds generated from Gaussian splatting, where artifacts and noise can sometimes affect the visual clarity of the scene. Gaussian splatting represents images as millions of small Gaussian particles, each defined by parameters such as position, scale, color, and opacity. These particles collectively create a photorealistic 3D scene, but the raw output may include extraneous splats due to noisy data or imperfections in the model training. Through SuperSplat, users can clean up these artifacts, isolating and removing splats that do not contribute meaningfully to the final visualization. The ability to selectively hide or delete splats in real time allows for efficient refinement of the model's structure and detail, helping to create a cleaner, more visually cohesive point cloud.

In addition to removing noise, SuperSplat is then applied to he scene in order to smooth its surfaces using smoothing algorithms to the remaining splats to improve the continuity and cohesiveness of the model. This process involves blending adjacent splats based on their proximity, orientation, and color similarity, which helps to eliminate sharp edges and abrupt transitions between different parts of the model. By consolidating splats in areas with gradual changes, SuperSplat creates a more natural-looking surface, enhancing the visual realism of the point cloud. This smoothing step is particularly important in regions of the model that represent textures or complex surfaces, as it allows the reconstructed scene to appear more seamless and lifelike. The result is a point cloud with a more refined structure and minimized artifacts, making it suitable for applications requiring high visual quality.

In the figures of the cleaning process below, the effects of SuperSplat on the model refinement is shown. This highlights the transformation in model quality after the cleaning process. Within these images it can be clearly seen that there is a clear improvement in visual clarity, which shows a significant reduction in noise and enhancement in foreground elements. The cleaned output, stored as an updated splat.ply file, can be reloaded into a 3D viewer, where the improvements become immediately noticeable. SuperSplat thus plays a vital role in the Gaussian Splatting workflow by transforming a raw, sometimes noisy 3D point cloud into a polished and refined model that accurately represents the input scene.

In virtual reality environments, the cleaned models provide the high-fidelity visuals necessary for maintaining user immersion. Architectural visualization benefits from more accurate material rendering and improved spatial relationships, enabling better decision-making dur-

ing the design process. In digital heritage preservation, the enhanced detail retention and color accuracy ensure that cultural artifacts are documented with unprecedented precision. Additionally, the optimized point clouds require less computational overhead while delivering superior visual quality, making them particularly valuable for real-time applications where both performance and aesthetics are crucial considerations.

