Bachelor Thesis

Multi-Camera Dynamic Scene Reconstruction with 4D Gaussian Splatting

Androulla Bekri

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

April 2026

UNIVERSITY OF CYPRUS

Faculty of Pure and Applied Sciences

DEPARTMENT OF COMPUTER SCIENCE

Multi-Camera Dynamic Scene Reconstruction with 4D Gaussian Splatting

Androulla Bekri

Supervisor

Dr. Yiorgos Chrysanthou

The Thesis was submitted in partial fulfillment of the requirements for obtaining the Computer Science degree at the Department of Computer Science of the University of Cyprus

April 2026

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Yiorgos Chrysanthou, for giving me the opportunity to work on this thesis topic and for his continuous guidance and support throughout the project. His advice, encouragement, and availability were very important during the development of this work.

I would also like to thank Mr. Dimitris Kontis for his valuable assistance with the multi-view camera system. His help was important during the data acquisition process, and I am also grateful for the additional dataset he provided, which allowed me to perform further experiments and extend the evaluation of the reconstruction pipeline.

ABSTRACT

This thesis focuses on 3D scene reconstruction using multi-view imaging techniques and 4D Gaussian Splatting. The main objective is to develop a complete pipeline that starts from synchronized image acquisition using multiple Intel RealSense cameras and results in the generation of 3D representations and rendered videos.

Initially, a suitable Ubuntu-based environment was installed and configured, including all necessary tools and libraries such as COLMAP, Python, OpenCV, PyTorch, and the Intel RealSense SDK. Data acquisition was carried out at the CYENS Centre of Excellence, where multiple Intel RealSense cameras were used to capture synchronized RGB, depth, and infrared data from different viewpoints.

Following data acquisition, the captured images were converted, organized, and preprocessed into a structure suitable for COLMAP. COLMAP was then used for camera pose estimation and sparse 3D reconstruction, producing registered camera views and sparse point cloud representations. The resulting camera parameters and registered images were further processed and converted into transformation files compatible with the 4D Gaussian Splatting framework.

In addition to the local development environment, part of the computational workflow was executed on a High Performance Computing (HPC) cluster using Slurm batch jobs. This allowed the use of shared storage for large datasets and GPU resources for the training and rendering stages. Among the tested configurations, the best overall result was obtained using 11 cameras with 39 frames per camera. From a total of 429 input images, COLMAP successfully registered 388 images, producing a sparse reconstruction with 25,656 points and a mean reprojection error of 1.159901 pixels. The trained 4D Gaussian Splatting model generated a 14-second rendered video with high visual quality, demonstrating that this configuration provided the most effective balance between viewpoint diversity and temporal sampling.

The study also investigates the impact of different parameters, such as the number of cameras, the number of frames per camera, and the balance of frame distribution across cameras, on reconstruction quality. The results demonstrate that an appropriate balance between viewpoint diversity and temporal sampling significantly affects both rendering quality and geometric accuracy.

The experiments demonstrate that balanced multi-camera configurations with sufficient temporal coverage produce more stable reconstructions and improved rendering quality compared to sparse or imbalanced datasets. Overall, the proposed workflow demonstrates the practical integration of COLMAP and 4D Gaussian Splatting for dynamic multi-view scene reconstruction using synchronized multi-camera capture systems.

Keywords: 3D Reconstruction, Gaussian Splatting, COLMAP, RealSense, Multi-view Imaging, 4D Gaussian Splatting, HPC

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

3D	Three-Dimensional
4D	Four-Dimensional
COLMAP	Structure-from-Motion and Multi-View Stereo Reconstruction Software
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
D-NeRF	Dynamic Neural Radiance Fields
FPS	Frames Per Second
GPU	Graphics Processing Unit
HPC	High Performance Computing
IR	Infrared
JSON	JavaScript Object Notation
MVS	Multi-View Stereo
NeRF	Neural Radiance Fields
PLY	Polygon File Format
RGB	Red, Green, Blue
SfM	Structure from Motion
Slurm	Simple Linux Utility for Resource Management
SIFT	Scale-Invariant Feature Transform
SDK	Software Development Kit

1 Introduction

In recent years, 3D reconstruction has become an essential field in computer vision, enabling the creation of digital representations of real-world objects and environments. The ability to reconstruct 3D scenes from 2D images has numerous applications, including virtual reality, augmented reality, robotics, medical imaging, and digital content creation.

Traditional approaches to 3D reconstruction rely on multi-view geometry techniques such as Structure from Motion (SfM) and Multi-View Stereo (MVS). These methods estimate camera positions and reconstruct sparse or dense point clouds based on feature matching across multiple images. While effective, such approaches often struggle with complex lighting conditions, textureless regions, and dynamic scenes.

Recently, neural rendering methods, such as Neural Radiance Fields (NeRF), have significantly improved the quality of 3D reconstruction by learning continuous scene representations. However, these methods can be computationally expensive and slow to train and render. More recently, Gaussian Splatting has emerged as a powerful alternative, offering high-quality real-time rendering by representing the scene as a collection of anisotropic Gaussians [1].

This thesis investigates the use of multi-view image data captured with Intel RealSense cameras for 3D and 4D scene reconstruction. The work follows a complete practical pipeline, starting from synchronized data acquisition and image preprocessing, continuing with camera pose estimation and sparse reconstruction using COLMAP, and finally training a 4D Gaussian Splatting model for scene representation and rendering [2]. In addition, part of the computational workflow was executed on a High Performance Computing (HPC) cluster using Slurm, allowing the processing of larger datasets and GPU-based training.

1.1 Motivation / Overview

This work focuses on the implementation of a multi-view 3D reconstruction pipeline using modern techniques in computer vision and neural rendering. A custom dataset is created by capturing images from multiple viewpoints using Intel RealSense cameras. The use of multiple cameras enables improved scene coverage and reconstruction accuracy.

The motivation behind this project is to examine how real multi-camera data can be transformed into a format suitable for modern neural rendering methods. In practical reconstruction scenarios, the quality of the final result depends not only on the reconstruction algorithm itself, but also on the quality, organization, synchronization, and distribution of the captured images. For this reason, this thesis also considers the preprocessing and dataset preparation stages as essential

parts of the reconstruction pipeline.

Furthermore, the use of 4D Gaussian Splatting introduces the possibility of representing scenes with a temporal component. This is particularly relevant for datasets captured from multiple synchronized viewpoints, where both spatial coverage and temporal sampling can influence the quality of the reconstructed scene and the generated renderings.

1.2 Purpose of the Work

The main objective of this project is to investigate the impact of camera configuration and dataset characteristics on the quality of the reconstructed 3D scene. In particular, the effect of varying the number of cameras and the number of captured images is examined in terms of rendering quality and geometric consistency.

More specifically, the work aims to develop and evaluate a complete pipeline that converts raw multi-camera image data into a structure suitable for COLMAP and 4D Gaussian Splatting. This includes the organization of captured images, the estimation of camera poses, the generation of sparse reconstructions, the creation of transformation files required by the training framework, and the execution of the training process on GPU resources.

A further goal is to identify how different dataset configurations affect reconstruction stability. For this reason, several experimental setups are considered, including datasets with different numbers of cameras, different numbers of frames per camera, and imbalanced frame distributions.

1.3 Methodology and Contributions

In this work, a reconstruction pipeline is implemented using COLMAP for camera pose estimation and 4D Gaussian Splatting for scene representation and rendering [1]. Additionally, a custom Python script is developed to automatically organize and preprocess the captured images [3], ensuring proper formatting and synchronization.

The configuration and operation of the cameras follow the official Intel RealSense guidelines [4]. After acquisition, the captured data are reorganized into a COLMAP-compatible structure, and COLMAP is used to extract features, match images, estimate camera poses, and generate sparse 3D reconstructions. The registered images and camera parameters are then converted into transformation files compatible with the 4D Gaussian Splatting framework.

An additional part of the methodology involves the use of an HPC cluster environment. The computationally demanding stages, including COLMAP reconstruction and 4D Gaussian Splat-

ting training, are executed through Slurm batch scripts. This enables the use of shared storage for large datasets and GPU nodes for the training stage.

The main contributions of this work include:

- Development of a multi-camera data acquisition setup using Intel RealSense cameras.
- Implementation of an automated preprocessing pipeline for organizing captured images.
- Preparation of COLMAP-compatible datasets from multi-camera image sequences.
- Estimation of camera poses and sparse reconstructions using COLMAP.
- Conversion of COLMAP outputs into transformation files suitable for 4D Gaussian Splatting training.
- Execution of the reconstruction and training pipeline on an HPC cluster using Slurm.
- Evaluation of reconstruction quality under different dataset configurations.

1.4 Outline / Document Structure

The remainder of this document is organized as follows. Chapter 2 presents the theoretical background and related work, including the main concepts related to multi-view reconstruction, COLMAP, and Gaussian Splatting. Chapter 3 describes the system architecture and implementation details, including data acquisition, preprocessing, COLMAP reconstruction, and 4D Gaussian Splatting training. Chapter 4 presents the experimental setup and discusses the results obtained from different camera and frame configurations. Finally, Chapter 5 concludes the work and discusses limitations and possible future directions.

2 Background and Related Work

This chapter presents the theoretical background and related technologies required for understanding the methodology followed in this thesis. The main concepts include multi-view 3D reconstruction, Structure from Motion, camera pose estimation, neural rendering, Gaussian Splatting, and the use of Intel RealSense cameras for multi-modal data acquisition.

2.1 3D Reconstruction

3D reconstruction is a fundamental task in computer vision that aims to recover the three-dimensional structure of a scene or object from two-dimensional image data. The goal is to create a digital representation that approximates the geometry and appearance of the real-world scene.

Depending on the available data and the reconstruction method, the output may take different forms, such as sparse point clouds, dense point clouds, meshes, volumetric representations, or neural scene representations. Sparse point clouds provide a set of 3D points corresponding to matched visual features, while denser representations attempt to recover a more complete surface or appearance model of the scene.

In this thesis, 3D reconstruction is approached through a multi-view imaging pipeline, where images are captured from several viewpoints using multiple cameras. These images are then processed to estimate camera positions and reconstruct the observed scene.

2.2 Multi-View Imaging

Multi-view imaging refers to the process of capturing a scene from multiple viewpoints. Instead of relying on a single image, multiple images provide additional geometric information, allowing the system to infer depth and spatial structure more accurately.

