Advancing Photometric Odometry to Dense Volumetric Simultaneous Localization and Mapping

by

Yan Song Hu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
System's Design Engineering

Waterloo, Ontario, Canada, 2025

© Yan Song Hu 2025

Examining Committee Membership

The following are the reviewers for this thesis. They are all faculty of the University of Waterloo.

Supervisor(s):	John Zelek Associate Professor, Systems Design Engineering
Internal Reviewer:	Nasser Lashgarian Azad Professor, Systems Design Engineering
Internal Reviewer:	Stephen L. Smith Professor, Electrical and Computer Engineering

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

I am the primary author for all of the work presented in this thesis. Contributions to Chapter 4 were made by Nicolas Abboud, Muhammad Qasim Ali, and Adam Srebrnjak Yang. Specifically, Muhammad Qasim Ali developed the majority of the SLAM backend presented in Section 4.4.2, Adam Srebrnjak Yang played a key role in evaluating MGSO, and Nicolas Abboud contributed to writing the discussion. All other work in this thesis was carried out solely by me. The work presented in chapter 4 is based on a preprint of a paper [27] that is approved to be published at IEEE International Conference on Robotics and Automation (ICRA) 2025.

Abstract

Navigating complex environments remains a fundamental challenge in robotics. At the core of this challenge is Simultaneous Localization and Mapping (SLAM), the process of creating a map of the environment while simultaneously using that map for navigation. SLAM is essential for mobile robotics because effective navigation is a prerequisite for nearly all real-world robotic applications. Visual SLAM, which relies solely on the input of RGB cameras is important because of the accessibility of cameras, which makes it an ideal solution for widespread robotic deployment.

Recent advances in graphics have driven innovation in the visual SLAM domain. Techniques like Neural Radiance Fields (NeRF) and 3D Gaussian Splatting (3DGS) enable the rapid generation of dense volumetric scenes from RGB images. Researchers have integrated these radiance field techniques into SLAM to address a key limitation of traditional systems. Although traditional SLAM excels at localization, the generated maps are often unsuitable for broader robotics applications. By incorporating radiance fields, SLAM systems have the potential for the real-time creation of volumetric metric-semantic maps, offering substantial benefits for robotics. However, current radiance field-based SLAM approaches face challenges, particularly in processing speed and map reconstruction quality.

This work introduces a solution that addresses limitations in current radiance fields SLAM systems. Direct SLAM, a traditional SLAM technique, shares key operational similarities with radiance field approaches that suggest potential synergies between the two systems. Both methods rely on photometric loss optimization, where the pixel differences between images guide the optimization process. This work demonstrates that the benefits of combining these complementary techniques extend beyond theory.

This work demonstrates the synergy between radiance field techniques and direct SLAM through a novel system that combines 3DGS with direct SLAM, achieving a superior combination of quality, memory efficiency, and speed compared to existing approaches. The system, named MGSO, addresses a challenge in current 3DGS SLAM systems: Initializing 3D Gaussians while performing SLAM simultaneously. The proposed approach leverages direct SLAM to produce dense and structured point clouds for 3DGS initialization. This results in faster optimization, memory compactness, and higher-quality maps even with mobile hardware. These results demonstrate that traditional direct SLAM techniques can be effectively integrated with radiance field representations, opening avenues for future research.

Acknowledgements

I extend my sincere gratitude to all members of the Vision and Imaging Processing Lab at the University of Waterloo. Your friendship and collaborative spirit made my time both enriching and memorable.

Dedication

This thesis is dedicated to the late Cheng Yan, who always wanted her nephew to become a great academic.

Table of Contents

Examining Committee Membership ii

Author's Declaration iii

Statement of Contributions iv

Abstract v

Acknowledgements vi

Dedication vii

List of Figures xii

List of Tables xiv

List of Abbreviations xv

1 Introduction 1

1.1 Overview of Thesis 2

1.2 Background 3

1.2.1 A Short Introduction to SLAM 3

1.2.2 Unique Challenges of VSLAM 5

1.2.3 What makes Direct SLAM Unique 5

2 Development of Direct SLAM **7**

2.1 Introduction . . . 7

2.2 History of VSLAM . . . 7

2.2.1 A Brief Clarification on SfM versus SLAM . . . 9

2.3 Development of Direct SLAM . . . 9

2.3.1 Early Work . . . 9

2.3.2 SVO and LSD-SLAM: Foundations of Modern Direct SLAM . . . 10

2.3.3 Development of DSO . . . 12

2.3.4 Short Summary of DSO . . . 12

2.4 DM-HSLAM: My contribution to the DSO family . . . 15

2.5 Future of Direct SLAM . . . 18

3 First Attempts in combining Direct SLAM with Neural Scene Representations **19**

3.1 Intro . . . 19

3.2 Dense SLAM History . . . 20

3.2.1 Hand-Crafted Methods . . . 20

3.2.2 A Brief comment on the Sensors used in Dense SLAM . . . 22

3.2.3 NeRF Introduction . . . 22

3.3 Development of NeRF SLAM . . . 24

3.3.1 Benefits of the NeRF Representation . . . 24

3.3.2 History of NeRF SLAM . . . 24

3.4 Reasons for Combining Direct SLAM and NeRF . . . 25

3.5 Method . . . 25

3.6 Results and Discussion . . . 26

3.7 Next Steps . . . 27

4 Monocular-GSO: Harnessing the Synergy of 3DGS and Direct SLAM 28
4.1 Introduction . . . 28
4.2 3D Gaussian Splatting for SLAM . . . 29
4.2.1 3D Gaussian Splatting . . . 29
4.2.2 Typical 3DGS Pipeline . . . 30
4.2.3 Improvements over NeRF for SLAM . . . 31
4.2.4 Prior 3DGS SLAM Systems . . . 31
4.3 Main Ideas behind MGSO . . . 33
4.3.1 Initializing 3DGS in New Areas . . . 35
4.3.2 Assisting Primitive Cloning and Pruning . . . 35
4.4 Method . . . 36
4.4.1 SLAM module . . . 36
4.4.2 Dense Reconstruction module . . . 38
4.5 Results . . . 39
4.6 Discussion . . . 41
4.6.1 Localization . . . 41
4.6.2 Ablations . . . 43
4.6.3 Insights . . . 48
4.7 Extending MGSO . . . 48

5 Conclusion 50
5.1 Summary and Contributions . . . 50
5.2 Future Research . . . 52

References 54

APPENDICES 63

A Extra Information **64**

A.1 A Brief Clarification on Visual Odometry versus SLAM 64

A.2 Important Modern VSLAM Systems 65

A.3 Relevant Extensions of DSO 66

B 3DGS Explanation **69**

B.1 3DGS Rasterization and Optimization Process 69

C Basics of Photometric SLAM **72**

C.1 Photometric Loss Example 72

C.2 Overview 72

C.2.1 Pixel Selection 73

C.2.2 Tracking 74

C.2.3 Mapping 75

C.2.4 Initialization 79

C.3 Mathematics 80

C.4 Residual Calculation 80

C.5 Jacobian Derivative Calculations 82

Glossary **89**

List of Figures

2.1 SLAM Timeline 9
2.2 SVO Operation 10
2.3 DSO Operation 11
2.4 DM-VIO Operation 13
2.5 DSO Pixel Pattern 14
2.6 DM-HSLAM Operation 16

3.1 DSO Dense Pixel Map 21
3.2 NeRF Render 23
3.3 NeRF Diagram 23
3.4 NeRF Map Drift 27

4.1 MGSO Renders 28
4.2 3DGS Diagram 30
4.3 Photometric SLAM and 3DGS Point Cloud Comparison 33
4.4 Photometric and Feature-based SLAM Point Cloud Comparison 34
4.5 ORBSLAM and MGSO Tracking Comparison 36
4.6 DSO and MGSO Point Cloud Comparison 37
4.7 MGSO Renders 1 44
4.8 MGSO Renders 2 45
4.9 Radar Diagram comparing dense SLAM systems 47

A.1 Timeline of extensions for DSO 67

B.1 3DGS Diagram 70

C.1 DSO Selection Grid Diagram 73

List of Tables

4.1 Existing 3DGS SLAM Systems 32
4.2 Datasets Used 39
4.3 Hardware Used 39
4.4 Absolute Trajectory Error 41
4.5 Reconstruction Results on Replica 41
4.6 Reconstruction Results on EuRoC 42
4.7 Reconstruction Results on TUM's 42
4.8 Replica Aggregated Results 46
4.9 Cloning and Pruning Interval Ablation 46
4.10 Reduced Input Points Ablation 46

A.1 Table of major modern SLAM systems 65

List of Abbreviations

3DGS 3D Gaussian Splatting 2, 7, 18, 22, 28, 29, 31, 33, 38, 48, 50, 52, 69

CAD Computer-Aided Design 20

DM-VIO Delayed Marginalization Visual-Inertial Odometry 15

DSO Direct Sparse Odometry 7, 12, 15, 34, 36, 50, 66

DTAM Dense Tracking and Mapping 9

HSLAM Hybrid Direct-Indirect Visual SLAM 15

ICP Iterative Closest Point 33

IMU Inertial Measurement Unit 5, 16

LiDAR Light Detection and Ranging 1

LPIPS Learned Perceptual Image Patch Similarity 40

LSD-SLAM Large-Scale Direct Monocular SLAM 10, 11, 15, 65

NeRF Neural Radiance Fields 2, 18, 22, 24, 30, 33, 69

PnP Perspective-n-Point 8

PSNR Peak Signal-to-Noise Ratio 35, 40

PTAM Parallel Tracking and Mapping 8, 9

SfM Structure from Motion 8, 9, 21, 30, 71

SLAM Simultaneous Localization and Mapping 1, 3, 19, 20, 52, 71

SSIM Structural Similarity Index Measure 40, 70

SVO Fast Semi-Direct Monocular Visual Odometry 10, 11, 65

VSLAM Visual Simultaneous Localization and Mapping 1–3, 7, 9, 28, 50

Chapter 1

Introduction

Simultaneous Localization and Mapping (SLAM) addresses a fundamental mobile robotics problem: how to navigate and create maps of new environments during online operation. This capability is essential for robots to become truly independent. As robotics applications expand beyond controlled industrial settings into homes, offices, and public spaces, the ability to map and navigate unknown environments in real-time has become important for widespread robotic deployment.

SLAM poses a fundamental paradox: It requires localization to construct a map while simultaneously requiring a map for accurate localization. Despite this challenge, researchers have developed numerous successful SLAM solutions that have transitioned from theory to practice. SLAM systems, particularly those that leverage Light Detection and Ranging (LiDAR) sensors, are now widely deployed in applications such as self-driving cars and industrial robots. Although established SLAM algorithms have demonstrated success, SLAM is still a research area as challenges remain with dynamic environments, difficult conditions, long-term operation detection, and computational efficiency in real-time applications. Visual Simultaneous Localization and Mapping (VSLAM) is a key direction for making SLAM more practical and accessible.

Cameras offer compelling advantages for SLAM systems: they are lightweight, energy efficient, and cost-effective while providing rich high-volume data. The human visual system demonstrates that sophisticated navigation and mapping is possible using only visual input, suggesting that VSLAM has great potential. Although VSLAM has advanced considerably, fundamental challenges persist, particularly in handling dynamic environments and maintaining consistent maps over extended periods. This thesis addresses another important challenge: the real-time generation of high-fidelity, semantically meaningful maps

that can support downstream robotic applications.