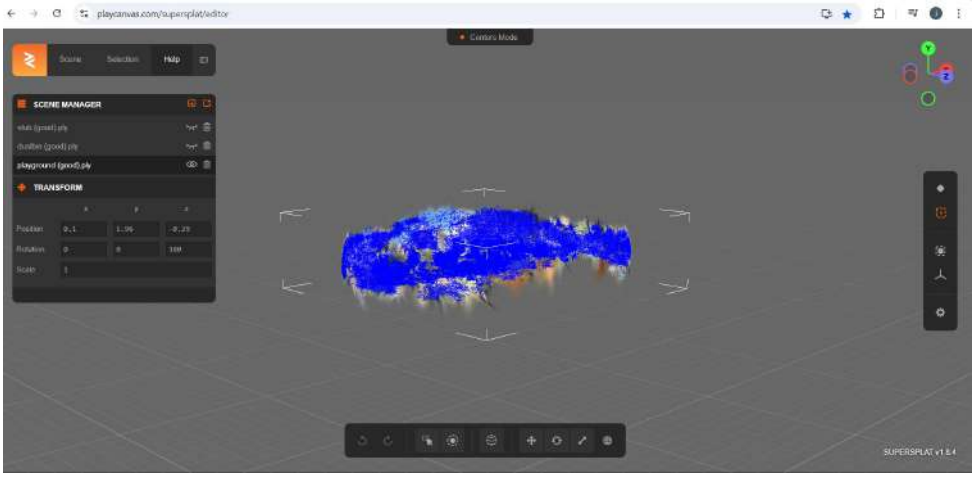

Figure 3.11: SuperSplat Workspace

Figure 3.12: SuperSplat Workspace without Points

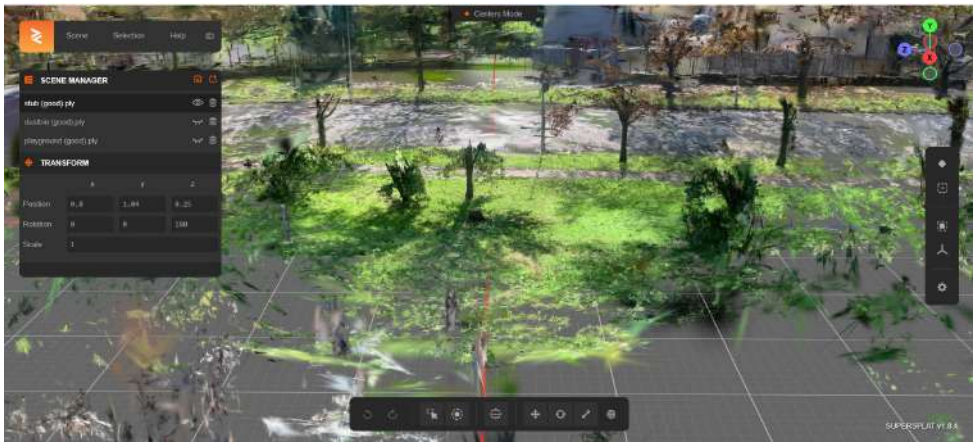

Figure 3.14: Cleaning of Sample Scene 2

Figure 3.15: Cleaning of Sample Scene 3

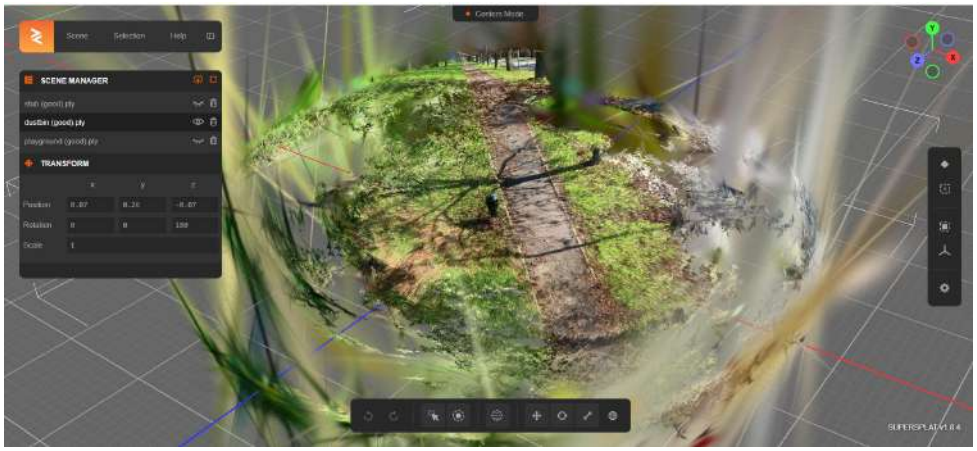

Figure 3.13: Cleaning of Sample Scene 1

3.3.5 VR Rendering

TODO: description of the setup. Thus, download of gracia app on a deesktop as well as on Meta Quest 2.with screenshots.

For the VR rendering of the 3D Gaussian splats, we used the Meta Quest 2 headset in

combination with the Gracia application as the primary software platform. This setup was chosen to provide an immersive experience in which users could explore the reconstructed 3D scenes as though they were physically inside them. The Gracia app was installed on both the desktop computer and the Meta Quest 2 headset, facilitating a smooth integration between the two devices. The setup is shown below:

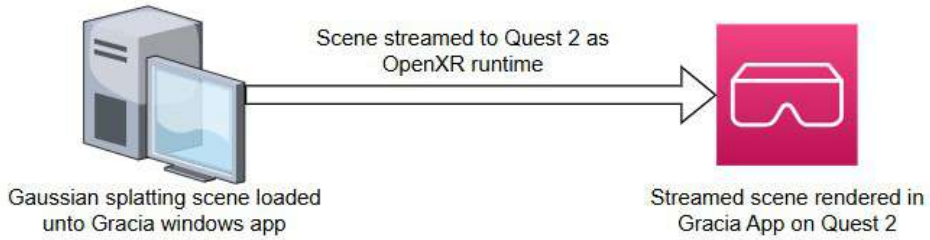

Figure 3.16: VR rendering setup

The primary computational load was managed on a desktop computer where the Gracia Windows application was installed. This desktop application allowed the Gaussian splat scene to be loaded, processed, and then streamed to the Quest 2 headset.

Using the OpenXR runtime, the desktop application streamed the scene directly to the Meta Quest 2, which had the Gracia app installed. This setup enabled the scene to be rendered in real-time within the Quest 2's VR environment, allowing users to view and explore the 3D scene interactively. With this configuration, users could move within the virtual environment, gaining an immersive experience as though they were physically present inside the reconstructed scene. The stream is enabled within the Quest 2 via questlink.

The figure illustrates the VR rendering workflow, showing how the scene is transferred from the desktop to the Quest 2 headset for visualization. This setup combines the powerful processing capabilities of the desktop with the portability and immersive features of the Quest 2, creating an efficient VR experience for inspecting detailed 3D Gaussian splat models.

The workflow involved loading the Gaussian splat scenes onto the desktop version of the Gracia app, which served as the primary processing and control hub. Once the scenes were prepared and optimized on the desktop, they were streamed to the Meta Quest 2 headset. This configuration allowed the user to view the scenes in real-time, experiencing the 3D space with full 6DOF (six degrees of freedom) movement. Within the VR environment, the user could navigate freely around the scene, inspect details from different angles, and gain an in-depth perspective of the 3D space generated by Gaussian splatting. This level of interactivity

provides a significant advantage for both evaluation and presentation of 3D reconstructions.

The combination of the Gracia app and the Meta Quest 2 headset provided an efficient and immersive platform for VR rendering. The desktop application handled the heavier processing requirements, reducing the load on the Quest 2, while the Quest 2's standalone VR capabilities ensured a seamless and interactive user experience. This setup was not only effective for showcasing high-fidelity 3D models but also allowed for real-time adjustments and scene manipulation, enhancing both the technical and experiential quality of the VR application. Below, several screenshots illustrate the setup process and the interface as seen within the Gracia app, both on the desktop and within the VR headset.