The main advantage of a multi-view setup is that the same scene points can be observed from different camera positions. By identifying correspondences between images, it becomes possible to estimate both the relative camera poses and the 3D positions of the observed points. This principle is widely used in photogrammetry, Structure from Motion, and modern neural rendering pipelines.

In the context of this thesis, multiple Intel RealSense cameras are used to capture synchronized data from different viewpoints. This setup improves scene coverage and provides richer information for the reconstruction process compared to a single-camera setup.

2.3 Structure from Motion

Structure from Motion (SfM) is a classical computer vision technique used to estimate the 3D structure of a scene from a set of overlapping images. At the same time, SfM estimates the camera poses, meaning the position and orientation of each camera view.

The typical SfM pipeline consists of several stages. First, local visual features are detected in each image. Then, these features are matched across different images in order to find correspondences. Based on these correspondences, the system estimates the relative camera positions and triangulates 3D points. The final result is usually a sparse 3D point cloud together with the estimated camera parameters.

SfM is an important step in many reconstruction pipelines because accurate camera poses are required before more advanced scene representations can be trained or rendered. In this work, COLMAP is used to perform this process.

2.4 COLMAP

COLMAP is a widely used Structure from Motion and Multi-View Stereo software package. It provides tools for feature extraction, feature matching, camera pose estimation, sparse reconstruction, and dense reconstruction.

In this thesis, COLMAP is used mainly for sparse reconstruction and camera pose estimation. The input to COLMAP consists of RGB images captured from multiple Intel RealSense cameras. COLMAP extracts visual features from the images, matches them across different views, and reconstructs a sparse 3D point cloud. At the same time, it estimates the camera intrinsics and extrinsics required for later processing.

The output of COLMAP is essential for the 4D Gaussian Splatting stage, since the training framework requires camera poses and image information in order to learn a scene representation. Therefore, COLMAP acts as an intermediate stage between data acquisition and neural rendering.

2.5 Neural Rendering

Neural rendering is a research area that combines computer graphics and machine learning in order to synthesize novel views of a scene. Instead of relying only on traditional geometric models, neural rendering methods learn scene representations from images and use them to generate new viewpoints.

One of the most influential approaches in this area is Neural Radiance Fields (NeRF). NeRF represents a scene as a continuous function that maps 3D position and viewing direction to color and density. This allows the generation of high-quality novel views. However, NeRF-based methods are often computationally expensive and can require long training and rendering times.

The limitations of NeRF motivated the development of more efficient approaches, such as Gaussian Splatting, which aims to achieve high-quality rendering with significantly faster performance.

2.6 Gaussian Splatting

Gaussian Splatting is a modern scene representation and rendering technique that models a 3D scene as a collection of anisotropic Gaussian primitives. Each Gaussian represents part of the scene and contains properties such as position, scale, rotation, opacity, and color.

During rendering, these Gaussians are projected onto the image plane and blended to synthesize a new view. This representation enables efficient rendering while preserving high visual quality. Compared to traditional neural rendering methods, Gaussian Splatting can offer faster training and real-time or near real-time rendering performance.

In the context of this thesis, Gaussian Splatting is important because it provides the foundation for the 4D Gaussian Splatting framework used in the experimental pipeline.

2.7 4D Gaussian Splatting

4D Gaussian Splatting extends the idea of 3D Gaussian Splatting by introducing a temporal component. Instead of representing only a static scene, 4D Gaussian Splatting aims to model scenes that change over time. This makes it suitable for dynamic scenes and multi-frame datasets.

The fourth dimension corresponds to time. Each image frame is associated with a normalized time value, allowing the model to learn how the scene evolves across frames. This is particularly relevant for datasets captured using synchronized cameras, where multiple viewpoints are available over a sequence of frames.

In this thesis, the captured multi-camera images are converted into a format compatible with the 4D Gaussian Splatting framework. This includes the generation of transformation files containing image paths, camera-to-world matrices, and normalized time values.

2.8 Intel RealSense Cameras

Intel RealSense cameras are depth-sensing devices capable of capturing RGB images, depth maps, and infrared data [5]. These modalities are useful for computer vision applications because they provide both appearance and geometric information about the scene.

RGB images provide color and texture information, while depth maps contain distance measurements from the camera to the observed surfaces. Infrared images can provide additional information that may be useful under different lighting conditions. The combination of these data streams makes RealSense cameras suitable for 3D reconstruction, object scanning, and avatar-related applications.

In this work, Intel RealSense cameras are used to capture synchronized multi-view data. The captured RGB images are primarily used for COLMAP and 4D Gaussian Splatting, while the availability of depth and infrared data provides additional flexibility for future extensions of the pipeline.

2.9 Related Work

The field of 3D reconstruction has evolved from classical geometric methods to modern neural rendering approaches. Traditional methods, such as Structure from Motion and Multi-View Stereo, rely on feature matching and geometric triangulation to reconstruct sparse or dense 3D representations. These methods remain highly relevant because they provide accurate camera pose estimation and are often used as preprocessing steps for newer techniques.

Neural rendering methods, such as NeRF, introduced a different approach by learning continuous scene representations from posed images. Although these methods achieved high-quality novel view synthesis, they are often computationally expensive. Gaussian Splatting addresses some of these limitations by representing scenes with explicit Gaussian primitives, enabling efficient optimization and faster rendering.

4D Gaussian Splatting further extends this representation to dynamic scenes by incorporating temporal information. This makes it suitable for multi-frame datasets and motivates its use in this thesis. The present work combines classical reconstruction tools, such as COLMAP, with a modern 4D Gaussian Splatting framework in order to build a complete practical pipeline from multi-camera acquisition to training and rendering.

3 Methodology

This chapter describes the methodology followed for the development of the proposed 3D reconstruction pipeline. The pipeline includes environment setup, multi-camera data acquisition, preprocessing of captured images, camera pose estimation using COLMAP, and training of a 4D Gaussian Splatting model.

The overall workflow begins with the acquisition of synchronized data using multiple Intel RealSense cameras. The captured images are then converted, organized, and prepared for COLMAP processing. COLMAP is used to estimate camera poses and generate sparse 3D reconstructions [2]. Finally, the registered images and estimated camera parameters are converted into a format suitable for 4D Gaussian Splatting training and rendering [1].

In addition to the local development setup, part of the computational workflow was executed on a High Performance Computing (HPC) cluster. This was necessary due to the computational requirements of COLMAP reconstruction and GPU-based 4D Gaussian Splatting training.

3.1 Environment Setup

An Ubuntu-based development environment was configured in order to support the execution of the complete reconstruction pipeline. The required software components included CUDA [6], PyTorch [7], COLMAP [2], OpenCV [8], NumPy, and the Intel RealSense SDK through the `pyrealsense2` library [5].

The 4D Gaussian Splatting framework was installed in a dedicated project directory, while a separate conda environment was used to manage the required Python packages and CUDA-related dependencies. This separation ensured that the training and rendering pipeline could be executed in a reproducible software environment.

For local experimentation, the pipeline was tested on a workstation with GPU support. However, larger datasets required additional computational resources. For this reason, part of the workflow was executed on the Prometheus HPC cluster using Slurm batch jobs [9, 10]. The HPC environment was used for computationally demanding stages such as COLMAP reconstruction and 4D Gaussian Splatting training, allowing access to shared storage and GPU nodes.

The use of the HPC cluster was necessary because large multi-camera datasets require significant memory, storage, and GPU resources. During experimentation, it was observed that datasets with a high number of cameras and frames could exceed the limitations of the local workstation, making the cluster environment important for processing larger configurations.

3.2 Data Acquisition

The data acquisition process was carried out at the CYENS Centre of Excellence, where a multi-camera setup based on Intel RealSense devices was available. These cameras were used to capture synchronized RGB, depth, and infrared data from multiple viewpoints around the subject [4, 5].

A custom Python script was developed using the `pyrealsense2` library in order to automate the capture process. The implementation was based on the Intel RealSense Python API and adapted for the needs of this project [3]. The script was responsible for detecting the connected cameras, configuring the streams, synchronizing the acquisition process, storing the captured data, and recording metadata for each session.

3.3 Multi-Camera Data Capture System

For the data collection process, a Python-based acquisition system was developed in order to capture synchronized visual and depth information using Intel RealSense cameras [5]. The primary objective of this system was the generation of a structured and high-quality dataset that could be used in subsequent stages for 3D reconstruction and scene representation.

The proposed system supports the simultaneous acquisition of RGB images, depth maps, and infrared (IR) frames, enabling the collection of multimodal data that captures both the appearance and geometric structure of the subject. In addition, the system records detailed metadata for each session, ensuring reproducibility and facilitating downstream processing tasks.

3.3.1 Capture Settings

The initial part of the script defines the fundamental acquisition parameters, including image resolution, frame rate, total number of frames, and the enabled sensing modalities. This configuration is illustrated in Figure 3.1.

```python
RESOLUTION_WIDTH = 1280
RESOLUTION_HEIGHT = 720
FRAME_RATE = 30


DISPOSE_FRAMES_FOR_STABILISATION = 30
MAX_FRAMES = 1800


SAVE_COLOR = True
SAVE_DEPTH = True
SAVE_IR = True
SHOW_PREVIEW = True
```

Figure 3.1: Capture settings used for the RealSense recording process.

These parameters directly influence the spatial and temporal characteristics of the captured dataset. In this implementation, the resolution was set to 1280×720 pixels and the frame rate to 30 frames per second, providing a balance between data quality and computational efficiency. The maximum number of frames was limited to 1800 in order to maintain a consistent dataset size across sessions.

Furthermore, the system enables the simultaneous recording of RGB, depth, and infrared streams, resulting in a multimodal dataset. This combination is useful for 3D reconstruction and future extensions of the pipeline, since it provides both visual texture and geometric information. The optional live preview functionality also improves usability by allowing real-time monitoring of the capture process.

3.3.2 Directory Creation and Metadata Storage

In order to ensure efficient data organization and traceability, dedicated helper functions were implemented for directory creation and metadata storage, as shown in Figure 3.2.

```python
def ensure_dir(path: Path) -> None:
    path.mkdir(parents=True, exist_ok=True)


def save_json(path: Path, data: dict) -> None:
    with open(path, "w", encoding="utf-8") as f:
        json.dump(data, f, indent=2)
```

Figure 3.2: Helper functions for directory creation and JSON metadata storage.