Recent breakthroughs in radiance field techniques, such as Neural Radiance Fields (NeRF) and 3D Gaussian Splatting (3DGS), enable the development of better VSLAM systems. Unlike traditional VSLAM approaches, these radiance field SLAM systems can generate volumetric reconstructions of environments in real-time. The continuous nature of these representations offers significant advantages for downstream robotic tasks such as navigation, manipulation, and scene understanding. This thesis advances the emerging field by introducing a novel technique to improve radiance field SLAM systems.

The core key insight of this thesis is that traditional direct SLAM principles naturally complement neural radiance field methods. This is demonstrated through a novel system, named Monocular-GSO or MGSO, that fuses 3DGS with direct SLAM techniques, achieving superior performance compared to other contemporary dense SLAM systems. Monocular-GSO gets its name from the two key components it builds upon: Direct Sparse Odometry (DSO), a SLAM system, and 3D Gaussian Splatting (GS), a dense reconstruction technique. The name GSO is derived by combining elements from both. The primary contribution of MGSO is a novel, synergistic integration of photometric SLAM with 3DGS that only requires a monocular RGB camera. Photometric SLAM provides dense, structured point clouds for 3DGS initialization, accelerating optimization and enabling more efficient maps with fewer Gaussians. As a result, MGSO achieves a superior balance of reconstruction quality, memory efficiency, and speed, outperforming the state-of-the-art in experiments. While MGSO represents one successful approach to this fusion, it opens up a broader research direction for integrating direct SLAM principles with radiance field rendering techniques.

1.1 Overview of Thesis

This thesis is structured into three primary sections, each building towards the introduction of a novel VSLAM system (MGSO) that synergistically combines direct SLAM and 3DGS. The first section explains how direct SLAM works. The second section examines the challenges and requirements of dense SLAM. The final section demonstrates how direct SLAM techniques can address the challenges of dense SLAM.

Chapter Two: This section chronicles the history of VSLAM from photogrammetry to direct SLAM approaches. It also explains the operational principles of influential direct SLAM systems, including the framework upon which the presented system is built. This

section provides essential information for understanding the operational principles of the MGSO's localization module.

Chapter Three: This section examines dense SLAM approaches and how the introduction of radiance fields methods for 3D scene reconstruction addressed challenges in the field. The section also summarizes my initial efforts to combine direct SLAM with radiance field representations. This early work was not successful, but it directly informed the design principles and technical innovations of MGSO.

Chapter Four: This section introduces the new dense SLAM system MGSO, which achieves state-of-the-art performance by synergistically integrating direct SLAM techniques with 3DGS.

Appendix: The appendix provides supplementary information that clarifies the ideas presented in the main thesis but is not essential to its core arguments. It is organized into three distinct sections. The first section presents broader discussions on general SLAM methodology that is not specifically focused on this thesis. This content serves as valuable context for newcomers or those less familiar with SLAM concepts. The second section provides an explanation on how 3DGS works. The third section documents insights into the DSO SLAM system that emerged during my research. These findings may prove particularly valuable for researchers working with DSO-related systems.

1.2 Background

The remainder of this introduction summarizes foundational concepts in general SLAM, VSLAM, and direct SLAM, serving both as a primer for those new to the field and as a refresher for experienced practitioners.

1.2.1 A Short Introduction to SLAM

SLAM is a computational task in which a system determines its location while simultaneously building a map of the surroundings. The core of all modern SLAM systems is determining the most probabilistically likely state of both the map and the observer based

on all sensor measurements and control inputs up to the current time t. This can be expressed mathematically as:

$$P(x_t|u_{1:t}, z_{1:t}) \tag{1.1}$$

Where x is the state of the map and observer, u are the measurements, and z are the control inputs.

This equation can be solved through recursive Bayesian Estimation using methods such as Extended Kalman Filters and Particle Filters. However, this thesis focuses on VSLAM, where the only input comes from standard RGB cameras. Visual data presents two significant challenges for filtering methods: it generates a high volume of information and exhibits highly non-linearity. These characteristics make filter-based approaches computationally intensive and unreliable. As a result, more flexible optimization-based techniques are a more common approach for vision-based SLAM systems.

Optimization-based SLAM approaches represent the localization and mapping problem as a graph optimization task, where each graph node represents either a state (of the map or observer), a sensor measurement, or a control input. The relationships between these nodes form the edges of the graph. These graph edges act as soft constraints on the system. For example, a measurement indicating that a landmark is 5 meters left of a state creates an edge suggesting that any valid solution should maintain approximately that spatial relationship. Likewise, edges between consecutive robot states encode the expectation that the robot's position changes smoothly and reasonably over time.

However, real-world uncertainties mean that it is typically impossible to find a solution that satisfies all of these constraints simultaneously. Instead, graph-based SLAM algorithms seek to find the configuration of nodes that minimizes the total error or deviation of these soft constraints. To maintain computational efficiency, most graph-based SLAM systems operate on a sliding window of recent measurements and states. As measurements and states age beyond this window, they are marginalized out of the graph through mathematical processes that preserve their accumulated information. To ensure long-term mapping accuracy, SLAM systems can maintain a separate comprehensive graph that is optimized less frequently. This global optimization, known as loop closure, typically occurs when the system recognizes a previously visited location. At these moments, the system can connect recent pose estimates (the "head" of the graph) back to earlier mapped positions, creating loops in the graph structure.

Although recent advances in machine learning have produced other methods to solve the fundamental SLAM problem, graph-based optimization remains the backbone of most

modern systems. **All VSLAM systems examined in detail within this thesis utilize graph-based optimization as their back-end framework**.

1.2.2 Unique Challenges of VSLAM

VSLAM processes only RGB camera data for navigation. Although this camera-only approach offers practical advantages in cost and simplicity, it introduces fundamental challenges. The primary challenge lies in the dimensional gap between our 3D world and 2D camera images. Cameras capture information by projecting our 3D world into a 2D plane, which causes depth information to be lost. VSLAM systems must therefore either develop algorithms that work directly with these 2D projections or reconstruct 3D information before performing SLAM computations.

This dimensional mismatch leads to a unique problem in monocular VSLAM named scale ambiguity. When viewing a scene through a single camera, it becomes impossible to determine absolute distances of objects. For example, a small object near the object appears the same as a larger object farther away. This limitation forces monocular systems to work with relative scales, which can lead to scale drift where the same scale is not kept throughout. Although this challenge can be addressed by using stereo cameras or depth cameras, using more advanced sensors limits the main advantage of VSLAM: lower cost, reduced power consumption, and more compact design. Scale error can also be addressed inexpensively by incorporating data from an Inertial Measurement Unit (IMU), which is discussed in section 2.4 in this thesis. Additionally, while cameras provide abundant 2D data, 2D data is often affected by significant noise and uncertainty. Camera images can be affected by varying lighting conditions, motion blur, and environmental factors that introduce noise into the measurements. This requires efficient processing algorithms that can account for the inherent noise in 2D images.

Although the robotics community has traditionally distinguished between visual odometry and VSLAM systems, **this thesis will treat visual odometry and VSLAM as equivalent terms**. 'VSLAM' is the unified terminology for both concepts following the convention in recent dense SLAM research. If there is any confusion about this convention, please refer to Appendix A.1.

1.2.3 What makes Direct SLAM Unique

To provide context for this thesis, this section defines and explains direct SLAM. Direct SLAM is defined as **VSLAM that directly uses image intensity values directly for**

tracking and mapping.

Traditional VSLAM approaches, known as feature-based or indirect methods, first convert raw visual data into sparse trackable feature points to facilitate easier mapping and localization. In contrast, direct methods process the intensities of the pixels directly, estimating the depth values of each pixel to create a dense 3D representation of the environment. Direct SLAM methods use full image intensity data rather than sparse features, with the aim of maximizing the use of available visual information. The foundation principle of direct SLAM is optimizing the loss of this photometric loss equation between frames i and j:

$$E := (I_j[\boldsymbol{p_j}] - I_i[\boldsymbol{p_i}]) \tag{1.2}$$

Here, I denotes the intensity of the pixel at position $\boldsymbol{p} = (u, v)$ in the image. The fundamental principle of this photometric error function is to calculate the camera pose transformation that minimizes the intensity differences between corresponding pixels in consecutive frames i and j. A simple example of how photometric loss works is presented in Appendix C. Photometric loss relies on the intensity values of the pixels, making it highly sensitive to distortions that affect light. As a result, it is crucial to calibrate a camera's response to light, which is known as photometric calibration, to ensure consistent and accurate measurements. The selection of pixels used in the optimization varies across systems: earlier approaches utilized all available pixels, while more recent implementations are semi-dense methods that only track only a subset. Furthermore, The optimization framework can be extended with additional terms to deal with effects such as lighting changes and camera exposure settings, allow for normalization, or apply heuristically designed weights.

Chapter 2

Development of Direct SLAM

2.1 Introduction

Direct Sparse Odometry (DSO) [15], the direct SLAM system that forms the localization backbone of MGSO. MGSO combines DSO's tracking and mapping capabilities with the reconstruction capabilities of 3DGS. This section examines the operational principles, historical development, and key advantages of DSO. This section begins with a comprehensive timeline of VSLAM's evolution leading to DSO's development, followed by a detailed analysis of DSO's operational principles. The section concludes with a brief discussion of a separate research project conducted during my Master's program, which is independent of MGSO.

2.2 History of VSLAM

VSLAM faces challenges due to the inherent noise and variability in camera images. For systems with high-precision sensors, SLAM has been solved using probabilistic state estimation techniques such as the Extended Kalman Filters and Rao-Blackwellized Particle Filters. These classical methods have been comprehensively surveyed by Durrant-Whyte and Bailey [13][3], and their fundamental concepts are thoroughly explained by Thrun et al.'s textbook 'Probabilistic Robotics' [68]. However, cameras capture only 2D projections of the 3D world, introducing uncertainties in 3D positions since depth must be estimated. Visual data is further complicated because variations in lighting, viewpoint, and perspective can dramatically alter image appearance. Due to the uncertainties of visual sensing,

the direct application of probabilistic state estimation techniques is difficult. Initial efforts in successfully implementing VSLAM required techniques from photogrammetry to reliably extract geometric information from 2D images.

Photogrammetry encompasses a family of techniques for extracting measurements from 2D images. Its origins can be traced to the 1800s when Aimé Laussedat pioneered the use of photographs for mapping [21], establishing photography as a tool for geometric measurement. Structure from Motion (SfM) is a photogrammetric technique that reconstructs the geometry of 3D scenes from multiple overlapping photographs. Given an unordered collection of images, the SfM algorithms simultaneously calculate both the camera positions at which the images were captured and some 3D information about the photographed scene.

An important advancement in SfM came from Agarwal et al. [2], who developed a system for robust SfM in large scenes. An important aspect of their work is the usage of feature points - distinctive image locations that can be reliably identified across multiple images. These features, detected using algorithms such as the Scale-Invariant Feature Transform (SIFT) developed by Lowe [40], serve as stable anchors for matching corresponding points, enabling the application of multi-view geometry for 3D reconstruction. Identifying corresponding points across multiple images can recover depth information through geometric parallax. The relative position and orientation of each camera can then be determined using Perspective-n-Point (PnP) algorithms once enough 3D points are found. These reconstruction techniques would be applied to develop the first VSLAM systems.