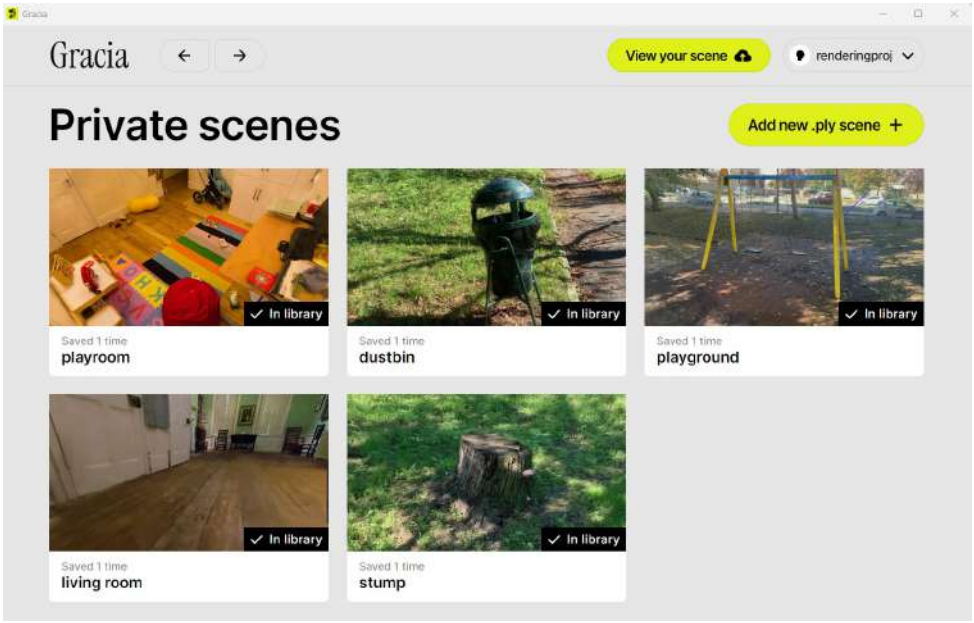

Figure 3.17: Gracia Desktop Application

Figure 3.18: Gracia launch via questlink on Quest 2

Figure 3.19: Gracia on Quest 2

3.4 Assessment

3.4.1 Single-User Visual Quality Assessment and User Experience

To evaluate the visual quality of the rendered scenes produced by the Gaussian splatting model, a subjective assessment approach was chosen. Given that this project is mainly single-user focused, the assessment will be based on direct observations and personal reflections gathered during VR interactions with the final rendered environment. This approach aims to capture detailed, qualitative insights into how well the model conveys realism, depth, and visual coherence within a VR setting. Assessing the results this way, shows the difference between the test images and the original images is the same, depending on how visible they are, which is affected by how people perceive things [17].

The assessment will focus on several key visual elements. First, color accuracy and vibrancy will be examined to determine how closely the rendering matches the original source images, as color plays a critical role in enhancing the sense of realism. Additionally, attention will be paid to the preservation of structural details within the scene, including an analysis of whether finer textures and intricate patterns are adequately retained. Rendering speed will

also be considered as this has a direct effect on great user experience in the context of VR rendering. This will seek to evaluate how well the model's output supports a realistic sense of three-dimensionality in the VR environment.

This assessment will take note of any visual artifacts, such as edge blending issues, blurring, or halo effects around objects. These observations will provide insights into potential areas for improvement in visual fidelity.

Chapter 4

Results and Discussion

The results of the entire development pipeline is presented in this chapter – from the image capture, preprocessing, training of the Gaussian splatting model to the ultimate goal of the creating an immersive VR experience in the Meta Quest 2 via the Gracia app. The entire system was designed to assess the effectiveness of Gaussian splatting in generating high-fidelity 3D reconstructions of scenes that are suitable for the real-time rendering in VR applications in the Meta Quest 2 headset. Each stage of the process including the image preprocessing, model model training, and post-processing for VR optimization, was carefully analyzed to evaluate the output quality and immersive experience provided by the final scene renderings.

In the sections that follow, the results of each stage of the system developed is presented and a discussion of the results is provided subsequently. The results are organized into three main areas: **Data Acquisition and Preprocessing**, **Model Training and Scene Cleaning**, and **VR Rendering and User Experience**. Each section describes the outcomes and evaluations for that particular stage of the pipeline. Key performance indicators include image resolution, visual coherence, fidelity of reconstructed scenes, and overall user experience in VR. By focusing on both quantitative and qualitative measures, these results demonstrate the pipeline's ability to transform high-definition images into realistic 3D environments that can be experienced in real time. To test the final scene in the Meta Quest 2, a screen recording was made of the experience within the VR headset and will provide the final result data to be analyzed and assessed. The focus is concentrated on how the lighting hits different surfaces and how the depth perception works and how that compares to only looking at still images. This presents insights into how well the system pipeline actually works for VR applications.

4.1 Data Acquisition and Preprocessing

COLMAP for 3D point generation and camera calibration, and how the processed data impacted the subsequent model training phase. obstacles faced during data preprocessing, such as issues with image alignment, point cloud quality, or data inconsistency. Explain how these issues were addressed, whether through adjustments to preprocessing scripts, parameter tuning, or additional data cleaning.

4.2 Model Training

The results of the model training for each of the sample scenes from the data acquisition stage is presented below. We can see that across board, the losst is generally low and even at 7,000 iterations for each of them the loss is low with impressive visual results which is sometimes identical to the results after 30,000 iterations.

Scene One (1):

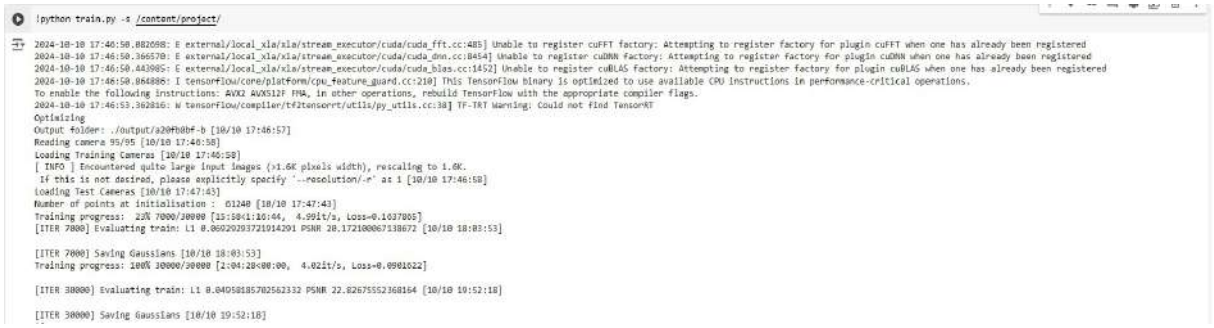

Figure 4.1: Scene 1 Training Results

From the above, it can be seen that the training ends with very low L1 loss and PSNR. The figure below compares the source image with an image from the scene after training to illustrate the quality of the training results.