The function `ensure_dir()` guarantees that the required directory structure exists prior to data storage, while `save_json()` serializes structured information into JSON format. This design

enables the systematic storage of session-level and frame-level metadata, including capture parameters, device identifiers, timestamps, and file paths.

Such organization is essential for large-scale data handling and ensures that the dataset remains consistent, interpretable, and easily accessible for further processing.

3.3.3 Camera Intrinsics

The system also extracts and stores the intrinsic calibration parameters of each camera, as illustrated in Figure 3.3.

```python
def get_intrinsics_dict(video_stream_profile: rs.video_stream_profile) -> dict:
    intr = video_stream_profile.get_intrinsics()
    return {
        "width": intr.width,
        "height": intr.height,
        "fx": intr.fx,
        "fy": intr.fy,
        "ppx": intr.ppx,
        "ppy": intr.ppy,
        "model": str(intr.model),
        "coeffs": list(intr.coeffs),
    }
```

Figure 3.3: Extraction of camera intrinsic parameters.

Camera intrinsics describe the internal imaging geometry of the sensor, including focal lengths, principal point coordinates, image resolution, and lens distortion coefficients. These parameters are important for accurate geometric processing, such as camera calibration, image alignment, and three-dimensional reconstruction.

By storing these values alongside the captured data, the system ensures that the dataset can be reliably used in later computational stages without requiring repeated calibration.

3.3.4 Saving RGB, Depth, and Infrared Frames

The script includes dedicated functions for storing RGB, depth, and infrared frames, as shown in Figure 3.4. OpenCV was used for image handling and saving operations [8].

```python
def save_depth_png(path: Path, depth_image: np.ndarray) -> None:
    if depth_image.dtype != np.uint16:
        depth_image = depth_image.astype(np.uint16)
    cv2.imwrite(str(path), depth_image)


def save_ir_png(path: Path, ir_image: np.ndarray) -> None:
    if ir_image.dtype != np.uint8:
        ir_image = ir_image.astype(np.uint8)
    cv2.imwrite(str(path), ir_image)


def save_color_png(path: Path, color_image: np.ndarray) -> None:
    cv2.imwrite(str(path), color_image)
```

Figure 3.4: Functions used to save RGB, depth, and infrared frames.

RGB images are stored as standard PNG files, preserving visual information required by COLMAP and the 4D Gaussian Splatting framework. Depth frames are stored as 16-bit images to maintain distance precision, while infrared images are stored as grayscale data. Although the main reconstruction pipeline in this thesis primarily uses RGB images, the availability of depth and infrared data provides flexibility for future extensions.

3.3.5 Multi-Camera Configuration

The RealSense devices are initialized through a device manager, which automatically detects and enables all connected cameras. This process is illustrated in Figure 3.5.

```
rs_config = rs.config()

if SAVE_DEPTH:
    rs_config.enable_stream(
        rs.stream.depth,
        RESOLUTION_WIDTH,
        RESOLUTION_HEIGHT,
        rs.format.z16,
        FRAME_RATE
    )

if SAVE_IR:
    rs_config.enable_stream(
        rs.stream.infrared,
        1,
        RESOLUTION_WIDTH,
        RESOLUTION_HEIGHT,
        rs.format.y8,
        FRAME_RATE
    )

if SAVE_COLOR:
    rs_config.enable_stream(
        rs.stream.color,
        RESOLUTION_WIDTH,
        RESOLUTION_HEIGHT,
        rs.format.bgr8,
        FRAME_RATE
    )

# Dhmiourgia DeviceManager kai energopoiisi olwn twn syndedemenwn syskevn
device_manager = DeviceManager(rs.context(), rs_config)
device_manager.enable_all_devices()
```

Figure 3.5: Initialization and enabling of all connected RealSense cameras.

When multiple cameras are available, the system supports hardware-level synchronization by assigning one device as the master and the remaining devices as slaves. This configuration reduces temporal discrepancies between captured frames and improves alignment across different viewpoints, which is important for multi-view reconstruction.

3.3.6 Warm-Up Phase

Prior to the main capture procedure, an initialization phase is performed in which a predefined number of frames are discarded, as shown in Figure 3.6.

```
for _ in range(DISPOSE_FRAMES_FOR_STABILISATION):
    device_manager.poll_frames()
```

Figure 3.6: Warm-up phase where initial frames are discarded.

This step allows the cameras to stabilize internal parameters such as exposure and white balance. As a result, the recorded dataset exhibits improved photometric consistency, reducing variability caused by transient sensor adjustments.

3.3.7 Main Capture Loop

The core data acquisition process is implemented through an iterative loop, as illustrated in Figure 3.7.

```
session_meta = {
    "session_name": session_name,
    "width": RESOLUTION_WIDTH,
    "height": RESOLUTION_HEIGHT,
    "fps": FRAME_RATE,
    "max_frames": MAX_FRAMES,
    "save_color": SAVE_COLOR,
    "save_depth": SAVE_DEPTH,
    "save_ir": SAVE_IR,
    "lock_exposure": LOCK_EXPOSURE,
    "master_serial_requested": MASTER_SERIAL,
    "devices": [],
}

device_dirs = {}

enabled_devices = getattr(device_manager, "_enabled_devices", {})


for device_info in available_devices:
    serial = safe_get_serial(device_info)


    device_root = session_dir / serial
    color_dir = device_root / "color"
    depth_dir = device_root / "depth"
    ir_dir = device_root / "ir"

    ensure_dir(device_root)
    if SAVE_COLOR:
        ensure_dir(color_dir)
    if SAVE_DEPTH:
        ensure_dir(depth_dir)
    if SAVE_IR:
        ensure_dir(ir_dir)
```

Figure 3.7: Main loop used to capture frames from all connected cameras.

At each iteration, the system retrieves the latest frames from all active devices. For every captured frame, a structured metadata entry is generated, containing the frame index, system timestamp, and per-device information. This approach ensures temporal tracking and supports synchronization between different data streams.

3.3.8 Frame Metadata

For each captured frame, detailed metadata is recorded, including file paths, timestamps, and frame indices, as shown in Figure 3.8.

```
while frame_count < MAX_FRAMES:
    # Pairnoume ta pio prosfata frames apo oles tis kameres
    frames_devices = device_manager.poll_frames()


    frame_entry = {
        "frame_index": frame_count,
        "system_time_s": time.time(),
        "devices": {},
    }

    # Eikones gia to preview mosaic
    preview_images = []

    # Loop se kathe kamera
    for device_info, frame in frames_devices.items():
        serial = safe_get_serial(device_info)
        folders = device_dirs[serial]

        # Pairnoume ta streams pou yparxoun sto trexon frame set
        color_frame = frame.get(rs.stream.color, None) if isinstance(frame, dict) else None
        depth_frame = frame.get(rs.stream.depth, None) if isinstance(frame, dict) else None

        ir_frame = None
        if isinstance(frame, dict):
            if (rs.stream.infrared, 1) in frame:
                ir_frame = frame[(rs.stream.infrared, 1)]
            elif rs.stream.infrared in frame:
                ir_frame = frame[rs.stream.infrared]
```

Figure 3.8: Frame-level metadata stored for each captured frame.

This metadata enables correspondence between RGB, depth, and infrared data. It also supports dataset indexing, debugging, and reproducibility during later processing stages.

3.3.9 Live Preview

To enhance usability and monitoring, a real-time preview mechanism is implemented, combining multiple camera streams into a unified mosaic display, as shown in Figure 3.9.

```python
# ========================
# Save color / depth / IR
# ========================
if SAVE_COLOR and color_frame is not None:
    color_image = np.asanyarray(color_frame.get_data())
    color_path = folders["color"] / f"{frame_count:05d}.png"
    save_color_png(color_path, color_image)

if SAVE_DEPTH and depth_frame is not None:
    depth_image = np.asanyarray(depth_frame.get_data())
    depth_path = folders["depth"] / f"{frame_count:05d}.png"
    save_depth_png(depth_path, depth_image)

if SAVE_IR and ir_frame is not None:
    ir_image = np.asanyarray(ir_frame.get_data())
    ir_path = folders["ir"] / f"{frame_count:05d}.png"
    save_ir_png(ir_path, ir_image)

# ========================
# Save intrinsics once
# ========================
# Ta intrinsics tis kathe kameras den allazoun ana frame,
# ara ta grafoume mia fora mono.
intrinsics_path = folders["root"] / "intrinsics.json"
if not intrinsics_path.exists():
    intrinsics_data = {
        "serial": serial,
    }

    if color_frame is not None:
        intrinsics_data["color"] = get_intrinsics_dict(
            color_frame.profile.as_video_stream_profile()
        )
    if depth_frame is not None:
        intrinsics_data["depth"] = get_intrinsics_dict(
            depth_frame.profile.as_video_stream_profile()
        )
    if ir_frame is not None:
        intrinsics_data["infrared"] = get_intrinsics_dict(
            ir_frame.profile.as_video_stream_profile()
        )

    try:
        enabled_device = enabled_devices.get(device_info)
        if enabled_device is not None:
            intrinsics_data["depth_scale"] = enabled_device.first_depth_sensor().get_depth_scale()
        else:
            intrinsics_data["depth_scale"] = None
    except Exception:
        intrinsics_data["depth_scale"] = None

    save_json(intrinsics_path, intrinsics_data)
```

Figure 3.9: Creation of a mosaic preview from multiple camera streams.

This functionality allows the operator to verify subject positioning, camera alignment, and data quality during acquisition, reducing the likelihood of recording errors.

3.3.10 Safe Termination

At the end of the acquisition process, all camera streams and visualization windows are properly closed, as shown in Figure 3.10.

```
    frame_entry["devices"][serial] = {
    "color_path": str(color_path.relative_to(session_dir)) if color_path else None,
    "depth_path": str(depth_path.relative_to(session_dir)) if depth_path else None,
    "ir_path": str(ir_path.relative_to(session_dir)) if ir_path else None,
    "color_timestamp_ms": color_frame.get_timestamp() if color_frame else None,
    "depth_timestamp_ms": depth_frame.get_timestamp() if depth_frame else None,
    "ir_timestamp_ms": ir_frame.get_timestamp() if ir_frame else None,
    "frame_number_color": color_frame.get_frame_number() if color_frame else None,
    "frame_number_depth": depth_frame.get_frame_number() if depth_frame else None,
    "frame_number_ir": ir_frame.get_frame_number() if ir_frame else None,
}

frame_log.append(frame_entry)

if frame_count % 100 == 0:
    save_json(session_dir / "frames.json", {"frames": frame_log})

save_json(session_dir / "frames.json", {"frames": frame_log})

finally:
    if device_manager is not None:
        device_manager.disable_streams()

    cv2.destroyAllWindows()
```

Figure 3.10: Safe termination of camera streams and OpenCV windows.