A significant advancement in VSLAM was the introduction of Mono-SLAM by Davison et al. [12]. The Mono-SLAM system employed a filtering approach combined with feature points. Mono-SLAM extracts distinctive image features across consecutive frames, calculates their depths through image correspondences, and updates the system's state estimate using the Extended Kalman Filters. However, early SLAM systems that used filtering struggled with fast computational complexity growth as map size increased. This issue was addressed by Klein and Murray's work [36] on Parallel Tracking and Mapping (PTAM) for small AR workspace. PTAM is also feature-based, but uses a keyframe-based bundle adjustment instead of filters. PTAM's keyframe-based map optimization pioneered the development of optimization-based SLAM techniques, which represent the mapping problem as a graph-optimization task. Since PTAM demonstrated the advantages of keyframe-based bundle adjustment, VSLAM systems have predominantly shifted towards optimization-based approaches. Optimization-based approaches better handle both the inherent non-linearity of the SLAM problem and the large volumes of visual data involved.

From these early systems, SLAM would continue to be improved upon to handle issues such as scalability, dynamic objects, and environmental variation. Due to their complex-

ity, SLAM systems are typically large frameworks that build upon preexisting systems by refining and improving various components. Many influential systems are detailed in Appendix A.2. The subsequent section will explore how direct SLAM evolved from these early VSLAM systems.

2.2.1 A Brief Clarification on SfM versus SLAM

VSLAM and SfM both reconstruct 3D scenes and camera poses from images. Although they share similar core algorithms, SLAM does online processing of temporally ordered images for navigation, while SfM is focused on offline reconstruction. SLAM's incremental processing of images enables accelerated computational processing. Using motion priors from the previous frame, SLAM can significantly narrow the search space for new frame pose estimation. Consequently, the system can perform localized calculations, avoiding the computational overhead of global optimization. In comparison, SfM works offline on unordered collections of images, focusing on accuracy and robustness through global optimization.

2.3 Development of Direct SLAM

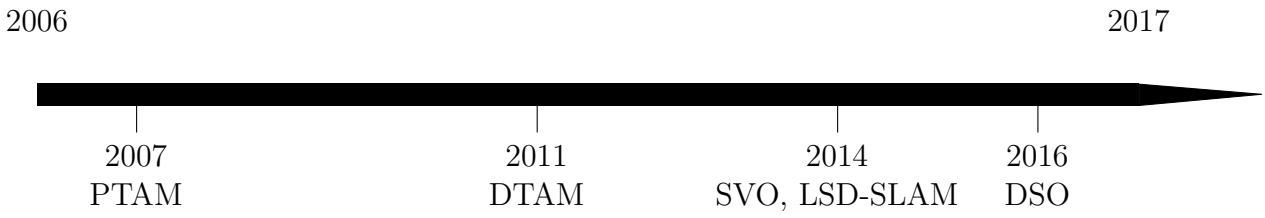


Figure 2.1: Timeline of important systems for direct SLAM

2.3.1 Early Work

Dense Tracking and Mapping (DTAM) by Newcombe, Lovegrove, and Davison [49] is one of the first direct SLAM systems. The goal of DTAM is to improve PTAM by performing tracking on all pixels in the image, enabling more comprehensive use of the inputted images compared to PTAM's sparse approach. Although indirect SLAM systems depend on

distinct environmental features for tracking, making them unreliable in texture-less scenes, direct methods can operate effectively in any environment that contains even subtle variations in image intensity. This dense formulation improved tracking robustness compared to PTAM, particularly during rapid camera motion, while simultaneously enabling more detailed and complete 3D scene reconstructions of the scene.

Although DTAM represented a major step forward, its intensive computational requirements posed practical limitations because computing photometric error over an entire image is very resource-intensive. To improve the efficiency of direct SLAM, researchers explored semi-dense systems that use photometric loss with only a subset of image pixels. The shift towards semi-dense methods was driven by a key insight: certain image regions, particularly those lacking texture or distinctive features, contribute negligible photometric information. Following the groundbreaking release of DTAM, two independent research laboratories simultaneously developed advances in semi-dense SLAM systems: the Robotics and Perception Group at the University of Zurich developed Fast Semi-Direct Monocular Visual Odometry (SVO) [18] and the Computer Vision Group at the Technical University of Munich developed Large-Scale Direct Monocular SLAM (LSD-SLAM) [16].

2.3.2 SVO and LSD-SLAM: Foundations of Modern Direct SLAM

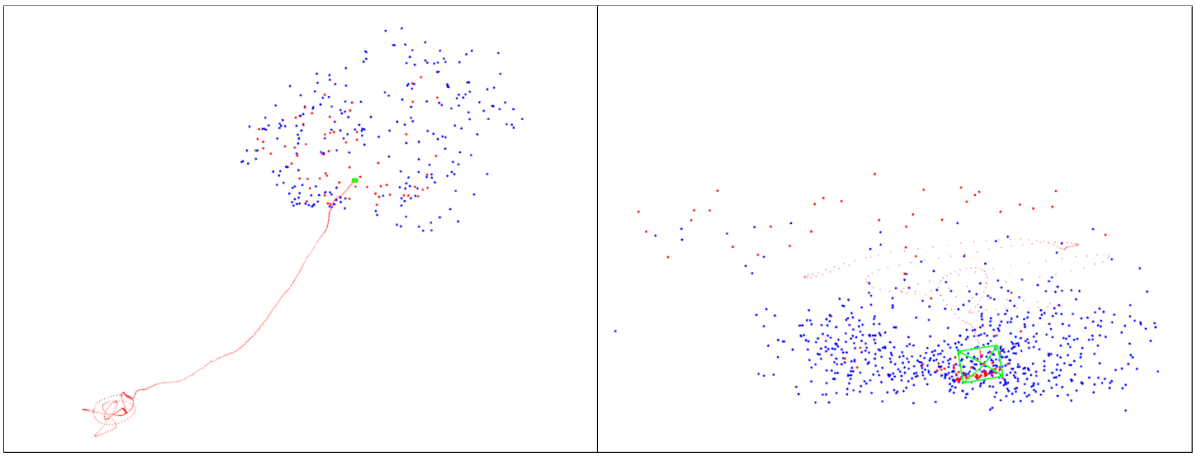


Figure 2.2: Images of SVO operation at different times on the EuRoC Micro Aerial Vehicle (MAV) dataset. The blue and red dots are the feature points used by SVO for mapping and tracking. Because SVO is an traditional visual odometry system and only utilizes the immediate spatial information around it, the dots are clustered around the camera.

SVO [18] is a precise, robust, and extremely computationally efficient semi-direct SLAM method. It is categorized as semi-direct because it both detects feature points and uses photometric loss. The system works by optimizing photometric loss across hundreds of small-pixel patches to determine camera pose changes. These pixel patches are placed at feature point locations when keyframes are initialized. SVO's usage of feature points is different from traditional indirect methods that must perform feature detection and matching every frame. The depth estimation for each pixel patch is estimated using a Bayesian filter, with pixel patches added only to the map once their depth estimates achieve sufficient consistency. By not doing feature detection and matching on every frame and only calculating photometric loss on a subset of pixels, SVO is much faster than previous direct and indirect SLAM systems.

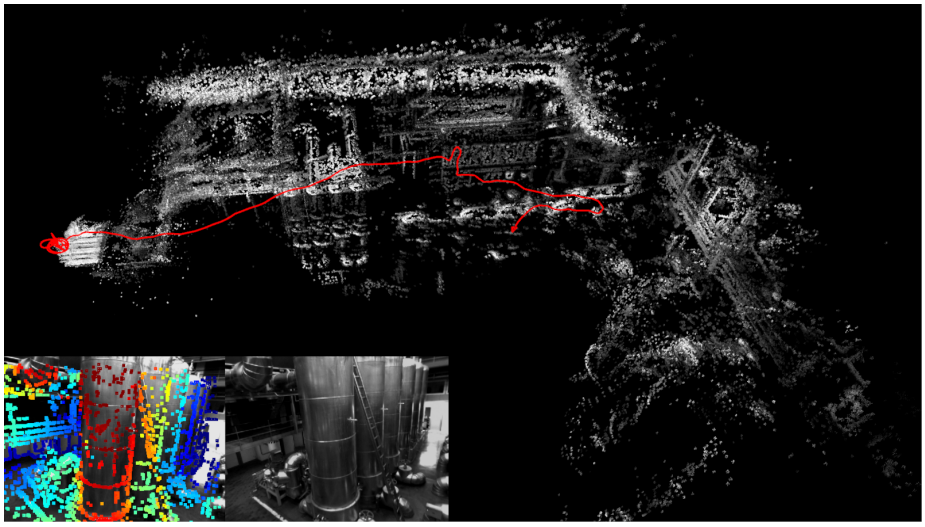


Figure 2.3: An images of DSO operating on the EuRoC Micro Aerial Vehicle (MAV) dataset. DSO is a direct extension of LSD-SLAM that refined the ideas introduced by LSD-SLAM and their output maps look very similar. The full map being shown in the image is a visual illusion created by DSO stitching together many local maps. DSO is a traditional visual odometry system and only uses immediate spatial information.

LSD-SLAM [16] used the novel approach of using high gradient regions instead of all available pixels for direct SLAM. It optimized tracking using a scale-drift aware photometric loss formulation, and the depths of the pixels are calculated through many stereo comparisons followed by spatial regularization and outlier removal. Furthermore, LSD-SLAM incorporated loop closure by passing the data to OPENFABMAP [20] to detect loops, an independent appearance-based mapping system. As one of the first direct SLAM

systems, LSD-SLAM established key foundations that would later be expanded by DSO [15], which is the system that forms the basis of this thesis work.

2.3.3 Development of DSO

The publication of DSO [15] by Engel, Koltun, and Cremers represents a major advancement in direct SLAM, building upon techniques from LSD-SLAM while introducing several innovations that have shaped modern direct SLAM research.

DSO's primary contribution was its departure from the geometric smoothness prior commonly used in previous direct SLAM systems. Earlier methods assumed that neighboring pixels should have smoothly varying depth values. In comparison, DSO employs a photometric loss formulation that does not enforce a smooth geometry prior. This formulation allows DSO to jointly optimize all parameters, including the depth of pixels. This contrasts with LSD-SLAM's approach of separately estimating pixel depths through multiple stereo comparisons.

By eliminating the smoothness prior, DSO has no mathematical restrictions on its pixel-point selection. This unrestrained point selection strategy enables more robust tracking compared to both previous direct methods (constrained by smoothness priors) and indirect methods (limited to feature detection). DSO selects points based on their intensity gradient magnitude distributed uniformly across the entire image, allowing tracking of both strong features and subtle intensity variations such as shading on texture-less walls. This sparse gradient-based sampling strategy makes DSO the first sparse direct SLAM system. Furthermore, DSO introduced a comprehensive photometric calibration model that accounts for real-world camera effects including camera exposure, vignetting, and gamma. When calibration parameters are unknown, DSO jointly optimizes an affine brightness transfer function alongside geometric parameters to handle lighting variations between frames.

DSO's innovations in photometric optimization and point selection enabled it to achieve superior accuracy and robustness compared to contemporary direct and indirect SLAM systems. Its strong performance has made it the foundation for numerous subsequent direct SLAM systems, which are outlined in Appendix A.3.