Figure 4.2: Scene 1 Source Image (Left) vs Rendered Scene View (Right)

Scene Two (2):

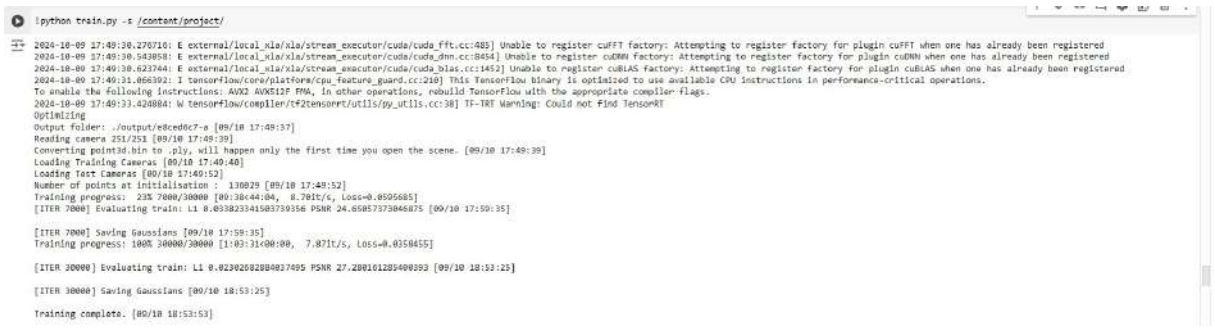

Figure 4.3: Scene 2 Training Results

Similar to Scene 1, it can be seen that for scene two the training losses are very well low. And this can be seen from the visual evidence of how closely similar a view from the scene is to the source image. The figure below compares both images.

Figure 4.4: Scene 2 Source Image (Left) vs Rendered Scene View (Right)

Scene Three (3):

Figure 4.5: Scene 3 Training Results

Finally, Scene 3, similarly has low training losses and this manifests in high visual quality and similarity between a view from the scene and the source image. The figure below compares both images.

Figure 4.6: Scene 3 Source Image (Left) vs Rendered Scene View (Right)

4.3 Scene Cleaning

The results of the Scene cleaning and enhancement process is presented for each of the three scenes. This is done with a focus on removing artifacts, refining details, and optimizing visual quality for VR rendering. This process aimed to improve user immersion and clarity within each virtual environment by selectively cleaning and refining specific elements that impact the overall quality. For each of the scenes presented below, the pre-cleaned scene is shown on the left while the cleaned version is shown on the right. The results of before and after effects of cleaning of each of the three (3) scenes is presented.

Figure 4.7: Scene 1 - Original (Left) vs Cleaned (Right)

Figure 4.8: Scene 2 - Original (Left) vs Cleaned (Right)

Figure 4.9: Scene 3 - Original (Left) vs Cleaned (Right)

4.4 VR Rendering and User Experience

Having prepared the image and video data into an appropriate form, trained the Gaussian Splatting model and cleaned the reconstructed scenes, the setup for the VR rendering in the Gracia App on the Meta Quest 2 comes into play. The trained scenes were each loaded into the scene library on the Gracia desktop App, after which the Meta Quest 2 was connected to the desktop computer and setup as the OpenXR runtime for the rendering of these scenes. Within the Quest 2 headset, connection was established through Quest link as shown below in order to make the device ready to receive the scene streams from the desktop App.

Figure 4.10: Quest 2 Connection to Desktop via Questlink

Below are some of the views of the rendering within the Gracia app in the Quest 2. We begin with Scene 1:

Scene 1

Figure 4.11: View 1 of Scene 1 in Gracia App on Quest 2

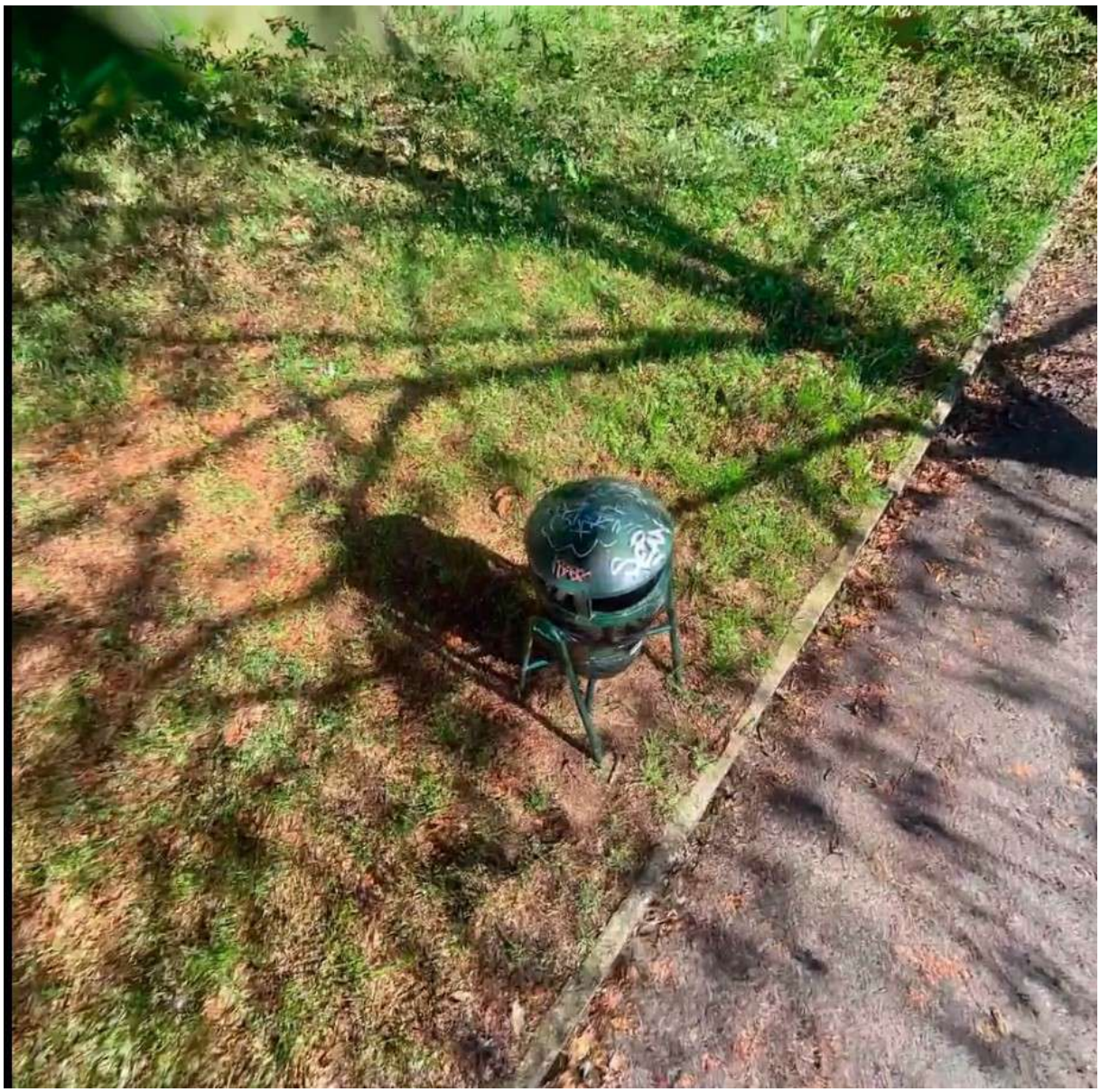

Figure 4.12: View 2 of Scene 1 in Gracia App on Quest 2

Discussion of Scene 1 Visual Quality and VR Experience

The VR scenes rendered on the Meta Quest 2 headset using the Gracia app demonstrate a compelling visual quality, capturing detailed textures and lighting. The Gaussian splatting approach effectively recreates a 3D environment from 2D image data, providing realistic scene depth and immersion. In the specific case of Scene 1, it can be seen that the environmental details like the pathway, grass, and shadows are well-preserved, making the virtual space appear lifelike. However, some artifacts—such as slight blurring around finer details—indicate limitations in resolution or training accuracy. This is not a perfect representation of the source scene but these results show an impressive representation of the scene and provides very good environment for immersive experiences. The user experience in all the VR environments were assessed based on immersion, clarity, and interactivity. The Quest 2 headset, coupled with the

Gracia app, offered smooth navigation and reasonable responsiveness. In the case of Scene 1, the scene could be explored from various perspectives, enhancing spatial awareness, which is crucial for ROV-related operations. The smoothness of the rendering while moving around within the environment provided for an impressive user experience.

Scene 2

Figure 4.13: View 1 of Scene 2 in Gracia App on Quest 2

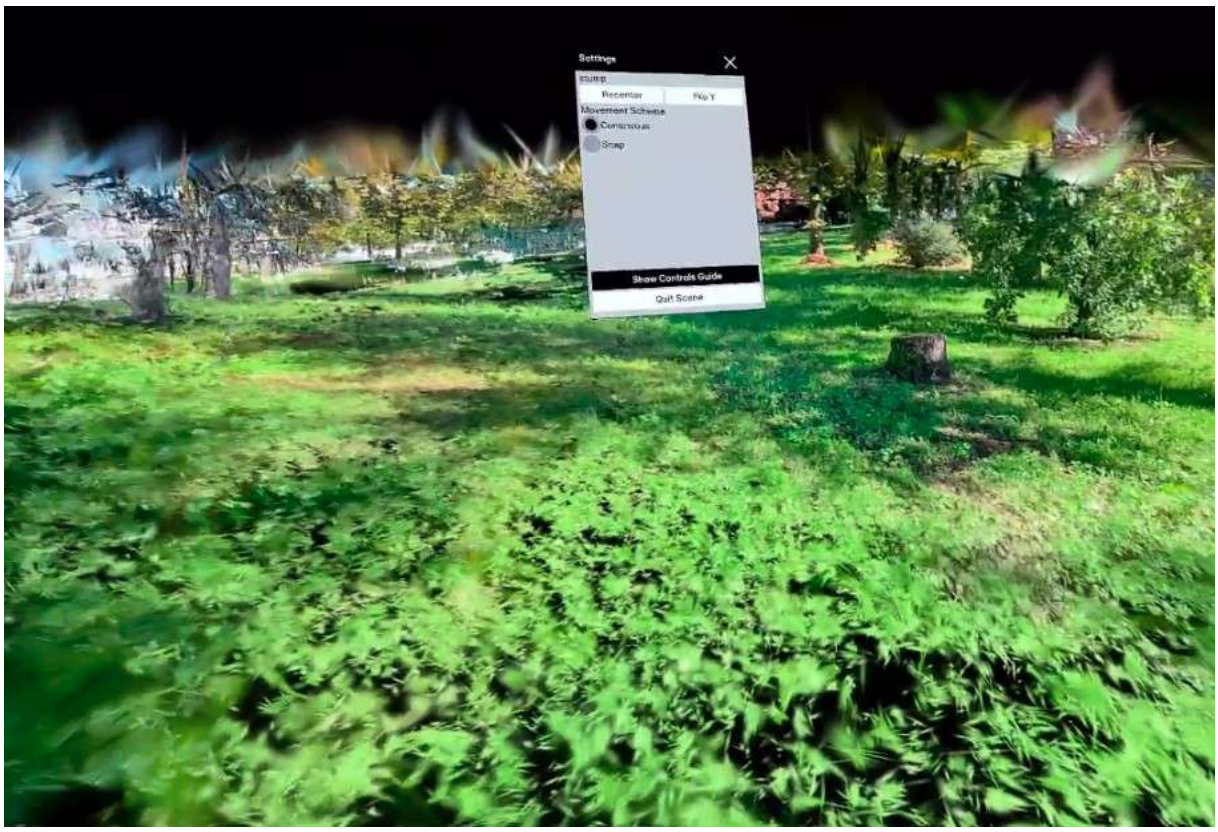

Figure 4.14: View 2 of Scene 2 in Gracia App on Quest 2

Discussion of Scene 2 Visual Quality and VR Experience

In the above scene (Scene 2), environmental details like the are well-preserved and look visually good, making the virtual space feel lifelike. This further proves the fact that the model training was good and approach offers an impressive representation of the scene. This creates a strong immersive environment ideal for applications in ROV navigation and remote PPDR operations.