The use of a `finally` block guarantees that system resources are released even in the presence of runtime errors or user interruption. This ensures robustness and prevents resource leakage.

3.3.11 Summary

In summary, the developed system provides a robust and flexible framework for multimodal data acquisition using Intel RealSense cameras. By combining synchronized RGB, depth, and infrared streams with structured metadata and calibration information, the system produces a comprehensive dataset suitable for advanced computer vision tasks.

The resulting dataset serves as the foundation for the subsequent stages of the project, including image preprocessing, camera pose estimation with COLMAP, sparse 3D reconstruction, and 4D Gaussian Splatting training.

3.4 Preprocessing for COLMAP

After data acquisition, the captured images were converted and organized into a structure suitable for COLMAP processing [11]. Since the original recording process produced separate folders for each camera and modality, the RGB images had to be extracted and renamed using a consistent naming convention.

Each image filename encoded both the camera identifier and the frame index, following a format such as:

```
cam1_00000.png
cam2_00000.png
...
cam8_00054.png
```

This naming convention preserved the relationship between each image, its corresponding camera, and its temporal frame index. It also ensured that all images could be processed consistently by COLMAP.

For some experiments, balanced subsets of the captured data were created in order to reduce computational cost and allow controlled comparisons between different camera and frame configurations. For example, one main dataset contained eight cameras with fifty-five frames per camera, resulting in 440 input images. Such balanced subsets were important for evaluating how the number of cameras and frames affects reconstruction quality.

To avoid unnecessary duplication of large image datasets, symbolic links were used in the HPC environment when possible. This allowed the same image files to be referenced from different experiment folders without increasing storage usage.

After COLMAP reconstruction, only the images successfully registered by COLMAP were used for 4D Gaussian Splatting training. This ensured that the training dataset contained only images with valid camera poses. The registered images and camera parameters were then converted into transformation files compatible with the 4D Gaussian Splatting framework.

The transformation files contained, for each registered image, the image path, the corresponding camera-to-world transformation matrix, and a normalized time value. The time value was computed from the frame index in order to represent the temporal dimension required by the 4D Gaussian Splatting model.

3.5 3D Reconstruction with COLMAP

COLMAP was used to estimate camera poses and generate sparse 3D reconstructions from the captured multi-view images [2, 12]. The COLMAP pipeline consisted of three main stages: feature extraction, feature matching, and sparse reconstruction.

During feature extraction, COLMAP detected local visual features in each input image. Since the datasets contained images from multiple cameras, the `ImageReader.single_camera` option was disabled, allowing COLMAP to estimate camera parameters appropriately for the multi-camera setup.

The second stage was feature matching. Exhaustive matching was used in the experiments, meaning that COLMAP attempted to match features between all pairs of images. Although this approach is computationally more expensive, it increases the probability of finding valid correspondences across different viewpoints, which is important for multi-view reconstruction.

The third stage was sparse reconstruction using the COLMAP mapper. In this stage, COLMAP estimated camera poses and triangulated 3D points from the matched image features. The output consisted of registered images, estimated camera parameters, and a sparse 3D point cloud.

After reconstruction, the COLMAP model was analyzed using the `model_analyzer` tool. This provided useful quantitative information, including the number of registered images, the number of reconstructed 3D points, and the mean reprojection error. These values were later used to compare the reconstruction quality of different experimental configurations.

For the dataset consisting of eight cameras and fifty-five frames per camera, the input contained 440 images. COLMAP successfully registered 432 images, producing a sparse reconstruction with 21,256 3D points and a mean reprojection error of 1.179051 pixels. This indicates that almost all images were successfully included in the reconstruction.

The distribution of registered images per camera was also examined. Seven cameras were fully registered with 55 images each, while one camera registered 47 images. This result suggests that the selected dataset provided sufficient overlap and feature correspondences for stable multi-view reconstruction.

The COLMAP reconstruction was then converted from binary format to text format using the COLMAP model converter. The resulting files, including camera parameters, image poses, and 3D points, were used in the next preprocessing step to generate the transformation files required by the 4D Gaussian Splatting framework.

The exact COLMAP commands used during the experiments are provided in the Appendix.

3.6 4D Gaussian Splatting Training

After the COLMAP reconstruction and preprocessing stages, the registered dataset was used to train the 4D Gaussian Splatting model [1]. The objective of this stage was to learn a scene representation from the registered multi-view images and their corresponding camera poses.

The training dataset was organized in a format compatible with the 4D Gaussian Splatting framework. It contained the registered RGB images and the transformation files `transforms_train.json`, `transforms_test.json`, and `transforms_val.json`. These files provided the model with image paths, camera-to-world transformation matrices, and normalized time values for each frame.

A configuration file was created for the selected dataset by adapting an existing D-NeRF configuration [13]. This configuration defined the dataset path and the parameters required for training and rendering.

The training stage required GPU acceleration through CUDA and PyTorch [6, 7]. Initial tests showed that the controller node of the cluster could not be used for training because it did not provide access to an NVIDIA GPU driver. Therefore, the training process was executed on a GPU node using a Slurm batch job [9]. The Slurm script requested GPU resources, CPU cores, memory, and an appropriate time limit for the training process.

During training, the model optimized a set of Gaussian primitives in order to represent the captured scene. These Gaussians encode spatial position, scale, rotation, opacity, and appearance information. The trained representation was then used for novel view rendering.

After training, the rendering stage generated visual outputs from the trained model. The produced rendered videos were used for qualitative evaluation of the reconstructed scene. This final stage completed the pipeline by converting the learned 4D Gaussian representation into visual results suitable for comparison across experimental configurations.

The detailed training and rendering commands, together with the Slurm batch scripts, are provided in the Appendix.

4 Experiments

4.1 Experimental Setup

This chapter presents the experimental setup used to evaluate the performance of the 4D Gaussian Splatting pipeline under different data configurations. The main objective of the experiments was to investigate how the number of cameras, the number of frames per camera, and the distribution of frames across cameras affect the quality of the reconstructed scene.

Multiple experimental scenarios were examined by varying the number of camera viewpoints and the number of temporal samples. Some experiments were performed using datasets captured during the main data acquisition process, while additional experiments were carried out using a separate dataset of images that was provided for further evaluation.

All datasets were processed using the same general pipeline. First, the images were preprocessed and organized into a structure compatible with COLMAP. The COLMAP pipeline was then executed in order to perform feature extraction, feature matching, camera pose estimation, and sparse reconstruction. The resulting camera poses and sparse point clouds were subsequently used for 4D Gaussian Splatting training [1].

For each experimental setup, the trained model was used to generate rendered videos of the reconstructed scene. These videos were used for qualitative evaluation of the visual consistency and appearance of the learned representation. The experiments therefore evaluate both the geometric reconstruction produced by COLMAP and the visual output produced by the 4D Gaussian Splatting model.

4.2 Evaluation Approach

The reconstruction quality was evaluated qualitatively using two main outputs: the sparse point cloud generated by COLMAP and the rendered video generated after 4D Gaussian Splatting training.

The COLMAP output is stored as a `.ply` file, which represents a sparse point cloud of the reconstructed scene. This file contains 3D points estimated from visual features that were detected, matched across multiple views, and triangulated in 3D space. Therefore, the `.ply` output does not represent a complete surface or mesh, but rather a sparse geometric approximation of the scene.

The density and coherence of the point cloud provide an indication of how successfully COLMAP

estimated camera poses and reconstructed the main visible structure. A denser and more coherent point cloud usually indicates stronger feature correspondences and better multi-view overlap. In contrast, scattered or isolated points may appear due to weak feature matches, background elements, limited texture, motion blur, poor lighting, or insufficient overlap between camera views.

In addition to the sparse point cloud, rendered videos were generated from the trained 4D Gaussian Splatting models. These videos were used as qualitative visual outputs, allowing the reconstructed scene to be observed from a sequence of viewpoints. While the point cloud provides information about the geometric reconstruction, the rendered video provides a clearer indication of the visual appearance and consistency of the learned Gaussian representation.

The following subsections present the results of the different experimental configurations. A comparative discussion of the experiments is provided at the end of the chapter.

4.3 Results

4.3.1 Reconstruction Using 7 Cameras and 41 Frames per Camera

In this experiment, a dataset consisting of images captured from seven different cameras was used, with each camera contributing 41 frames. The purpose of this experiment was to evaluate whether a multi-view setup with a moderate number of frames per camera could provide sufficient information for stable sparse reconstruction.

The images were processed using COLMAP to estimate camera poses and generate a sparse 3D reconstruction in the form of a point cloud stored in `.ply` format. The resulting point cloud is illustrated in Figure 4.1.

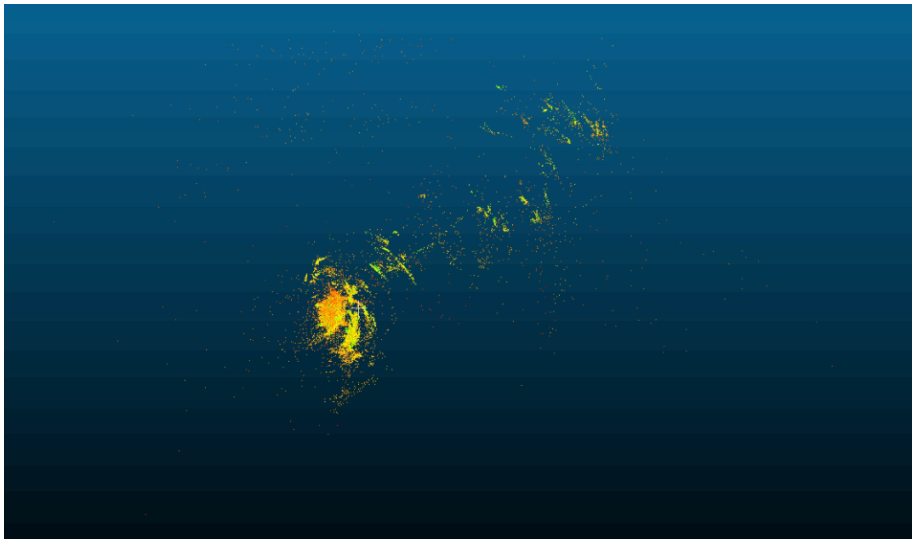

Figure 4.1: Point cloud reconstruction using 7 cameras and 41 frames per camera.