2.3.4 Short Summary of DSO

Because DSO serves as the foundation for MGSO, this section provides a brief overview of its key concepts. To maintain brevity, this discussion will focus on the key aspects of

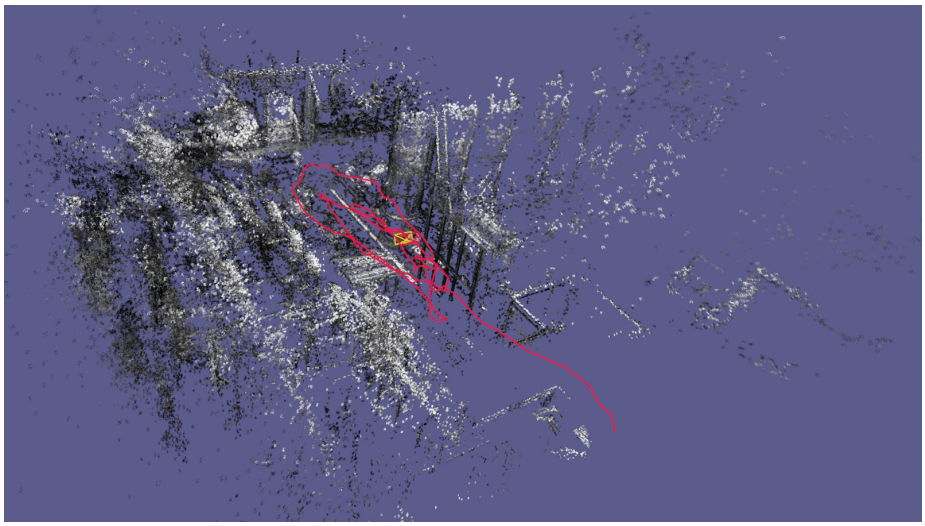


Figure 2.4: DM-VIO operating on a drone dataset. DM-VIO is an extension of DSO that integrates an IMU, but they output the same visuals. The colored points in the image is the map representation generated by the pixel points. The red lines and camera are the calculated pose, and the yellow camera is the ground truth.

DSO, while omitting less critical details. If more information is desired, detailed technical aspects of DSO are discussed in the Appendix C.2 and in the original publication.

An important concept to understand about DSO is it's sparse direct approach. While traditional direct methods track dense pixel patches and indirect methods track feature descriptors, DSO tracks individual pixels points distributed sparsely across the image. Each pixel point consists of a small patch of eight pixels, which is arranged in the pattern shown in figure 2.5. Each of the pixel in the pixel point shares the same geometric position of the pixel points for optimization efficiency, but provides a separate photometric residual. These tracked pixels points function conceptually similar to feature points in indirect systems, but without requiring explicit feature detection or matching. Each tracked pixel is anchored to a host keyframe and maintains a list of observations in subsequent frames where it is visible.

DSO's tracking and mapping operations are both calculated by a photometric error minimization equation. The loss of this equation evaluates the intensity differences between frames i and j for each valid pixel point in frame F_i. This equation can be formulated as:

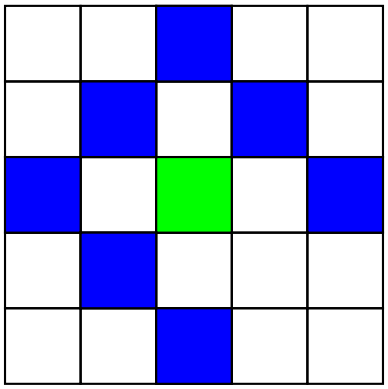


Figure 2.5: Every pixel point consists of eight pixels. This pattern was determined as the best through experiments by the original DSO paper.

$$E := \sum_{\boldsymbol{p} \in F_i} \left((I_j[\boldsymbol{p'}] - b_j) - \frac{s_j e^{a_j}}{s_i e^{a_i}} (I_i[\boldsymbol{p}] - b_i) \right) \tag{2.1}$$

$$\mathbf{p}' = \Pi_c(\mathbf{R}\Pi_c^{-1}(\mathbf{p}, 1) + d_{\mathbf{p}}\mathbf{t}) \tag{2.2}$$

Where:

- $\boldsymbol{p} = (u, v)$ are the location of the pixel point in pixel coordinates. $\boldsymbol{p'}$ is the new coordinate of the pixel point after the transformation.
- F_i represents the set of all valid pixel points in frame i
- a and b are affine variable to handle changes in photometric conditions, and s is the exposure time.
- $\boldsymbol{R}$ is a rotation matrix, $\vec{t}$ is a translation vector, and $d_{\boldsymbol{p}}$ is the inverse depth
- $\boldsymbol{K_c}$ is a camera calibration matrix with variables $\boldsymbol{c} = [f_x, f_y, c_x, c_y]$. Π_c represents a projection from 3D to 2D using $\boldsymbol{K_c}$, and the inverse of Π_c is the transformation back to 3D.

DSO operates two threads: tracking and mapping. The tracking thread runs at camera frame rate and estimates the camera's pose transformation between consecutive frames. It operates by minimizing the photometric error between the new input frame and the most

recent keyframe while assuming that the pixel point depths are constant. To compute this error, the system first creates a pixel depth map by projecting all valid points from all other keyframes onto the reference keyframe. The photometric loss is calculated by projecting the pixel points, and a non-linear least-squares optimization is done using the Gauss-Newton method. **The optimization determines the transformation and photometric adjustment that best align the observed intensities between frames while keeping the depths of the pixel points constant**.

The mapping thread is performed when new keyframes are introduced into the system. Keyframe insertion is triggered by a heuristic that detects substantial changes in either camera motion, occlusion, or illumination. During keyframe creation, the system identifies new candidate points using a designed pixel selection heuristic that ensures that points are well-distributed and have sufficiently high relative image gradient magnitudes. Candidate points have initial depth estimates computed by stereo matching between frames, which is inspired by techniques from LSD-SLAM. The mapping process optimization process is a sliding-window photometric bundle adjustment that jointly refines point depths for all tracked features, camera poses for all active keyframes, and photometric parameters using a windowed Gauss-Newton algorithm.

Marginalization is also completed at keyframe insertion to ensure that the optimization graph is computationally efficient. When the sliding window reaches its size limit or a keyframe is no longer relevant, a heuristic selects frames for marginalization that prioritizes keeping frames with visible points and frames that are spaced well from each other. Points are marginalized when they fall outside the field of view, are classified as outliers, or belong to a keyframe selected for marginalization. All marginalization is done using the Schur complement to reduce residual terms that affect sparsity.

2.4 DM-HSLAM: My contribution to the DSO family

My initial research focused on extending DSO by incorporating loop closure detection and IMU integration. This enhanced system is named DM-HSLAM [26]. Although this early work explores technical approaches different from the subsequent dense SLAM research in this thesis, it is included here to provide a complete record of my research during my Master's program. The basic principle behind DM-HSLAM is to combine two high-performance extensions of DSO: the direct-indirect hybrid system Hybrid Direct-Indirect Visual SLAM (HSLAM) and the IMU integrated Delayed Marginalization Visual-Inertial Odometry (DM-VIO).

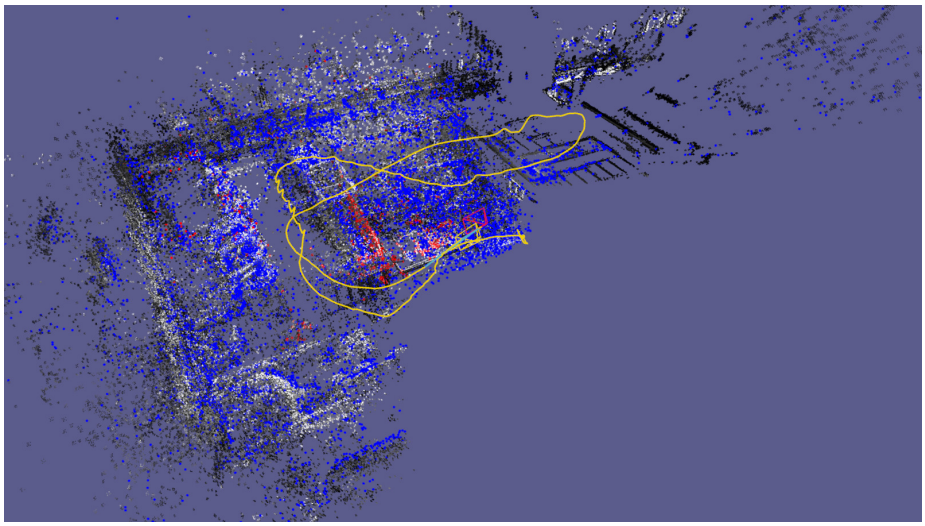


Figure 2.6: DM-HSLAM operating on a drone dataset. The coloured points are indirect key points and the gray points are the dense point cloud from the direct features. The system is also using IMU data, which cannot be shown graphically.

HSLAM, by Younes et al. [81], improves DSO through a unified optimization framework that performs joint photometric and geometric tracking using sparse direct points and indirect feature points. This hybrid approach addresses a fundamental limitation in DSO: sparse direct points change significantly when re-observed, making them unsuitable for loop closure. Effective loop closure requires landmarks that maintain consistent appearances across multiple observations over a long time, which is a characteristic of indirect feature points. Previous DSO extensions attempted to bridge this gap by associating feature-based information with direct sparse points, but restricted these indirect features to global mapping operations rather than integrating them into local tracking. This architectural separation may cause global and local maps to gradually drift apart because of their limited coupling. HSLAM resolves by having indirect points actively contribute to both tracking and mapping processes, thus maintaining robust alignment between local and global spatial representations.

DM-VIO, by Stumbery and Cremers, [69] enhances DSO through the integration of acceleration and rotational velocity data from an IMU. IMU data be directly added to the photometric loss optimization of DSO by pre-integrating the more frequent IMU data, which combines measurements into a single motion, to match the framerate. IMU integration enhances DSO in two ways: it maintains tracking during complete visual failures and enables absolute scale determination. This scale determination is possible because

IMUs measure physical motion in metric units. However, immediate scale estimates are unreliable, but waiting for accurate scale determination would impractically delay a SLAM system. DM-VIO solves this through a dual-graph architecture. The primary graph performs the standard visual inertial localization process, while a secondary graph preserves complete information until a reliable scale estimate is obtained. Once the system determines a reliable scale, it performs a complete graph optimization with the secondary graph to update the localization process with the refined scale information.

DM-HSLAM combines these systems through a new multi-objective optimization process that includes photometric, geometric, and inertial information:

$$\mathbf{E}(\xi) = W(e_{visual}) \left[\frac{\|\mathbf{E}_p(\xi)\|}{n_p \sigma_p^2} + K \frac{\|\mathbf{E}_g(\xi)\|}{n_g \sigma_g^2} \right] + E_{imu}(\xi) + E_{prior}(\xi) \tag{2.3}$$

Where $E_p(\xi)$ is the photometric residual, $E_g(\xi)$ is the geometric residual, and $E_{imu}(\xi)$ is the inertial residual. The photometric residual is the same one used by DSO, the geometric residuals are the distances between matched points once re-projected, and the inertial residual is the deviation of the current state from the state predicted by the pre-integrated IMU measurements. To normalize the influence between the photometric and geometric residuals, each energy is divided by n, the count of each type of feature, and σ^2, the residual variance. There is an additional K factor to reduce the influence of geometric residuals in low-texture or blurred environments where they perform worse:

$$K = \frac{5 \exp(-2l)}{1 + \exp(\frac{30 - N_g}{4})} \tag{2.4}$$

Where l is the pyramid level and N_g is the number of geometric matches that are inliers.