In terms of user experience, this scene rendered in the Gracia App on Quest 2 was incredibly immersive. The Quest 2 headset, combined with the Gracia app, provided smooth navigation and excellent responsiveness. Scene exploration allowed for the observation from various perspectives, enhancing spatial awareness—an essential feature for ROV-related operations. The smooth rendering and movement within the environment contributed to a high-quality, immersive experience, demonstrating the effectiveness of this VR setup in mission-critical applications.

Scene 3

Figure 4.15: View 1 of Scene 3 in Gracia App on Quest 2

Figure 4.16: View 1 of Scene 3 in Gracia App on Quest 2

Discussion of Scene 2 Visual Quality and VR Experience

Scene 3 rendered on the Meta Quest 2 headset using the Gracia app showcased an impressive visual quality achieved by Gaussian splatting, effectively capturing real-world details in a virtual setting. In the attached Scene images, realistic features and great details of parts of the environment such as the grass textures and details of pathway is observed. The VR experience was similarly as immersive for this scene as it was for the previous two scenes.

4.5 Discussion of Results and Proposed Framework

The results of this project demonstrate the potential of using Gaussian splatting techniques for modeling and rendering virtual scenes for remotely operated vehicles (ROVs) in Public

Protection and Disaster Relief (PPDR) applications. The pipeline successfully allowed for the capture, preprocessing, and rendering of 3D scenes in virtual reality (VR), viewed through the Meta Quest 2 headset. The immersive nature of VR offers significant advantages for operators in disaster scenarios, as it provides a realistic sense of the environment from a safe distance.

The resulsts as shown above, illustrate a robust pipeline that produces results good enough to ensure that remote operators of ROVs have an immersive experience when dealing with situations that require the use of such vehicles. It is also clear to see that the visual quality of the final scenes rendered in the HMD is of very high quality with very little and not noticeable differences between the rendered scenes and the original images and videos. This speaks to the pipeline that works well and produces excellent resutlts.

In spite of the great results, some hurdles where faced. During model training, the use of Gaussian splatting proved effective in reconstructing detailed 3D scenes, though certain limitations were noted. A key challenge was that the original Gaussian splatting implementation did not support higher-resolution inputs such as 4K images, which would improve the clarity and accuracy of scene reconstruction. Additionally, the strong dependency of Gaussian splatting on COLMAP for feature matching and 3D point cloud generation restricted the flexibility of the pipeline. This tight coupling limited the ability to integrate other 3D scanning methods, such as LiDAR, which could provide more accurate or efficient results.

Data preprocessing and model cleaning were also significant hurdles in the project. The need for manual intervention during the model cleaning process resulted in delays, as the Gaussian splatting algorithm requires substantial adjustment to remove artifacts and optimize visual quality. Moreover, capturing scenes without focal objects, such as barren landscapes or complex environments with few distinguishable features, was challenging. This limitation indicates a need for more effective methods for capturing diverse types of environments.

Regardless, the pipeline successfully demonstrated how Gaussian splatting could be applied in real-time VR rendering for ROV operators. The rendered scenes displayed high realism, offering valuable insights for disaster management teams. This proves the feasibility of integrating such a system within the PPDR framework to enhance operators' situational awareness in disaster areas.

4.5.1 Proposed Framework for Integration with 5G Technology for ROVs

To integrate Gaussian splatting with a private 5G network to enhance the capabilities of remotely operated vehicles (ROVs) in mission-critical operations, the following key components must be taken into consideration:

Edge Computing and Local 5G Transceivers: By deploying edge computing nodes with a local 5G transceiver on-site, data processing can occur in real time, reducing latency and enabling quick decision-making for ROV operators. This setup ensures that large datasets from 3D scene captures are processed locally before being transmitted to remote systems.

High-Resolution Scene Capture: The use of high-resolution cameras and advanced imaging systems will enhance the quality of 3D models generated for VR rendering. These high-quality inputs will provide more accurate and immersive virtual environments, ensuring that operators have the best possible data for their assessments.

Telco Cloud for Offline VR Model Reconstruction: The telco cloud can be used for offline VR model reconstruction, allowing for more complex computations and detailed scene rendering without overloading the edge computing nodes. This offloading to the cloud would improve the overall system's performance by enabling more sophisticated VR model training, which can then be streamed to the head-mounted display (HMD) for real-time viewing.

Within the above stated system, the high-speed transmission of data, enhanced scene modeling, and real-time VR rendering will be improved, providing ROV operators with the tools they need to assess disaster scenarios efficiently. The use of 5G technology ensures that the system can handle large datasets with minimal latency, improving response times and operational effectiveness in remote locations.

Chapter 5

Conclusion

This project focused on the creation of a complete pipeline for capturing, modeling, and rendering scenes in virtual reality (VR), aimed specifically at improving the use of remotely operated vehicles (ROVs) within the Public Protection and Disaster Relief (PPDR) framework. The goal was to provide operators with a more realistic and immersive experience when assessing disaster sites through VR. The project successfully met its objectives, including the development of a model training pipeline for VR applications, the rendering of these models on a head-mounted display (HMD) like the Meta Quest 2, and the proposal of a framework to integrate this system with 5G technology for ROVs used in PPDR scenarios.