A rendered video was also generated from the trained 4D Gaussian Splatting model. A representative frame from the video is shown in Figure 4.2. The video was used as an additional qualitative output for evaluating the visual appearance and consistency of the reconstructed scene.

Figure 4.2: Rendered video frame from the 4D Gaussian Splatting model trained using 7 cameras and 41 frames per camera. The full video is provided as supplementary material.

The visualization shows a relatively dense and coherent cluster of points corresponding to the main subject. The central region contains a higher point density, indicating that the subject was observed from multiple viewpoints and that COLMAP was able to identify sufficient feature correspondences.

Some scattered points are also visible around the main reconstructed area. These points are expected in sparse reconstructions and may correspond to weaker matches, background features, or regions with less reliable visual information. Overall, the result shows that this configuration was able to produce a recognizable sparse reconstruction of the scene.

4.3.2 Reconstruction Using 13 Cameras and 21 Frames per Camera

In this experiment, a dataset consisting of 13 cameras was used, with each camera contributing 21 frames. The purpose of this experiment was to evaluate the effect of using a larger number of camera viewpoints while keeping the number of frames per camera relatively low.

The reconstruction was performed using COLMAP, generating a sparse point cloud in `.ply` format. The resulting point cloud is shown in Figure 4.3.

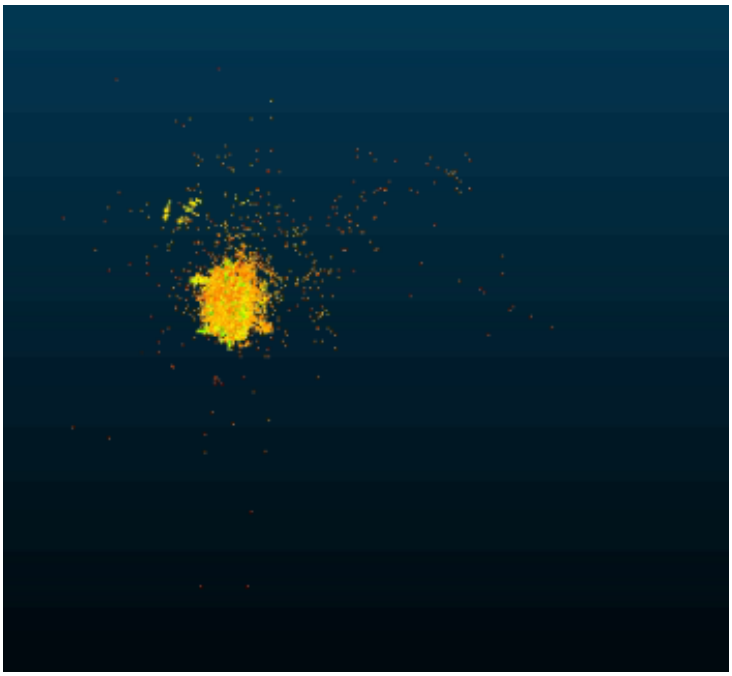

Figure 4.3: Point cloud reconstruction using 13 cameras and 21 frames per camera.

A rendered video was also generated from the trained 4D Gaussian Splatting model. A representative frame from the video is shown in Figure 4.4. The video was used to examine whether the learned representation could produce a visually coherent output from the sparse reconstruction.

Figure 4.4: Rendered video frame from the 4D Gaussian Splatting model trained using 13 cameras and 21 frames per camera. The full video is provided as supplementary material.

The resulting reconstruction appears sparse and partially fragmented. Although the dataset included a larger number of camera viewpoints, the limited number of frames per camera reduced the amount of visual information available from each viewpoint.

The point cloud contains a visible central region, but it lacks strong global coherence and density. This suggests that the available images did not provide enough consistent feature correspondences across all views. As a result, COLMAP was able to reconstruct only parts of the scene reliably, while other regions remained sparse or disconnected.

4.3.3 Reconstruction Using an Imbalanced Frame Distribution

In this experiment, three cameras were used with an imbalanced number of frames. One camera contributed 120 frames, while the other two cameras contributed 56 frames each. The purpose of this experiment was to observe the effect of uneven frame distribution across cameras on the reconstruction process.

The resulting point cloud is shown in Figure 4.5.

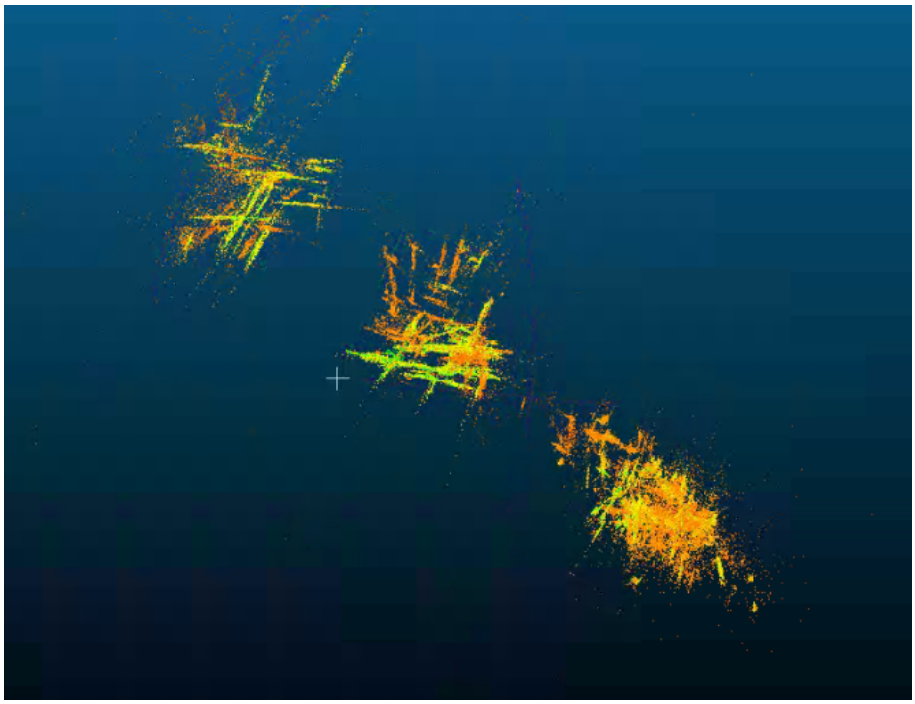

Figure 4.5: Point cloud reconstruction with imbalanced frame distribution across cameras.

A rendered video was also generated from the trained 4D Gaussian Splatting model. A representative frame from the video is shown in Figure 4.6. The video was used to assess how the imbalanced frame distribution affected the visual stability of the learned representation.

Figure 4.6: Rendered video frame from the 4D Gaussian Splatting model trained using an imbalanced frame distribution across cameras. The full video is provided as supplementary material.

The reconstruction exhibits clear structural inconsistencies. The point cloud is split into disconnected clusters, indicating that COLMAP was not able to establish a globally coherent reconstruction across all camera views.

The imbalanced distribution of frames likely caused some viewpoints to dominate the reconstruction, while others provided less consistent support. This resulted in uneven point density and weaker connections between reconstructed regions. The rendered output produced from this configuration was also very short, suggesting that the 4D Gaussian Splatting model did not learn a stable temporal representation from the available data.

4.3.4 Reconstruction Using 8 Cameras and 35 Frames per Camera

In this experiment, a dataset consisting of eight cameras was used, with each camera contributing 35 frames. The purpose of this experiment was to evaluate whether a balanced multi-camera configuration with a moderate number of frames per camera could provide sufficient information for camera pose estimation, sparse reconstruction, and 4D Gaussian Splatting training.

The captured images were first preprocessed and organized into a COLMAP-compatible structure. COLMAP was then used to estimate the camera poses and generate a sparse point cloud representation of the scene. The resulting point cloud is shown in Figure 4.7.

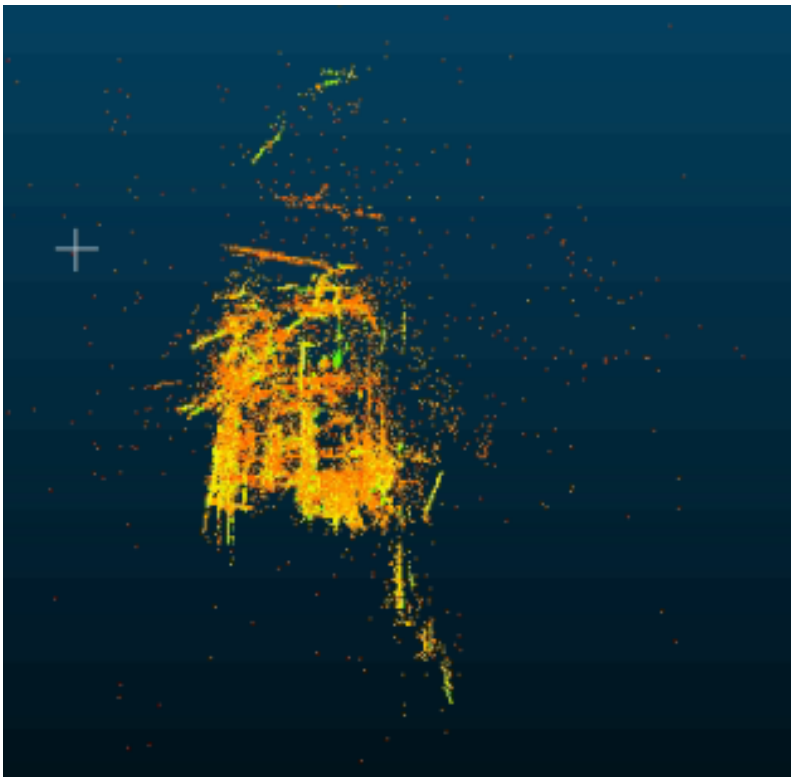

Figure 4.7: Point cloud reconstruction using 8 cameras and 35 frames per camera.

As shown in Figure 4.7, the reconstructed point cloud forms a dense central region corresponding to the main subject. This indicates that the subject was visible from several viewpoints and that COLMAP was able to detect and match a sufficient number of common features.