DM-HSLAM represents an advancement in SLAM architecture by combining three complementary data streams: photometric information (direct visual tracking), geometric features (indirect tracking), and inertial measurements. Each approach compensates for the weaknesses of the others. Direct methods excel by utilizing the full image data rather than just feature points, enabling robust performance in low-texture environments where traditional feature-based methods struggle. However, they're sensitive to lighting changes, which is offset by the inclusion of indirect feature tracking. Indirect features also enable reliable long-term loop closure. The inertial component further strengthens the system by maintaining tracking during visual failures, providing absolute scale information, and contributing additional motion constraints to improve overall accuracy. This three-way integration distinguishes DM-HSLAM in the field of SLAM systems, as most existing

approaches utilize only one or two of these complementary data sources. The result is a more resilient system that maintains high performance in a wider range of challenging environments.

2.5 Future of Direct SLAM

Although traditional direct and semi-direct SLAM systems primarily leveraged photometric methods to enhance tracking accuracy, their broader potential remained largely unexplored. Recent advances in radiance field techniques, particularly 3DGS and NeRF, have demonstrated that photometric approaches can excel in other aspects, such as dense reconstruction tasks. This thesis demonstrates that the fundamental techniques developed for direct SLAM systems remain relevant and can be effectively integrated with modern innovations that utilize photometric loss.

Chapter 3

First Attempts in combining Direct SLAM with Neural Scene Representations

3.1 Intro

MGSO is part of a family of SLAM systems that addresses a fundamental challenge in modern SLAM systems: the need for richer map representations. As SLAM finds use in tasks such as augmented reality, robotics, and digital replica generation, researchers are shifting SLAM's focus from pose accuracy to developing more useful map representations. Improvements in machine vision and autonomy have increased the need for highly accurate information-rich scene representations that can be generated in real time. Traditional SLAM systems are successful at precise tracking, but they use sparse feature point-based maps that offer limited utility for other tasks. To address this need, much current research focuses on SLAM that provides meaningful spatial information beyond simple location tracking.

This section will explore the challenges and history of dense SLAM, which provides dense volumetric maps of scenes. While traditional SLAM methods typically generate sparse maps, dense SLAM creates models that more faithfully capture the geometric and semantic properties of scenes. These maps are more valuable for applications that require detailed environmental understanding, such as robotic navigation and augmented reality. The section also highlights how the development of early dense SLAM systems has been

shaped by direct SLAM techniques, which share core principles with classical dense reconstruction approaches from computer vision. This section also provides details on the precursor system for MGSO, which was unsuccessful. MGSO emerged from lessons learned through the precursor system, whose limitations informed MGSO's development.

3.2 Dense SLAM History

3.2.1 Hand-Crafted Methods

Researchers have always tried to develop the ideal SLAM system. Traditional SLAM approaches have focused primarily on quantitative performance criteria: global consistency, precise pose tracking, robust noise handling, scalable mapping, and resilience to dynamic environmental changes. However, there is a set of qualitative criteria that is just as important: surface modeling, semantic information, and predicting unobserved regions. These qualitative aspects have always been an important focus for dense SLAM systems, which aim to create useful maps.

The choice of scene representation is fundamental to creating effective maps in SLAM. Although sparse point-cloud maps excel at pose tracking, they often fall short for more complex tasks. Handcrafted Computer-Aided Design (CAD) maps are considered the best maps for usefulness, offering rich semantic information and detail in a compact representation. However, generating CAD maps online in real time is infeasible. This creates a challenge for dense SLAM systems, which must strike a balance between two competing requirements: the ability to generate maps quickly and the need to capture sufficiently detailed actionable environment information. Traditional dense SLAM approaches have explored representations such as 3D grids (voxel), surfel clouds, and dense point clouds.

3D Grids (Voxels) KinectFusion, developed by Newcombe et al. [31], was a major advance in real-time dense SLAM systems that became the foundation for many subsequent voxel-based approaches. It combined the signed distance field representation with an RGB-D sensor to enable fast volumetric mapping of complex scenes. However, the computational demands of maintaining and updating a voxel representation can be significant, particularly as the mapped environment grows larger.

Surfel Clouds Surfel clouds represent 3D surfaces through collections of surface elements, which resemble oriented circular disks. When properly arranged with sufficient

overlap, these surfels collectively form a detailed representation of continuous surfaces. Several high-performance dense SLAM systems have successfully employed this technique, including BadSLAM [74] and ElasticFusion [62]. Although surfel clouds offer an efficient and flexible approach, their discrete and discontinuous nature presents challenges. The gaps between individual surfels can become noticeable, particularly when viewed from some angles.

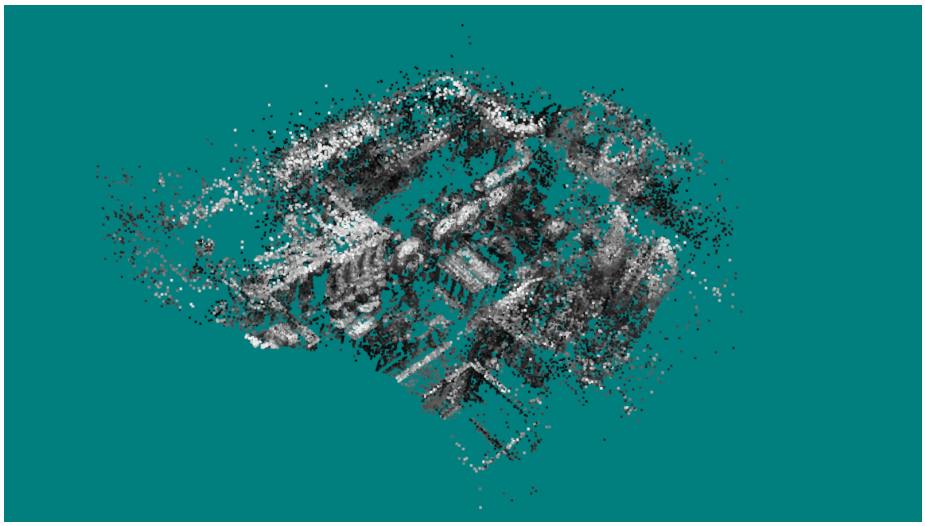


Figure 3.1: A dense pixel map created by the SLAM system Direct Sparse Odometry.

Dense Point Clouds The traditional approach to generating dense 3D representations begins with multi-view stereo techniques, which project every image pixel into 3D space by calculating correspondences across multiple views. These dense point clouds are then converted into continuous surfaces through mesh generation techniques. While this pipeline produces high-quality reconstructions, its computational demands make it primarily suitable for offline SfM applications. The direct and semi-direct SLAM methods, discussed in section 2.3, share some similarities with this approach. While these direct methods were some of the first SLAM systems to generate dense representations, their primary focus is on tracking performance rather than reconstruction quality. Although dense maps created by direct SLAM systems contain noise and errors, these imperfections are manageable for tracking purposes because they can be treated as outliers in optimization. However, such errors would significantly degrade the quality of a 3D reconstruction. An example of a dense map created by direct SLAM is shown in Figure 3.1.

Efficiently generating high-fidelity maps for real-time applications is still challenging regardless of the map representation. However, recent innovations in NeRF [46] and 3DGS [33] may present better representations for dense SLAM. The next section examines NeRF's impact on dense SLAM and the challenges it still faces. A comprehensive comparison of NeRF with the more recent 3DGS approach will be presented in the subsequent chapter.

3.2.2 A Brief comment on the Sensors used in Dense SLAM

Dense SLAM systems typically use visual sensors with depth-sensing capabilities to simplify the reconstruction process. It is difficult to algorithmically estimate depth information at all locations of an image, so having direct depth measurements for every pixel significantly simplifies the dense SLAM problem. However, depth sensors have limitations. First, depth sensors tend to have both minimum and maximum ranges. Depth measurements become unreliable both beyond their maximum range in open spaces and below their minimum range in close proximity. Second, sensors can struggle with complex geometries and unique materials that interfere with the depth-sensing process. These limitations have motivated continued research into purely monocular dense SLAM methods, which aim to achieve robust dense reconstruction using only RGB camera images.

3.2.3 NeRF Introduction

The idea of representing scenes as NeRF [46] revolutionized the field of computer graphics. At its core, NeRF converts a sparse set of images into a photorealistic 3D model of a scene using a neural network. The scene is represented as a radiance field: a representation that encodes how light interacts with a scene and travels through it.

The key innovation of NeRF lies in how it stores the radiance field. Rather than storing explicit geometry, NeRF stores the entire scene as a continuous function learned by a neural network. The scene is represented as a neural radiance field function that can be queried using a viewpoint (a location and viewing direction) to return a pixel value.

Because this radiance field function is continuous and differentiable, it can be trained to encode a volumetric representation of the scene. It is trained by giving the sparse set of input images and their associated viewpoints as input. The training compares the actual input photographs against the network's predictions from those same viewpoints, using a photometric loss that measures the pixel difference between real and generated images. As the system is optimized by calculating the gradient of the photometric loss, the internal weights of the radiance field function will be adjusted so that it returns the



Figure 3.2: Render of a fox trophy by InstantNGP. NeRF systems can only render areas where it is given images, so the dark and blurry areas are where the system received little or no images.

$$(x, y, z, \theta, \phi) \rightarrow N \rightarrow (RGB\sigma) \tag{3.1}$$

Figure 3.3: The position (x, y, z) and viewing direction (θ, ϕ) are sent into the neural network N. The network will output a pixel colour and density $(RGB\sigma)$. By querying multiple pixels, a whole image can be outputted.

input images at their viewpoints. Because the network learns a continuous representation of the scene, it can smoothly interpolate between the trained viewpoints to synthesize novel views. Furthermore, the network also learns lighting effects about the scene, which allows for complex lighting phenomena such as reflections and transparency. If depth data is available, the NeRF model can be adjusted to also include a depth output for every viewpoint query. Including depth data into NeRF will generally improve its accuracy.

After the introduction of NeRF, many papers followed to improve on its visual quality and speed. While the original NeRF sampled a scene using a single ray per pixel, MipNerf [4] replaced each ray with conical frustums, significantly improving performance across different image scales and resolutions. InstantNGP [47] introduced the idea of using a grid to significantly improve the training speed of NeRF. By augmenting the network with a set of multiscale occupancy grids, InstantNGP is able to skip sampling over empty or blocked areas, therefore significantly boosting the training speed to near instant levels.

3.3 Development of NeRF SLAM

3.3.1 Benefits of the NeRF Representation

The NeRF representation offers several advantages and properties that make it well-suited for dense SLAM applications:

- Gradient-based optimization process is very flexible and robust. The neural network is trained with an iterative learning process that can be paused, resumed, or modified at any point during reconstruction. This means that a partially optimized map can be saved and later refined with new observations. Reusing and building upon previously made maps is essential for any SLAM map representation.
- The continuous nature of NeRF allows for a smooth representations of scene geometry without harsh discontinuous or gaps.
- A continuous representation allows NeRF to automatically adapt the resolution dependent on the quantity of information in an area. Unlike discrete representations that have fixed resolutions, NeRF can adapt memory use and resolution to be specific to the detail at an area.
- Experiments with NeRF SLAM systems [87] have shown that the NeRF representation handles noise and dynamic objects well. The system achieves this because it naturally learns the average static representation shown in the images. This means that high-frequency noise and fast dynamic objects will be forgotten by the network.
- NeRF also demonstrates an ability to fill in gaps in unobserved or partially obscured areas. This is because of NeRF's nature to smoothly interpolate between observed viewpoints, including unobserved areas.