Challenges Encountered

Several challenges were encountered during the project, particularly with the use of the Gaussian splatting technique. One notable limitation was that the original Gaussian splatting implementation does not support high-resolution images, such as 4K inputs. The use of higher resolution images could significantly improve the quality of the reconstructed 3D scenes, allowing for more accurate and detailed visualizations. Another issue arose from the tight coupling between Gaussian splatting and COLMAP. The original implementation expects the outputs of COLMAP as input, which restricted the flexibility of using alternative sources of 3D point clouds, such as LiDAR or other 3D scanning technologies.

Additionally, the model cleaning process required significant manual effort, which created delays in the development timeline. Despite attempts to automate parts of the process, model cleaning remained a bottleneck, adding extra time to the project. Another challenge was the difficulty in capturing scenes without focal objects, such as empty environments or areas with minimal structure. Capturing such scenes proved challenging, as the Gaussian splatting method performs best with distinct points of reference or key objects in the environment. This

suggests a need for more effective image and video capture techniques to handle these types of scenes.

Proposals for Future Work

To address the limitations encountered, there are several proposals for future work that could improve the pipeline. First, using a modified or improved version of the Gaussian splatting implementation would help overcome the restrictions of the original code, particularly regarding its input requirements and support for higher resolution images. Future implementations could be more flexible and capable of working with various input types, such as LiDAR data or 360-degree images, which would improve the quality of the 3D point cloud generation.

Furthermore, replacing traditional cameras and COLMAP with 360-degree cameras and LiDAR for 3D point cloud generation could offer more accurate data. This would allow for better modeling of complex environments, especially those with sparse or challenging features. These technologies would also reduce the reliance on manual intervention during the data capture and model cleaning phases, streamlining the overall process and improving the efficiency of the pipeline.

This project successfully demonstrated the feasibility of integrating Gaussian splatting, VR rendering, and ROV technology in the PPDR context. While the pipeline met its objectives, there is room for improvement in terms of the flexibility of the input data, the automation of model cleaning, and the handling of complex scenes. Future enhancements to the system, including the use of more advanced imaging technologies and improved algorithms, will help address these challenges and push the boundaries of VR applications in disaster response.

Bibliography

[1] 3rd Generation Partnership Project (3GPP). TR 22.891: Study on new services and markets technology enablers. Technical Report TR 22.891, 3GPP, 2016.

[2] 80.lv. New Open-Source Browser-Based 3D Gaussian Splat Editor, 2023. Accessed: 2023-11-10. URL: https://80.lv/articles/new-open-source-browser-based-3d-gaussian-splat-editor/.

[3] K L Bhawan. Public Protection and Disaster Relief (PPDR) Communication System.

[4] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon Mesh Processing*. CRC Press, 2010.

[5] Norbert Bus and Tamy Boubekeur. Double hierarchies for directional importance sampling in monte carlo rendering. *Journal of Computer Graphics Techniques (JCGT)*, 6(3):25–37, 2017. URL: http://jcgt.org/published/0006/03/02.

[6] Anurag Dalal, Daniel Hagen, Kjell G. Robbersmyr, and Kristian Muri Knausgård. Gaussian Splatting: 3D Reconstruction and Novel View Synthesis, a Review. *IEEE Access*, 12:96797–96820, 2024. arXiv:2405.03417 [cs]. URL: http://arxiv.org/abs/2405.03417, doi:10.1109/ACCESS.2024.3408318.

[7] Corbin Davenport. The Meta Quest 2 Is Returning to Its Original Price, June 2023. Accessed: 2023-11-10. URL: https://www.howtogeek.com/896688/the-meta-quest-2-is-returning-to-its-original-price/.

[8] Mohamed Debbagh. Neural radiance fields (nerfs): A review and some recent developments, 2023. URL: https://arxiv.org/abs/2305.00375, arXiv:2305.00375.

[9] Sayed Ishaq Deliry and Uğur Avdan. Accuracy of Unmanned Aerial Systems Photogrammetry and Structure from Motion in Surveying and Mapping: A Re-

view. *Journal of the Indian Society of Remote Sensing*, 49(8):1997–2017, August 2021. URL: https://link.springer.com/10.1007/s12524-021-01366-x, doi:10.1007/s12524-021-01366-x.

[10] Viktor Enfeldt and Prashant Goswami. A polarizing filter function for real-time rendering. *Journal of Computer Graphics Techniques (JCGT)*, 10(2):59–82, 2021. URL: http://jcgt.org/published/0010/02/03/.

[11] Ben Fei, Jingyi Xu, Rui Zhang, Qingyuan Zhou, Weidong Yang, and Ying He. 3d gaussian splatting as new era: A survey. *IEEE Transactions on Visualization and Computer Graphics*, page 1–20, 2024. URL: http://dx.doi.org/10.1109/TVCG.2024.3397828, doi:10.1109/tvcg.2024.3397828.

[12] Matthew Fridovich-Keil, Qinhong Chen Yu, Benjamin Recht Tancik, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[13] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. NeRF: Neural Radiance Field in 3D Vision, A Comprehensive Review, November 2023. arXiv:2210.00379 [cs]. URL: http://arxiv.org/abs/2210.00379.

[14] Zohreh GharehTappeh and Qingjin Peng. Simplification and unfolding of 3d mesh models: review and evaluation of existing tools. *Procedia CIRP*, 100:121–126, 06 2021. doi:10.1016/j.procir.2021.05.023.

[15] Gracia. Gracia - Spatial Computing Application, 2024. Accessed: 2024-11-10. URL: https://www.gracia.ai/.

[16] Taina Hult and Jyri Rajamäki. ICT Integration of Public Protection and Disaster Relief (PPDR): Mobile Object Bus Interaction (MOBI) Research and Development Project. *Computer Engineering*.

[17] Iman Hussein AL-Qinani. A review paper on image quality assessment techniques. 6, 08 2019.