At the same time, scattered points can be observed around the main structure. The reconstruction appears vertically elongated and not fully complete, suggesting that some regions were not captured with enough consistency or did not contain enough distinctive features for reliable triangulation. Nevertheless, the configuration produced a recognizable sparse reconstruction.

In addition to the sparse point cloud, a rendered video was generated from the trained 4D Gaussian Splatting model. A representative frame from the rendered video is shown in Figure 4.8.

Figure 4.8: Rendered video frame from the 4D Gaussian Splatting model trained using 8 cameras and 35 frames per camera. The full video is provided as supplementary material.

The rendered video shows that the model was able to learn the main structure and appearance of the scene. However, some visual limitations may still be present, especially in areas where the COLMAP reconstruction was sparse or incomplete. These limitations may appear as slight distortions, unstable details, or incomplete regions in the rendered output.

4.3.5 Reconstruction Using 9 Cameras and 39 Frames per Camera

In this experiment, a different dataset was tested, consisting of images that were provided separately for additional evaluation. This dataset contained nine cameras, with each camera contributing 39 frames. The purpose of this experiment was to evaluate whether this multi-camera dataset could provide sufficient visual information for camera pose estimation, sparse reconstruction, and 4D Gaussian Splatting training.

The captured images from the provided dataset were preprocessed and organized into a COLMAP-compatible structure. COLMAP was then used to estimate the camera poses and generate a sparse 3D point cloud of the scene. The resulting reconstruction is shown in Figure 4.9.

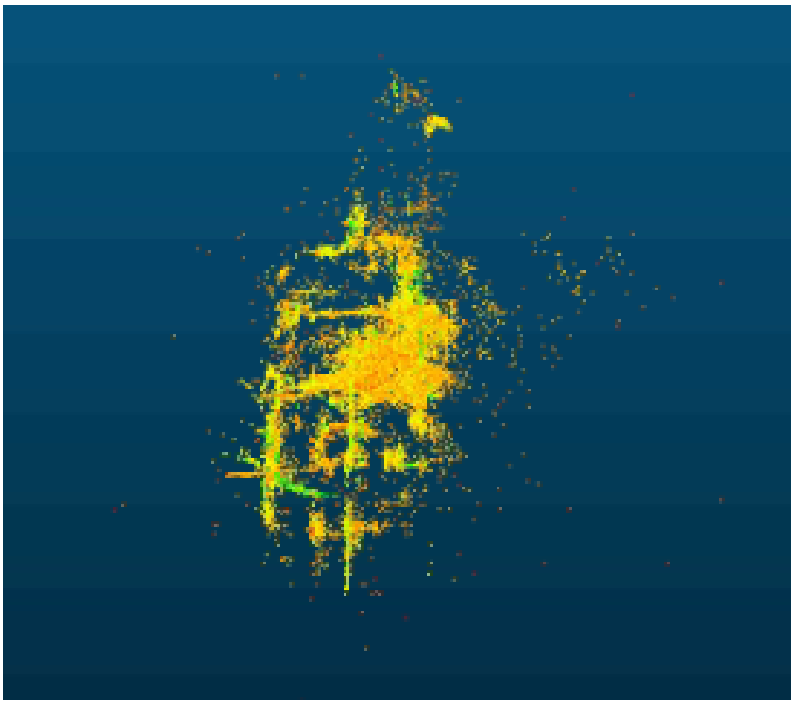

Figure 4.9: Point cloud reconstruction using 9 cameras and 39 frames per camera.

As shown in Figure 4.9, the reconstructed point cloud forms a visible central structure corresponding to the main subject. The central area contains a higher concentration of points, indicating that this region was observed consistently from multiple viewpoints and provided sufficient feature correspondences for reconstruction.

Scattered points are also visible around the main reconstructed region. The reconstruction appears partially incomplete in some areas, which may be due to limited overlap between certain viewpoints, insufficient texture, or regions of the scene that were not observed consistently enough. Despite these limitations, the point cloud shows that COLMAP was able to reconstruct the dominant structure of the scene.

A rendered video was also generated from the trained 4D Gaussian Splatting model. A representative frame from the video is shown in Figure 4.10.

Figure 4.10: Rendered video frame from the 4D Gaussian Splatting model trained using 9 cameras and 39 frames per camera. The full video is provided as supplementary material.

The rendered output provides a qualitative view of the learned Gaussian representation. The model was able to capture the general appearance of the scene, although minor artifacts or unstable details may appear in areas where the COLMAP point cloud was sparse or noisy.

4.3.6 Reconstruction Using 8 Cameras and 55 Frames per Camera on the Prometheus HPC Cluster

In this experiment, a dataset consisting of eight cameras was used, with each camera contributing 55 frames. The purpose of this experiment was to evaluate a more balanced configuration with a larger number of frames per camera, while also testing the execution of the reconstruction and training pipeline on an HPC environment.

For this experiment, the processing was carried out on the Prometheus HPC cluster at CYENS. Prometheus is a high-performance computing cluster designed for deep learning and scientific computing workloads. It provides access to NVIDIA GPUs, AMD EPYC processors, high-performance Lustre storage, and Slurm job scheduling [10]. This environment was particularly useful for this project because both COLMAP reconstruction and 4D Gaussian Splatting training require significant computational resources, especially when processing multi-camera image datasets.

The captured images were preprocessed and organized into a COLMAP-compatible structure. COLMAP was then executed on the cluster in order to estimate camera poses and generate a sparse 3D point cloud of the scene. The resulting reconstruction is shown in Figure 4.11.

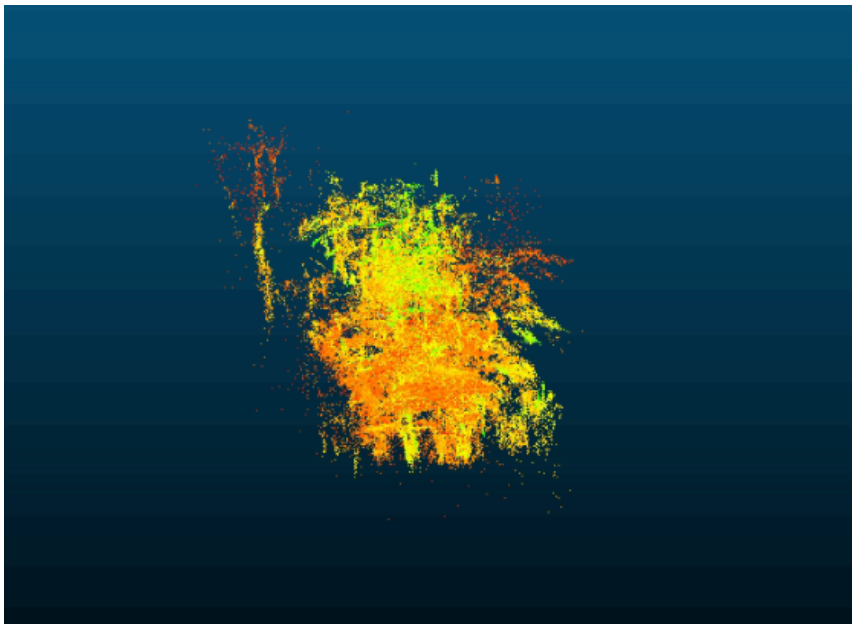

Figure 4.11: Point cloud reconstruction using 8 cameras and 55 frames per camera.

As shown in Figure 4.11, the point cloud forms a dense central structure corresponding to the main subject. The reconstruction appears more concentrated around the main region of interest, indicating that the available camera views and frame sequence provided sufficient feature correspondences for COLMAP to estimate camera poses and triangulate the dominant scene structure.

Some scattered points are still visible around the main reconstruction. These points are expected in sparse point cloud outputs and may be caused by weak feature matches, background details, limited texture, or inconsistencies between camera viewpoints. However, the main reconstructed region remains clearly visible and recognizable.

A rendered video was also generated from the trained 4D Gaussian Splatting model. A representative frame from the rendered output is shown in Figure 4.12.

Figure 4.12: Rendered video frame from the 4D Gaussian Splatting model trained using 8 cameras and 55 frames per camera. The full video is provided as supplementary material.

The rendered output provides a qualitative view of the learned Gaussian representation. The video indicates that the model was able to capture the main structure and appearance of the scene. Since this experiment used a larger number of frames per camera, the model had access to more temporal observations from each viewpoint, which supported a more stable reconstruction and rendering process.

Overall, this experiment shows that the 8-camera and 55-frame configuration was suitable for the complete pipeline, from COLMAP pose estimation to 4D Gaussian Splatting training and rendering. The use of the Prometheus HPC cluster also enabled the processing of this larger dataset using shared storage and GPU resources.

4.3.7 Reconstruction Using 10 Cameras and 55 Frames per Camera on the Prometheus HPC Cluster

In this experiment, a dataset consisting of ten cameras was used, with each camera contributing 55 frames. The experiment was executed on the Prometheus HPC cluster, following the same processing pipeline used in the previous HPC-based experiment. The purpose of this experiment was to evaluate whether increasing the number of camera viewpoints while keeping the number of frames per camera constant could provide more complete spatial coverage for reconstruction.

The captured images were preprocessed and organized into a COLMAP-compatible structure. COLMAP was then used to estimate the camera poses and generate a sparse 3D point cloud of the scene. The resulting reconstruction is shown in Figure 4.13.

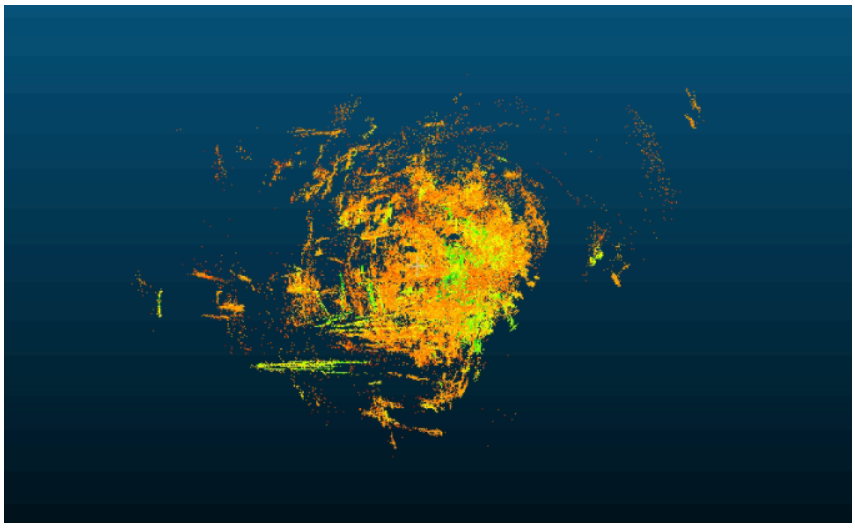

Figure 4.13: Point cloud reconstruction using 10 cameras and 55 frames per camera.