3.3.2 History of NeRF SLAM

Due to these potential benefits, researchers have begun to integrate NeRF with SLAM systems. Sucar et al. [66] created iMAP, the first RGB-D NeRF SLAM system. iMAP demonstrated NeRF's ability to generate detailed geometric scene representations with automatic detail control and the filling of unobserved regions. iMAP's localization process works by continuously refining camera poses through photometric optimization of it's renders. The NeRF map is adjusted by jointly optimizing the NeRF representation. At every

keyframe addition, iMAP will optimize both the poses and radiance field using photometric loss.

Building upon iMAP's foundation, subsequent research made improvements through increasingly sophisticated scene representations. NICE-SLAM [87] introduced a grid-based hierarchical representation, allowing the system to better capture the information of the local scene. Following NICE-SLAM, Vox-Fusion [79] took a hybrid approach that bridged the voxel and NeRF techniques. It encodes NeRF's scene information within individual voxels to allow for easy scene expansion and better performance. Taking a different approach, Point-SLAM [55] applied the NICE-SLAM technique with a neural point cloud instead of a grid, adjusting the NeRf density based on the information from the input data. This strategy offers a more precise control over the detail in NeRF's scene representation compared to grid-based approaches. A third strategy, exemplified by Orbeez-SLAM [11], integrates NeRF technology with proven monocular SLAM systems like ORB-SLAM. This hybrid approach leverages ORB-SLAM's robust camera tracking and depth estimation capabilities while using instant-ngp's fast NeRF implementation to create detailed 3D reconstructions in real time.

3.4 Reasons for Combining Direct SLAM and NeRF

NeRF and direct SLAM share fundamental characteristics that suggest synergy between the two approaches. Most notably, both systems rely on photometric loss for optimization. This key similarity motivated my investigation to determine whether combining these systems could enhance overall performance and results.

My initial research focused on advancing Orbeez-SLAM. Orbeez-SLAM relied on an indirect feature-based tracking system, and I hypothesized that incorporating a direct tracking approach using DSO could enhance its performance. This hypothesis led me to develop and experimentally evaluate a novel uncoupled architecture that combines DSO's sophisticated direct tracking capabilities for localization with NeRF's representation for dense 3D reconstruction.

3.5 Method

Orbeez-SLAM advanced real-time NeRF SLAM by decoupling camera tracking from scene reconstruction. The system passes poses from ORBSLAM directly to InstantNGP for neural rendering. Previous NeRF-based SLAM systems operated slowly because they required

high-quality NeRF renderings before tracking can be performed, creating a bottleneck. By separating these processes, Orbeez-SLAM enables real-time performance while maintaining reconstruction quality.

Orbeez-SLAM architectural choice presents a trade-off: separating tracking and mapping can potentially reduce reconstruction accuracy since these processes naturally should inform and refine each other. To address the trade-off between decoupled operation and reconstruction quality, Orbeez-SLAM attempts to leverage tracking information during reconstruction by integrating it with the density grid system. Modern NeRF systems use density grids to identify empty spaces and potential surfaces, allowing them to focus computational resources on areas that contribute meaningfully to the reconstruction. Orbeez-SLAM enhances this approach by using the point cloud from its localization system to inform the density grid, since SLAM naturally identifies points on real surfaces. Although the initial implementation used ORBSLAM for localization, DSO emerged as a potentially superior alternative because of its direct photometric optimization approach, which tends to generate point clouds that better represent continuous surfaces.

3.6 Results and Discussion

Experimental evaluation of the modified system on the Replica dataset [64] revealed that the architectural changes produced no measurable impact on system performance. Specifically, using DSO's theoretically superior point cloud did not improve reconstruction quality and tracking accuracy remained unchanged. This result suggests two possible interpretations. First, DSO's point cloud representation might not actually provide better surface information. Alternatively, this outcome might indicate that Orbeez-SLAM decoupled approach between tracking and reconstruction is so complete that even substantial changes in the input point cloud quality have minimal impact on the reconstruction.

Further experiments provided strong evidence that the decoupling between tracking and reconstruction was indeed the primary issue. Detailed visualization analysis revealed a significant spatial drift between the NeRF's reconstructed geometry and the input point clouds from the SLAM system. This is shown in Figure 3.4. This drift indicates that the initial geometric hints provided by the tracking system were not sufficient in influencing the NeRF's optimization process.

Although the density grid initialization using SLAM point clouds was intended to create a bridge between the tracking and reconstruction systems, this connection proved too weak to effectively anchor the NeRF reconstruction to the tracked geometry. This divergence



Figure 3.4: Two images of the NeRF representation and the input points. These images show that the two representations drifted from one another. The shape of the points cloud (small red and cyan dots) is slightly offset from the NeRF generated scene.

suggests that the density grid initialization alone, without stronger geometric constraints, is insufficient to maintain consistency between the two decoupled systems.

3.7 Next Steps

The experimental results did not demonstrate the anticipated synergy between the NeRF representation and direct SLAM systems. Further analysis revealed that the two systems remained too decoupled to meaningfully influence each other's performance. Although this particular implementation did not achieve the desired results, a more tightly integrated approach combining NeRF and direct SLAM may prove effective. Current NeRF SLAM systems rely on basic photometric optimization across entire images for tracking. Incorporating techniques from semi-direct systems such as SVO or DSO into the tracking optimization process could enhance both computational speed and accuracy. Additionally, alternative radiance field representations, such as 3DGS, might offer superior performance because their similarity to dense SLAM extends beyond photometric loss, which is explored in the following section.

Chapter 4

Monocular-GSO: Harnessing the Synergy of 3DGS and Direct SLAM

4.1 Introduction



Figure 4.1: Renders of MGSO on different scenes. MGSO is able to generate high quality scenes online in real-time. © 2025 IEEE

Monocular-GSO (MGSO) advances the field of dense volumetric VSLAM through the novel integration of 3DGS with photometric SLAM techniques. MGSO tackles a core challenge in current dense SLAM systems: striking a good combination of hardware efficiency, computational speed, and high-quality reconstruction. Current dense SLAM systems often excel in one or two of these aspects but struggle to excel in all three dimensions.

MGSO is a dense VSLAM system that performs real-time online operation 3D reconstruction using only a single monocular camera. MGSO leverages 3DGS, a scene representation that has shown promising properties for SLAM. By combining 3DGS with the photometric SLAM system DSO in a synergistic framework, MGSO introduces an efficient solution for real-time dense 3D reconstruction. MGSO addresses the challenging problem of online Gaussian initialization during SLAM operations through the integration of photometric SLAM techniques. The tracking part of MGSO generates structured dense point clouds that serve as optimal initialization points for 3D Gaussians. This integration accelerates the optimization process and enables the creation of memory-efficient maps.

Experimental results demonstrate MGSO's superior performance across multiple standard benchmarks, including the Replica, TUM-RGBD, and EuRoC datasets. MGSO achieves a combination of reconstruction quality, memory efficiency, and speed that outperforms current state-of-the-art methods. Notably, it operates effectively with only RGB input and maintains high performance even on mobile-level hardware. These features make MGSO particularly valuable for practical applications in robotics, augmented reality, and other real-time scenarios requiring dense 3D reconstruction.

The research methodology in this section draws on a preprint version of a paper[27] that has been accepted for publication at the IEEE International Conference on Robotics and Automation (ICRA) 2025. While the textual content of this chapter has been substantially revised compared to the pre-print, several figures and tables from the original pre-print and contained in the final publication have been reprinted.

4.2 3D Gaussian Splatting for SLAM

This section explores the potential of 3DGS for SLAM systems, providing context on the development and advantages of MGSO. This section begins by examining the core principles of 3DGS. Building on this foundation, the unique characteristics that make 3DGS particularly valuable for SLAM applications are analyzed, followed by a comprehensive review of existing VSLAM systems that have integrated 3DGS.

4.2.1 3D Gaussian Splatting

3DGS is an approach to scene representation that models the environment as a large set of 3D Gaussians, which resemble blurry semitransparent overlapping clouds. Each Gaussian store location, orientation, scale, and color information. The rendering of images involves

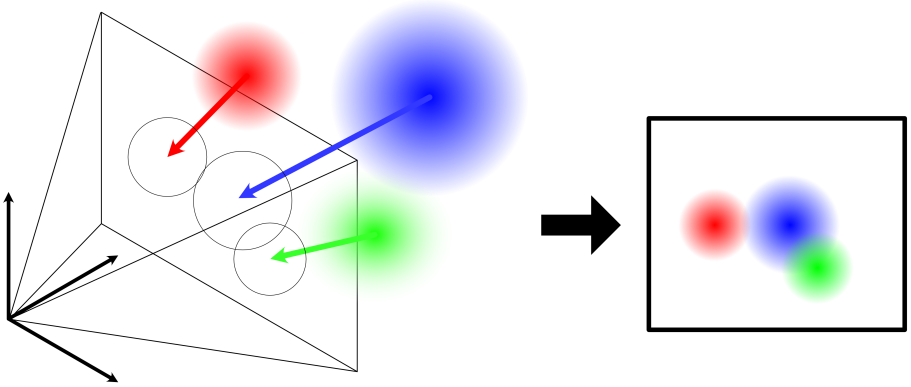


Figure 4.2: Diagram showing how 3DGS operates. The Gaussians are rasterized by sorting them by depth and projecting them onto the camera plane.

a rasterization process that projects Gaussians onto the camera plane in a manner where they are sorted by depth and blended front-to-back.

Like NeRF, 3DGS optimizes scene reconstruction through gradient descent on a photometric loss, comparing rendered outputs to training images. The optimization adjusts each Gaussian's parameters to better match the ground truth views. Unlike these continuous parameters, the creation and removal of Gaussians cannot be directly optimized through gradients. Instead, 3DGS employs heuristic rules to determine when to split Gaussians in undersampled regions or remove them from oversampled areas, dynamically adapting the scene representation's density. More details of how 3DGS's operates are provided in Appendix B.1.

4.2.2 Typical 3DGS Pipeline

Traditional 3D Gaussian Splatting begins with a pre-processing stage that uses a SfM program like COLMAP [58] to establish initial camera poses and a sparse point cloud. The points from the sparse point cloud serve as seeds for the first 3D Gaussians. The system then enters an optimization phase where it refines each Gaussian's to minimize the difference between renders and the original input images. Throughout this process,

the system heuristically manages the Gaussian density by deleting or cloning the existing points. However, when integrating 3DGS with VSLAM, this pipeline must be modified to give continuous camera poses and initial points.

4.2.3 Improvements over NeRF for SLAM

Similarly to NeRF, 3DGS provides a dense, differentiable scene representation ideal for SLAM applications. It has all the advantages of NeRF discussed in section 3.3. However, 3DGS offers several key advantages: explicit 3D geometry, real-time rendering capabilities, and lack of natural bounds.

- 3DGS uses explicit 3D Gaussian primitives that directly encode geometric and appearance information. This explicit representation enables direct manipulation of the geometry of the scene through operations such as splitting, merging, and measurement. It also simplifies conversion to other 3D representations, whereas NeRF typically requires an intermediate conversion step to extract explicit geometry.
- 3DGS achieves real-time rendering speeds exceeding 100 frames per second through efficient rasterization of Gaussian primitives. Unlike NeRF's ray-based approach, which must process empty space along each ray, 3DGS directly projects and blends only the relevant Gaussians onto the image plane. Furthermore, the original implementation uses tile-based rasterization with pre-sorting for parallel processing. This fast rendering is particularly crucial for SLAM applications, where real-time operation is considered essential.
- The original NeRF implementation has to pre-define the bounds of the scene to limit the area where ray calculations and sampling occurs. 3DGS naturally handles unbounded environments through its Gaussian representation. The rasterization process only considers visible Gaussians within the camera-frustum, eliminating the need for explicit volume boundaries. This property makes 3DGS particularly suitable for the large-scale environments and open scenes that are common in SLAM.