[18] International Telecommunication Union - Radiocommunication Sector (ITU-R). M.2083-0: IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond. Recommendation M.2083-0, ITU-R, Geneva, Switzerland, 2015.

[19] Pekka Isto, Tapio Heikkilä, Aarne Mämmelä, Mikko Uitto, Tuomas Seppälä, and Jari Ahola. 5g based machine remote operation development utilizing digital twin. *Open Engineering*, 10:265–272, 05 2020. doi:10.1515/eng-2020-0039.

[20] A. Kaufman, D. Cohen, and R. Yagel. Volume graphics. *IEEE Computer*, 26(7):51–64, July 1993.

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering, August 2023. arXiv:2308.04079 [cs]. URL: http://arxiv.org/abs/2308.04079.

[22] Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. Neural point catacaustics for novel-view synthesis of reflections. *ACM Transactions on Graphics (SIGGRAPH Asia Conference Proceedings)*, 41(6):201, 2022. URL: http://www-sop.inria.fr/reves/Basilic/2022/KLRJD22.

[23] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. *Computer Graphics Forum*, 40(4):29–43, 2021. doi:10.1111/cgf.14339.

[24] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1440–1449, 2021.

[25] Xiaoyu Li, Qi Zhang, Di Kang, Weihao Cheng, Yiming Gao, Jingbo Zhang, Zhihao Liang, Jing Liao, Yan-Pei Cao, and Ying Shan. Advances in 3d generation: A survey, 2024. URL: https://arxiv.org/abs/2401.17807, arXiv:2401.17807.

[26] You Li and Javier Ibanez-Guzman. Lidar for Autonomous Driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Processing Magazine*, 37(4):50–61, July 2020. arXiv:2004.08467 [cs]. URL: http://arxiv.org/abs/2004.08467, doi:10.1109/MSP.2020.2973615.

[27] Yufei Lin. The review of modeling and rendering 3d environment: A survey. *Highlights in Science, Engineering and Technology*, 93:304–315, 05 2024. doi:10.54097/hpw9dz69.

[28] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view syn-

thesis, 2020. URL: https://arxiv.org/abs/2003.08934, arXiv:2003.08934.

[29] Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter Battaglia. PolyGen: An autoregressive generative model of 3D meshes. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7220–7229. PMLR, 13–18 Jul 2020. URL: https://proceedings.mlr.press/v119/nash20a.html.

[30] Floriano Neto, Jorge Granjal, and Vasco Pereira. A Survey on Security Approaches on PPDR Systems Toward 5G and Beyond. *IEEE Access*, 10:117118–117140, 2022. URL: https://ieeexplore.ieee.org/document/9930517/, doi:10.1109/ACCESS.2022.3217223.

[31] Duc Nguyen, Tran Thuy Hien, and Truong Thu Huong. A subjective quality evaluation of 3d mesh with dynamic level of detail in virtual reality, 2024. URL: https://arxiv.org/abs/2406.06888, arXiv:2406.06888.

[32] S. Patil and B. Ravi. Voxel-based representation, display and thickness analysis of intricate shapes. volume 2005, pages 6 pp.–, 01 2006. doi:10.1109/CAD-CG.2005.86.

[33] Tiago Pessoa, Raul Medeiros, Thiago Nepomuceno, Gui-Bin Bian, V.H.C. Albuquerque, and Pedro Pedrosa Filho. Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *IEEE Access*, PP:1–1, 10 2018. doi:10.1109/ACCESS.2018.2874767.

[34] PlayCanvas Contributors. Supersplat: Open-source browser-based 3D Gaussian splat editor, 2023. Accessed: 2023-11-10. URL: https://github.com/playcanvas/supersplat.

[35] PlayCanvas Team. *Gaussian Splatting - PlayCanvas Developer Documentation*, 2023. Accessed: 2023-11-10. URL: https://developer.playcanvas.com/user-manual/graphics/gaussian-splatting/.

[36] PlayCanvas Team. Screenshot of the SuperSplat Editor Platform, 2023. Accessed: 2023-11-10. URL: https://playcanvas.com/supersplat/editor.

[37] Radiance Fields Team. Supersplat Platform, 2023. Accessed: 2023-11-10. URL: https://radiancefields.com/platforms/supersplat.

[38] Emma Raymer, Áine MacDermott, and Alex Akinbi. Virtual reality forensics: Forensic analysis of meta quest 2. *Forensic Science International: Digital Investigation*, 47:301658, 2023. URL: https://www.sciencedirect.com/science/article/pii/S2666281723001774, doi:10.1016/j.fsidi.2023.301658.

[39] Dirk Rieke-Zapp and Santiago Royo. *Structured Light 3D Scanning*. 01 2017. doi:10.5040/9781641899444.019.

[40] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, Las Vegas, NV, USA, June 2016. IEEE. URL: http://ieeexplore.ieee.org/document/7780814/, doi:10.1109/CVPR.2016.445.

[41] L. A. Shirman and C. H. Sequin. Local surface interpolation with bézier patches. *Computer Aided Geometric Design*, 4(4):279–295, 1987.

[42] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[43] Zhengren Wang. 3D Representation Methods: A Survey, October 2024. arXiv:2410.06475 [cs]. URL: http://arxiv.org/abs/2410.06475.

[44] X. Ma, V. Hegde, and L. Yolyan. *3D Deep Learning with Python: Design and develop your computer vision model with 3D data using PyTorch3D and more.* Packt Publishing Ltd, 2022.

[45] Wang Yifan, Feliciano Serena, Shaofei Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019.

[46] Huang Zhang, Changshuo Wang, Shengwei Tian, Baoli Lu, Liping Zhang, Xin Ning, and Xiao Bai. Deep Learning-based 3D Point Cloud Classification: A Systematic Survey and Outlook. *Displays*, 79:102456, September 2023. arXiv:2311.02608 [cs]. URL: http://arxiv.org/abs/2311.02608, doi:10.1016/j.displa.2023.102456.