As shown in Figure 4.13, the reconstructed point cloud contains a dense central region corresponding to the main subject. The larger number of camera viewpoints provides more spatial observations of the scene, allowing COLMAP to reconstruct a wider area around the subject.

The point cloud also contains several scattered regions around the main structure. These points may correspond to background elements, weaker feature matches, or parts of the scene that were only partially observed from some viewpoints. Although the reconstruction is not a complete surface model, the dominant structure of the scene remains visible and recognizable.

The use of 55 frames per camera provided a consistent number of observations from each viewpoint. This supported the camera pose estimation process and provided the 4D Gaussian Splatting model with a balanced input sequence for training.

A rendered video was also generated from the trained 4D Gaussian Splatting model. A representative frame from the rendered output is shown in Figure 4.14.

Figure 4.14: Rendered video frame from the 4D Gaussian Splatting model trained using 10 cameras and 55 frames per camera. The full video is provided as supplementary material.

The rendered output provides a qualitative view of the learned Gaussian representation. The model was able to capture the general structure and appearance of the scene, while the increased number of camera viewpoints provided additional visual information during training. However, some artifacts or incomplete areas may still appear in regions where the COLMAP reconstruction was sparse, noisy, or affected by weak feature correspondences.

Overall, the 10-camera and 55-frame configuration produced a recognizable reconstruction and demonstrated that the pipeline can process a larger multi-camera dataset on the Prometheus HPC cluster. The result shows that increasing the number of cameras can improve scene coverage, provided that each camera contributes a sufficient and balanced number of frames.

4.3.8 Reconstruction Using 11 Cameras and 39 Frames per Camera on the Prometheus HPC Cluster

In this experiment, a dataset consisting of 11 cameras was used, with each camera contributing 39 frames. The experiment was executed on the Prometheus HPC cluster, using the same reconstruction and training pipeline described in the previous HPC-based experiments. The purpose of this experiment was to evaluate whether a larger number of camera viewpoints could provide sufficient spatial coverage for sparse reconstruction and 4D Gaussian Splatting training, while maintaining a moderate number of frames per camera.

The images were preprocessed and organized into a COLMAP-compatible structure. COLMAP was then used to estimate camera poses and generate a sparse 3D point cloud of the scene. The resulting reconstruction is shown in Figure 4.15.

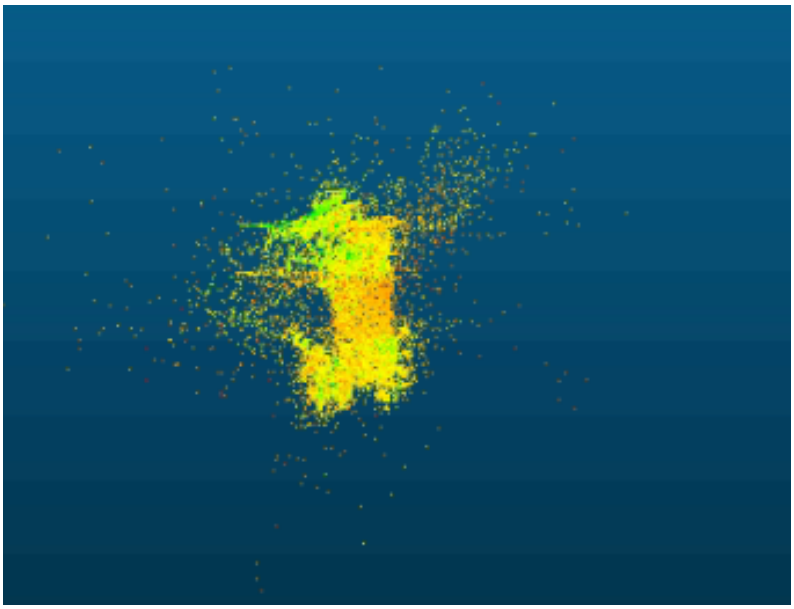

Figure 4.15: Point cloud reconstruction using 11 cameras and 39 frames per camera.

As shown in Figure 4.15, the reconstructed point cloud forms a visible central structure corresponding to the main subject. The highest concentration of points appears in the central region, indicating that this part of the scene was observed consistently across multiple viewpoints and provided sufficient feature correspondences for triangulation.

At the same time, the point cloud remains relatively sparse and includes scattered points around the main reconstructed structure. These isolated points are likely caused by weaker feature matches, background elements, limited texture, or insufficient overlap between some camera views. As a result, the reconstruction captures the dominant structure of the scene, but it does not form a complete or uniformly dense representation.

A rendered video was also generated from the trained 4D Gaussian Splatting model. A representative frame from the rendered output is shown in Figure 4.16.

Figure 4.16: Rendered video frame from the 4D Gaussian Splatting model trained using 11 cameras and 39 frames per camera. The full video is provided as supplementary material.

The rendered output provides a qualitative view of the learned Gaussian representation. The model was able to reconstruct the general appearance of the scene, although some incomplete regions or unstable details may still appear in areas where the COLMAP reconstruction was sparse. This indicates that, while the dataset was sufficient to support the full pipeline on the Prometheus HPC cluster, the reconstruction quality remained dependent on the strength of the underlying feature correspondences and the consistency of the available observations.

Overall, this experiment shows that the 11-camera and 39-frame configuration was able to produce a recognizable reconstruction and a usable rendered output. The Prometheus HPC cluster supported the execution of the full pipeline, including COLMAP processing, 4D Gaussian Splatting training, and rendering.

4.4 Quantitative Summary of Experiments

Table 4.1 summarizes the main quantitative and qualitative results obtained from the different experimental configurations. The table includes the number of cameras, the number of frames per camera, the total number of input images, the number of images successfully registered by COLMAP, the number of reconstructed sparse points, the mean reprojection error, the duration of the rendered video, and the execution environment.

Experiment	Images	Registered	Points	Error (px)	Video	Quality	Environment
7 cams × 41 frames	287	164	20064	1.274161	4 sec	Medium	PC
13 cams × 21 frames	273	210	11712	0.981148	6 sec	Low	PC
3 cams imbalanced	232	8	52	0.646313	1 sec	Low	PC
8 cams × 35 frames	280	140	6302	1.085519	4 sec	Good	PC
9 cams × 39 frames	351	351	16432	1.154718	10 sec	Good	PC
8 cams × 55 frames	440	432	21256	1.179051	14 sec	Low	HPC
10 cams × 55 frames	550	434	13411	1.034072	14 sec	Low	HPC
11 cams × 39 frames	429	388	25656	1.159901	14 sec	High	HPC

Table 4.1: Summary of the experimental configurations and reconstruction results.

4.5 Discussion

The experimental results show that reconstruction quality is affected by several factors, including the number of cameras, the number of frames per camera, the balance of frame distribution, and the quality of the camera pose estimation produced by COLMAP. The results also show that a higher number of input images does not always guarantee a better reconstruction. Instead, the quality of the final result depends on whether the images provide sufficient overlap, reliable feature correspondences, and balanced observations across the different camera viewpoints.

The experiment using seven cameras and 41 frames per camera produced a recognizable sparse reconstruction and a rendered output of medium visual quality. Although not all images were registered by COLMAP, the reconstructed point cloud remained relatively coherent around the main subject. This indicates that the configuration provided enough visual information for COLMAP to recover the dominant scene structure, but incomplete registration limited the final reconstruction and rendering quality.

The experiment using 13 cameras and 21 frames per camera showed that increasing the number of cameras alone is not sufficient to improve the result. Although more viewpoints were available, the reduced number of frames per camera limited the amount of consistent information provided by each viewpoint. As a result, the point cloud appeared sparse and partially fragmented, and the rendered video had lower visual quality. This suggests that viewpoint diversity must be supported by sufficient temporal sampling.

The imbalanced experiment produced the weakest result. In this setup, one camera contributed significantly more frames than the other two cameras. This uneven distribution caused the reconstruction to become unstable, with only a very small number of images successfully registered by COLMAP. Although the mean reprojection error was low, this value was not representative of a successful reconstruction because it was computed from very few registered images and

reconstructed points. This confirms that reprojection error must be interpreted together with the number of registered images, the number of reconstructed points, and the visual quality of the rendered output.

The experiments using eight cameras and 35 frames per camera and nine cameras and 39 frames per camera produced more stable results. In particular, the nine-camera configuration achieved complete image registration and generated a rendered video of good visual quality. This shows that a balanced dataset with sufficient overlap and a moderate number of frames per camera can provide a strong basis for 4D Gaussian Splatting training.

The experiments executed on the Prometheus HPC cluster allowed larger datasets to be processed. The eight-camera and 55-frame configuration achieved a high number of registered images and sparse points, but the rendered output was still visually limited. Similarly, the ten-camera and 55-frame configuration included the largest number of input images, but did not produce the best qualitative result. These results show that COLMAP reconstruction quality and 4D Gaussian Splatting rendering quality are related but not identical. A strong sparse reconstruction does not always guarantee a high-quality rendered video, since the final output is also affected by training stability, scene appearance, camera consistency, and input image quality.

Among the tested configurations, the 11-camera and 39-frame experiment produced the best overall qualitative result. It generated the highest number of sparse points and resulted in the most visually coherent rendered output. This suggests that this configuration achieved the best balance between viewpoint diversity and temporal sampling among the tested experiments.

Overall, the experiments demonstrate that the most effective configurations are not necessarily those with the largest number of cameras or the largest number of images. Instead, reliable reconstruction requires a balanced combination of sufficient camera coverage, adequate frames per camera, consistent observations, and strong feature correspondences. Balanced data acquisition across cameras is therefore essential for stable COLMAP reconstruction and high-quality 4D Gaussian Splatting rendering.

5 Conclusion and Future Work

This chapter concludes the thesis by summarizing the main objectives, the implemented pipeline, and the experimental findings. It also discusses the limitations of the work and presents possible directions for future improvement.