4.2.4 Prior 3DGS SLAM Systems

The potential of 3DGS for dense SLAM has sparked the development of numerous 3DGS-based SLAM systems. Early implementations generally follow two architectures: decoupled

and coupled. Decoupled systems maintain a clear separation between tracking and reconstruction, typically employing traditional SLAM for pose estimation while running dense 3DGS reconstruction in parallel. In contrast, coupled systems tightly integrate 3DGS reconstruction with both mapping and tracking, enabling more accurate scene understanding, but introducing computational challenges. These coupled systems require a high-quality dense map for accurate localization, but building this map takes time, creating performance bottlenecks.

Early systems like MonoGS [45], SplaTAM [32], GS-SLAM [75], and Gaussian-SLAM [83] pioneered tightly coupled architectures where the tracking and mapping phases share a unified photometric optimization framework. This coupled approach was further refined in subsequent systems such as CG-SLAM [25], RTG-SLAM [51], and SplatSLAM [56]. While coupled systems tend to produce better reconstructions, their dependency on map refinement before tracking results in slow performance. The computational demands of coupled optimization led to the development of uncoupled architectures, such as Photo-SLAM [30], IG-SLAM [57], and GS-ICP [22]. It can be observed from Table 4.1 that most coupled systems struggle to achieve speeds that would be considered real-time. Another critical limitation shared across most early 3DGS SLAM systems is their reliance on RGB-D sensors for depth information, which constrains their practical applicability.

Table 4.1: Existing 3DGS SLAM Systems © 2025 IEEE

Name	Type	Possible Sensors	FPS
MonoGS [45]	Coupled	RGB,RGB-D	<5
SplaTAM [32]	Coupled	RGB-D	<5
GS-SLAM [75]	Coupled	RGB-D	>5, <10
Gaussian-SLAM [83]	Coupled	RGB-D	<5
CG-SLAM [25]	Coupled	RGB-D	>15, <20
RTG-SLAM [51]	Coupled	RGB-D	>15, <20
SplatSLAM [56]	Coupled	RGB	<5
GS-ICP [22]	Decoupled	RGB-D	>30
Photo-SLAM [30]	Decoupled	RGB*,RGB-D	>30
IG-SLAM [57]	Decoupled	RGB	>5, <10
MGSO	Decoupled	RGB	>30

*Both monocular and stereo.

MGSO improves on earlier 3DGS-based SLAM by achieving high-quality reconstruc-

tions at real-time speeds using RGB data alone. MGSO achieves speeds exceeding 30 frames per second on standard and mobile hardware, which is only matched by Photo-SLAM and GS-ICP (see Table 4.1). Among existing approaches, MGSO shares the closest architectural parallels with Photo-SLAM, which combines 3DGS with ORBSLAM [48], as both systems build upon the integration of 3DGS with traditional SLAM. However, Photo-SLAM tends to generate large, memory-inefficient maps because ORBSLAM does not create good initializations for 3DGS. GS-ICP achieves fast processing speeds by combining 3DGS with Iterative Closest Point (ICP) tracking, but it relies on depth sensor data to initialize the iterative closest point algorithm. In contrast, MGSO can operate only using monocular cameras. IG-SLAM [57], which integrates DROID-SLAM [67] and 3DGS, is also able to only use RGB data, but its machine-learned pseudo-depth module adds computational overhead that compromises real-time performance. MGSO demonstrates that such compromises are unnecessary, achieving both real-time processing speeds and high-quality reconstructions while maintaining memory-efficient map representations.

4.3 Main Ideas behind MGSO



Figure 4.3: Comparison of 3DGS point clouds from MGSO and the original 3DGS on Replica room0. Left: Map from original 3DGS after 10,240 iterations with Gaussian size set to 0.1. Right: MGSO input point cloud. These points clouds are quite similar in structure. © 2025 IEEE

The similarities between 3DGS and direct SLAM systems suggest that their integration would be beneficial. At their core, both techniques optimize scene representations using photometric loss, but unlike NeRF and direct SLAM, their connection goes deeper than their optimization. The underlying representation in 3D Gaussian Splatting can be conceptualized as an enhanced point cloud, where each point carries additional volumetric

parameters. When these Gaussians are stripped down to points, fully optimized 3DGS point clouds bear a striking resemblance to the dense point clouds generated by SLAM systems. This can be observed in Figure 4.3. This structural parallel suggests an inherent compatibility between the two approaches, potentially enabling dense SLAM systems to leverage direct SLAM and 3DGS to generate rich representations in real-time. Drawing inspiration from these intrinsic similarities, I developed MGSO, a hybrid system that harmoniously combines the direct SLAM and 3DGS methodologies to take advantage of the distinct advantages of each approach.

MGSO introduces an uncoupled architecture in which a photometric SLAM system continuously performs localization while providing point clouds and poses a concurrently operating 3DGS module. The name MGSO combines its foundational technologies: DSO, which provides the underlying SLAM capabilities, and Gaussian Splatting (GS), which provides the reconstruction capabilities.

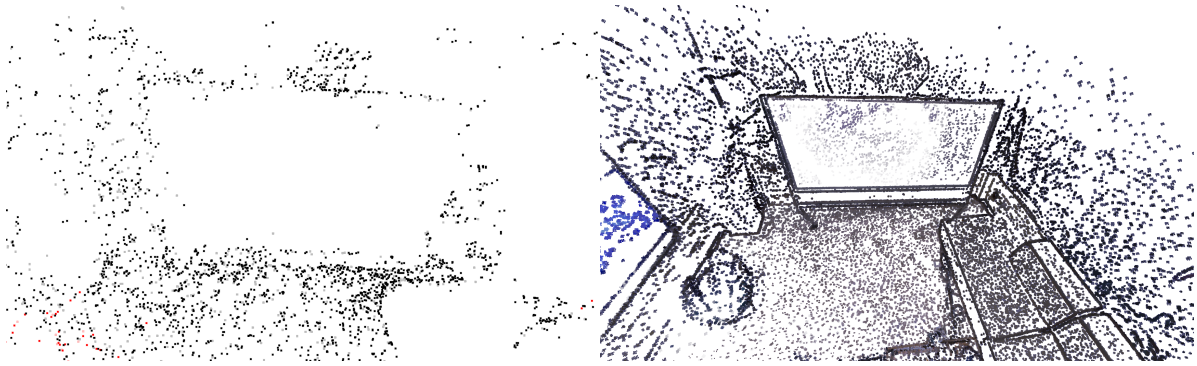


Figure 4.4: Comparison of the input point clouds from MGSO and point clouds from ORBSLAM, an indirect SLAM system. MGSO's dense point cloud capture details in the scene much better than ORBSLAM. © 2025 IEEE

A weakness in uncoupled 3DGS SLAM systems is that while both modules function independently, they fail to complement each other's strengths. 3DGS requires high-quality initial point clouds for optimal reconstruction. Conventional feature-based SLAM systems generate sparse point clouds, which makes them suboptimal for initializing 3DGS. This can be seen in Figure 4.4. MGSO's tracking system is specially designed to cater to the needs of 3DGS. Unlike traditional feature-based SLAM systems that track only distinctive points in the environment, MGSO can track from a much larger set of image pixels to create initial point clouds of higher density and structure. MGSO's tracking generates a dense, well-structured point cloud that captures the scene's geometry in greater detail and

density. This dense point cloud is able to help the 3DGS reconstruction process in two ways.

4.3.1 Initializing 3DGS in New Areas

Precise positioning of initial Gaussian seeds profoundly impacts 3DGS performance because the system's Gaussian management relies entirely on existing primitives. Since new Gaussians are created by cloning existing ones, poor initial distribution leads to suboptimal scene coverage. Furthermore, experiments by Lu et al. [41] show that 3DGS does not penalize excessive Gaussian numbers effectively. This means that initializing too many redundant Gaussians in an area will affect the memory efficiency of the final map. The significance of proper initialization is also quantitatively demonstrated in the original 3DGS paper [33], where random initialization resulted in a substantial 6 Peak Signal-to-Noise Ratio (PSNR) decrease in reconstruction quality.

MGSO's tracking system is specifically engineered to generate dense, well-structured point clouds that serve as optimal initialization seeds for 3D Gaussian Splatting (3DGS). This careful integration provides 3DGS with high-quality geometric priors when mapping new regions of the environment, enabling more efficient and accurate scene reconstruction. By generating initial point clouds that closely approximate the expected distribution of optimized Gaussian primitives, the system significantly accelerates the overall optimization process.

Strategic placement of points in the initial cloud offers two key advantages. First, it reduces the need for cloning operations, since points are already positioned in relevant locations throughout the scene. Second, it minimizes necessary pruning operations by avoiding redundant Gaussian placement at the start. In contrast, traditional SfM and uncoupled 3DGS SLAM systems produce point clouds that are mere by-products of the pose estimation process. The poorly distributed point clouds of these systems often provide suboptimal initialization for 3DGS, resulting in slower optimization convergence, degraded reconstruction quality, and inefficient memory utilization.

4.3.2 Assisting Primitive Cloning and Pruning

Ablation studies with MGSO reveal an interesting relationship between reconstruction parameters and system efficiency. When the frequency of Gaussian cloning and pruning operations is reduced, the system maintains its rendering performance with minimal degradation in quality. More significantly, this reduction in dynamic operations leads to

improved memory efficiency in the final reconstruction. The heuristically cloning and pruning process is the weakest aspect of the original 3DGS paper because it lacks theoretical guarantees of optimality, relying heavily on carefully tuned parameters. The experimental results indicate that MGSO appears to provide better candidates for Gaussian cloning compared to the original heuristic approach.

4.4 Method

MGSO has two core components that operate in parallel: a SLAM module and a 3D dense reconstruction module.

4.4.1 SLAM module

MGSO's tracking system is based on DSO, which was discussed earlier in this thesis in Subsection 2.3.4. DSO was chosen as the foundation because its pixel selection strategy generates point clouds that most closely approximate the distribution of fully optimized 3DGS primitives. Compared to other direct SLAM systems, DSO deliberately distributes pixels across the entire image space including harder-to-track areas where there are less gradients, creating a more uniform and comprehensive spatial representation. The difference between the point tracking of a direct SLAM system and an indirect SLAM system can be observed in Figure 4.5.