5.1 Conclusion

The purpose of this thesis was to develop and evaluate a complete multi-camera reconstruction pipeline using Intel RealSense cameras, COLMAP, and 4D Gaussian Splatting. The work focused on the full process from data acquisition to preprocessing, camera pose estimation, sparse reconstruction, model training, and rendering. Through this pipeline, synchronized multi-view image data were transformed into a format suitable for 4D Gaussian Splatting, allowing the generation of reconstructed scenes and rendered videos.

A Python-based multi-camera acquisition system was used to capture RGB, depth, and infrared data from multiple Intel RealSense cameras. The system also stored metadata and camera intrinsic parameters, ensuring that the captured data could be organized and reused in later processing stages. After acquisition, the images were converted, renamed, and structured into a COLMAP-compatible format. COLMAP was then used to estimate camera poses and generate sparse point cloud reconstructions, which were later converted into transformation files required by the 4D Gaussian Splatting framework.

An important part of the work was the execution of computationally demanding stages on the Prometheus HPC cluster. The use of Slurm batch jobs, shared storage, and GPU resources enabled the processing of larger datasets and supported the training and rendering stages of the 4D Gaussian Splatting pipeline. This was particularly important for experiments involving larger numbers of cameras and frames.

Several experiments were conducted in order to evaluate how different dataset configurations affect reconstruction quality. The results showed that reconstruction quality depends not only on the number of cameras, but also on the number of frames per camera, the balance of frame distribution, and the consistency of visual overlap between viewpoints. Experiments with imbalanced frame distributions produced weak and fragmented reconstructions, while more balanced configurations generally produced more coherent point clouds and more useful rendered outputs.

The quantitative results further demonstrated that a high number of input images does not automatically guarantee better reconstruction quality. For example, some configurations registered many images but still produced low-quality rendered videos, while other configurations with

fewer frames produced better qualitative results. This shows that COLMAP registration statistics, sparse point density, reprojection error, and rendered video quality must be considered together when evaluating the pipeline.

Among the tested configurations, the experiment using 11 cameras and 39 frames per camera produced the strongest overall qualitative result, generating the highest number of sparse points and a high-quality rendered video. The 9-camera and 39-frame experiment also performed well, achieving complete image registration and good visual quality. In contrast, the imbalanced experiment produced the weakest result, confirming that balanced data acquisition across cameras is essential for stable reconstruction and rendering.

Overall, this thesis demonstrates that combining COLMAP with 4D Gaussian Splatting can provide a practical pipeline for multi-camera dynamic scene reconstruction. The results confirm that an appropriate balance between viewpoint diversity and temporal sampling is necessary in order to obtain stable camera pose estimation, coherent sparse reconstructions, and visually meaningful rendered outputs.

5.1.1 Limitations

Although the proposed pipeline produced usable reconstruction and rendering results, several limitations were observed during the development and experimentation process.

First, the 4D Gaussian Splatting training stage was limited by the available GPU memory and system memory. Larger datasets, especially those containing a high number of cameras and frames, required significant computational resources and in some cases caused the training process to terminate. This shows that dataset size must be carefully balanced with the available hardware resources.

Second, the quality of the reconstruction was strongly dependent on the COLMAP camera pose estimation stage. COLMAP was sensitive to sparse visual features, weak texture, motion blur, and insufficient overlap between camera views. When the captured images did not provide enough reliable feature correspondences, the resulting sparse point cloud became fragmented or incomplete.

Another limitation was related to synchronization between cameras. Although the capture process was designed to collect synchronized multi-view data, small timing differences between cameras may still affect the consistency of dynamic scenes. Such differences can reduce the quality of feature matching and make the reconstruction less stable.

Training instability was also observed in some experimental setups, especially when the dataset was small, imbalanced, or contained insufficient viewpoint diversity. In such cases, the rendered video could be short, unstable, or visually inconsistent.

Finally, rendering artifacts were present in some outputs. These artifacts appeared as noisy regions, incomplete structures, or unstable details in the rendered videos. They were mainly caused by sparse or noisy COLMAP reconstructions, limited viewpoint overlap, and inconsistencies in the input data.

Overall, these limitations show that the quality of the final reconstruction depends not only on the 4D Gaussian Splatting model, but also on the quality of the captured data, the camera synchronization, the viewpoint coverage, and the reliability of the COLMAP reconstruction.

5.2 Future Work

Although the implemented pipeline produced recognizable reconstructions and rendered videos, several limitations remain and could be addressed in future work.

First, future experiments could investigate more camera and frame configurations in a more controlled way. In this thesis, several different combinations were tested, but a larger and more systematic evaluation could provide clearer conclusions about the optimal balance between the number of cameras and the number of frames per camera. Additional experiments could also include repeated captures under the same setup in order to evaluate the consistency and repeatability of the results.

Second, the camera placement and overlap between viewpoints could be improved. Since COLMAP depends heavily on visual feature correspondences, insufficient overlap or poor camera positioning can lead to sparse, noisy, or fragmented reconstructions. A more carefully designed camera arrangement could improve feature matching and produce more complete point clouds.

Third, the quality of the captured images could be improved through better lighting conditions, reduced motion blur, and more stable subject positioning. Poor lighting, textureless regions, and inconsistent image quality can reduce the number of reliable feature matches and affect both COLMAP reconstruction and 4D Gaussian Splatting training.

Future work could also make greater use of the depth and infrared data captured by the Intel RealSense cameras. In this thesis, the RGB images were mainly used for COLMAP and 4D Gaussian Splatting. However, depth information could potentially be integrated to improve geometric accuracy, support scale estimation, or assist in filtering noisy points from the reconstruction.

In addition, future experiments could include more quantitative evaluation metrics for the rendered videos. The current evaluation was mainly based on COLMAP statistics, sparse point cloud inspection, and qualitative video analysis. Metrics such as PSNR, SSIM, LPIPS, or comparison against reference views could provide a more objective assessment of rendering quality.

Finally, the pipeline could be extended to support more complex dynamic scenes and more

realistic avatar-generation scenarios. This would require improved synchronization, more robust preprocessing, and possibly more advanced 4D Gaussian Splatting configurations. With these improvements, the proposed workflow could become a stronger foundation for high-quality multi-view dynamic reconstruction and personalized 3D/4D scene representation.

BIBLIOGRAPHY

[1] G. Wu, "4d gaussian splatting project page," https://guanjunwu.github.io/4dgs/, 2024, accessed: 2026-05-14.

[2] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[3] Intel RealSense, "Box dimensioner multicam demo," https://github.com/IntelRealSense/librealsense/blob/master/wrappers/python/examples/box_dimensioner_multicam/box_dimensioner_multicam_demo.py, 2023, accessed: 2026-05-14.

[4] Intel Corporation, "Intel realsense viewer user guide," https://www.generationrobots.com/media/intel-realsense/Intel_RealSense_Viewer_Userguide.pdf, 2022, accessed: 2026-05-14.

[5] ——, "Intel realsense sdk 2.0," https://github.com/IntelRealSense/librealsense, 2026, accessed: 2026-05-14.

[6] NVIDIA Corporation, "Cuda toolkit documentation," https://developer.nvidia.com/cuda-toolkit, 2026, accessed: 2026-05-14.

[7] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[8] OpenCV Team, "Opencv documentation," https://opencv.org/, 2026, accessed: 2026-05-14.

[9] SchedMD, "Slurm workload manager documentation," https://slurm.schedmd.com/documentation.html, 2026, accessed: 2026-05-14.

[10] CYENS Centre of Excellence, "Prometheus HPC Cluster Documentation," https://prometheus-docs.cyens.org.cy/docs/, 2025, accessed: 2026-05-14.

[11] Schönberger, Johannes L., "Colmap documentation," https://colmap.github.io/, 2026, accessed: 2026-05-14.

[12] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision*, 2016.

[13] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, “D-nerf: Neural radiance fields for dynamic scenes,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 318–10 327.

APPENDICES

APPENDIX I
Additional Implementation Details

This appendix provides additional implementation details related to the reconstruction pipeline developed in this thesis. The purpose of this appendix is to describe supporting technical aspects of the implementation without interrupting the main flow of the methodology chapter.

I.1 Dataset Organization

After the data acquisition stage, the captured images were organized into a structured format suitable for multi-view reconstruction. The images were grouped according to the camera from which they were captured and were renamed using a consistent naming convention. This organization made it easier to process the dataset automatically and reduced the possibility of mismatches between cameras and frames.

For the reconstruction experiments, subsets of the captured data were selected in order to evaluate different combinations of camera viewpoints and temporal samples. The selected images were prepared in a format compatible with the reconstruction tools used in the project.

I.2 COLMAP Processing Workflow

The COLMAP processing stage consisted of feature extraction, feature matching, and sparse reconstruction. During feature extraction, visual features were detected in each image. During feature matching, correspondences between images from different viewpoints were identified. Finally, sparse reconstruction was performed in order to estimate camera poses and reconstruct a sparse point cloud of the scene.

The generated COLMAP reconstruction was then inspected in order to verify the number of successfully registered images, the stability of the reconstruction, and the overall quality of the sparse point cloud. The resulting camera parameters were used in the next stage of the pipeline.

I.3 Transformation File Generation

The 4D Gaussian Splatting framework requires transformation files that describe the relationship between each image and its corresponding camera pose. For this reason, the camera parameters estimated by COLMAP were converted into a format compatible with the training framework.

Each image entry in the transformation files includes the image path, the camera transformation matrix, and a normalized temporal value. The temporal value represents the relative position of each frame within the captured sequence and allows the dataset to be used in a 4D reconstruction setting.

I.4 Training and Rendering Workflow

After preprocessing and transformation file generation, the registered dataset was used to train the 4D Gaussian Splatting model. The training process used the prepared images and camera parameters in order to learn a scene representation based on Gaussian primitives.

Once training was completed, a rendering stage was used to generate visual outputs from the trained model. These outputs were used for qualitative evaluation of the reconstruction and for comparing the effect of different dataset configurations.

I.5 Use of HPC Resources

Due to the computational requirements of COLMAP reconstruction and 4D Gaussian Splatting training, part of the workflow was executed on an HPC cluster. Batch jobs were used to manage long-running tasks and to access GPU resources when required.

Using the HPC environment allowed the pipeline to process larger datasets more efficiently and reduced the limitations imposed by local hardware. Shared storage was also used to manage large image datasets and intermediate reconstruction outputs.