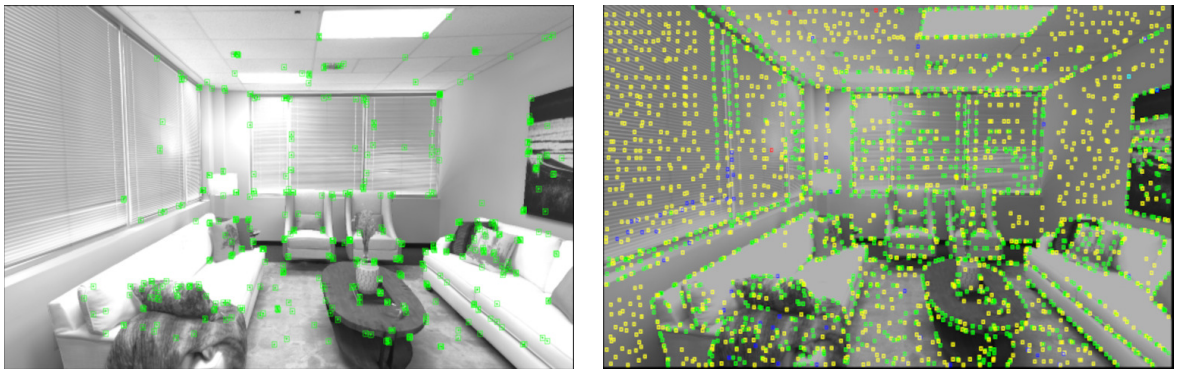


Figure 4.5: Tracked points from ORBSLAM (left) compared to MGSO (right). MGSO tracks much more points, which results in denser point clouds outputs. © 2025 IEEE

The dense point cloud structure of DSO is due to its pixel selection strategy. Rather than only tracking obvious features like corners, DSO systematically samples pixels across the entire image through a multi-pass approach. It first divides the image into small blocks, selecting the highest-gradient pixel above a threshold in each block, then repeats this process using larger blocks with lower thresholds. There are three passes in total. This pixel selection approach ensures detailed tracking in visually complex regions while maintaining adequate coverage in areas with less distinctive features. This approach aligns well with 3DGS, as both techniques maintain representation throughout the entire scene while concentrating detail in complex areas.

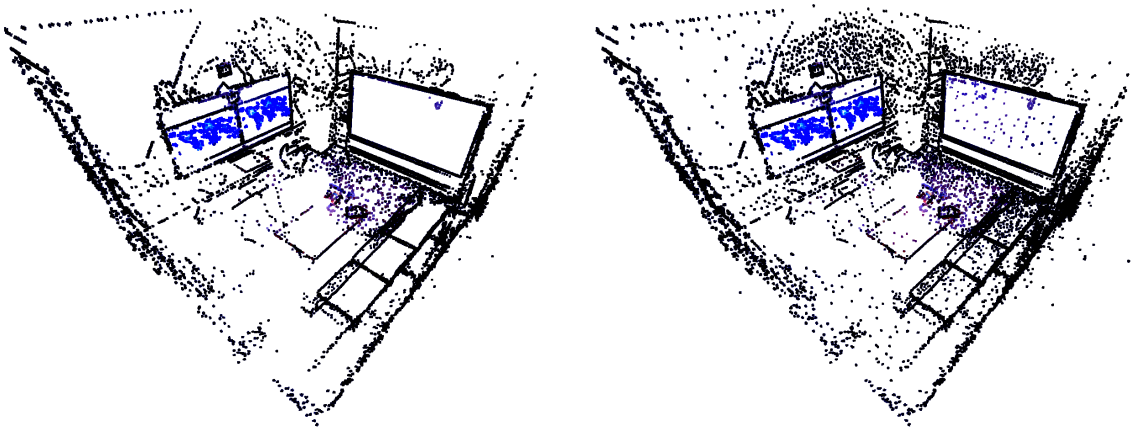


Figure 4.6: Original DSO Point cloud (left) compared MGSO's point cloud (right). MGSO has much more output points, especially in flat low-gradient regions. © 2025 IEEE

Experiments revealed that while DSO's default pixel density works well for tracking, 3D Gaussian Splatting performs better with higher pixel density settings. This is particularly true in areas with no or low gradients. To address this, DSO was modified to track additional pixels that do not participate in pose estimation, allowing optimal performance for both tracking and 3D reconstruction. However, completely flat regions with no gradients remain a challenge because DSO requires some gradient for tracking, while pixels in totally gradient-less areas cannot be tracked at all regardless of DSO's settings.

MGSO testing revealed that 3DGS performs better with approximate point estimates in flat areas than with no points at all. To address regions where gradient-based tracking fails, an interpolation system using Delaunay Triangulation is used. The interpolation approach leverages Delaunay Triangulation to partition the image using tracked pixels as vertices, creating a mesh of non-overlapping triangles. Within each triangle, the depth

of new points is estimated through a weighted average of the known depths at its three vertices, with weights inversely proportional to the distance between the new point and each vertex. This method is particularly effective for the planar surfaces where it is most needed, since it assumes a linear depth variation across each triangular region. While other SLAM systems like Photo-SLAM also use point interpolation, MGSO's method achieves superior results through two advantages: it starts with a higher density of tracked points, and it only applies interpolation in planar regions where depth can be estimated most accurately. The differences between the input point clouds of DSO and MGSO can be observed in Figure 4.6.

4.4.2 Dense Reconstruction module

MGSO employs 3DGS as its foundation for dense reconstruction. Similarly to Mono-GS [45], the system does not use spherical harmonics, a modification that delivers computational gains while preserving reconstruction quality. The rendering pipeline implements the original differentiable tile-based rasterization approach established in 3DGS.

To enhance system speed, this work implements the image pyramid learning approach from Photo-SLAM [30] for progressive training of the Gaussian map. The pyramid structure significantly accelerates training, particularly in live video scenarios. The method operates by first generating a multiscale image pyramid through smoothing and downsampling of ground-truth images. During the training phase, photometric loss calculations begin at the highest pyramid level and progressively transition to lower levels as optimization advances. The implementation utilizes an optimized CUDA backend [50], which achieves higher performance speeds compared to the original 3DGS implementation.

Following Photo-SLAM's integration with ORBSLAM [8], MGSO's point handling approach is customized to complement the point clouds generated by its SLAM module. Due to the well-structured nature of MGSO's point clouds, the system performs cloning and pruning operations less frequently than comparable systems. The adaptive control strategy implements periodic densification and pruning of Gaussians at 1000 iteration intervals. During keyframe processing, new Gaussians are initialized by extracting both position and color information from the SLAM-generated point cloud.

Initial experiments revealed floating artifacts when using 3DGS's standard densification and pruning methods. To address this challenge, MGSO incorporates AbsGS adaptive control strategies [80]. The system periodically identifies Gaussians with high homodirectional view-space position gradients for densification through strategic splitting and cloning operations. The algorithm splits large Gaussians with high variance while cloning smaller ones

in under-reconstructed regions. To eliminate transparent floating artifacts, the pipeline systematically prunes low-opacity Gaussians. The implementation maintains the original splitting and cloning parameters from 3DGS.

4.5 Results

MGSO is evaluated against the contemporary 3DGS dense SLAM systems: MonoGS [45], GlORIE-SLAM [56], Splat-SLAM [32], IG-SLAM [57], and Photo-SLAM [30].

Datasets: The datasets used for evaluations are described in table 4.2. These datasets are commonly used to evaluate traditional SLAM and dense SLAM systems.

Table 4.2: Replica and TUM-RGBD have depth data, but it is unused in the evaluations

Dataset	Publisher	Type
Replica [64]	Facebook Research	Synthetic
EuRoC-MAV [7]	ETH Zurich	RGB Drone
TUM-RGBD [65]	Technical University of Munich	RGBD Handheld

Hardware: The hardware used is described in table 4.3.

Table 4.3: The hardware used is standard for the Dense SLAM field

Run	CPU	GPU
Replica	Intel i9-14900K	NVIDIA RTX 4090
Replica (mobile)	Intel i7-12700H	NVIDIA RTX 3080 mobile
EuRoC-MAV and TUM-RGBD	Intel i5-12600KF	NVIDIA RTX 3090

Experimental Setup: 3DGS operates with its default optimization parameters, with only the densification interval modified to 1000. The SLAM component implements DSO's [15] default tracking configuration. The parameters for the untracked points were determined through a combination of qualitative analysis and empirical evaluation. First, we visually examined the output points across many different parameter settings. Additionally, we conducted evaluations on the Replica dataset for different untracked points settings, but each scene of Replica was run a single time to save time. The parameter tuning for the untracked points were done separately from the presented evaluations.

The EuRoC MAV dataset experiments incorporate a preprocessing pipeline that undistorts and crops images before their integration into the SLAM systems. This preprocessing step proved essential for resolving pose alignment discrepancies between the undistorted 3DGS map and the distorted ground truth images. Reevaluation of Photo-SLAM using the pre-processed dataset revealed substantial performance improvements compared to previously reported results.

Evaluation: For a fair comparison, the MGSO evaluation uses the original 3DGS evaluation scripts. The reconstruction quality assessment uses established dense reconstruction image metrics: PSNR, Structural Similarity Index Measure (SSIM) [73], and Learned Perceptual Image Patch Similarity (LPIPS) [73]. The use of an independent evaluation framework, rather than a system-specific one, ensures results better reflect real-world scenarios. The testing methodology captures real-world performance factors, including potential misalignment between poses and the 3DGS map.

Following standard practice in dense SLAM evaluation, the evaluation examines every fifth frame. Photo-SLAM's evaluations were updated using this protocol for a fairer comparison. The experimental validation comprised ten runs each for the Replica and EuRoC-MAV datasets, and five runs for TUM-RGBD. The results for other dense SLAM systems were obtained from their respective publications, with the exception of Mono-GS, whose results were obtained from Splat-SLAM [56].

The 3DGS evaluation methodology introduces a bias favoring slower systems because extended training periods enhance 3DGS's performance. This performance-speed tradeoff should be considered when interpreting comparative results. Because MGSO inherits real-time handling from DSO, real-time processing speeds are enforced during evaluation to make the results more realistic. The evaluation framework processes Replica and TUM-RGBD datasets at 30 fps, while EuRoC operates at 20 fps.

4.6 Discussion

4.6.1 Localization

Table 4.4: Absolute Trajectory Error of Tracking on Replica (RMSE in cm) © 2025 IEEE

Method	r0	r1	r2	o0	o1	o2	o3	o4	Avg.
Photo-SLAM	0.58	**0.32**	**5.03**	0.47	0.58	0.35	1.18	0.23	**1.09**
MGSO	**0.35**	1.02	5.93	**0.22**	**0.54**	**0.28**	**0.34**	**0.2**	1.11

While tracking metrics are reported, MGSO focuses primarily on reconstruction rather than localization accuracy. The implementation employs a largely unmodified DSO tracking system for pose estimation, thus inheriting the tracking performance from the original DSO paper [15]. The comparable tracking results between MGSO and Photo-SLAM indicate that differences in rendering quality stem from the reconstruction methods rather than pose estimation accuracy. The tracking results are reported in Table 4.4.

Reconstruction Quality

Table 4.5: Reconstruction Results on Replica (cm) © 2025 IEEE

Method	metric	o0	o1	o2	o3	o4	r0	r1	r2	Avg.	MS	FPS
PS	PSNR ↑	35.22	34.35	**29.58**	28.55	**32.05**	26.75	27.78	29.43	30.46	22.5	>30
	SSIM ↑	0.94	0.93	0.91	0.89	0.92	0.79	0.84	0.89	0.89		
	LPIPS↓	**0.21**	**0.23**	**0.26**	**0.26**	**0.22**	**0.31**	0.28	**0.24**	**0.25**		
MGSO	PSNR ↑	35.85	37.15	29.19	**30.44**	30.08	27.71	29.50	31.33	31.41	**4.6**	30
	SSIM ↑	0.94	0.94	0.90	0.90	0.91	0.79	0.86	0.91	0.89		
	LPIPS↓	0.22	0.25	0.29	**0.26**	0.26	0.33	**0.27**	**0.24**	0.27		
MGSO (l)	PSNR ↑	**36.34**	**38.20**	28.90	30.27	31.41	**28.11**	**30.04**	**31.89**	**31.90**	5.2	30
	SSIM ↑	**0.95**	**0.96**	0.90	**0.91**	**0.93**	**0.82**	**0.87**	**0.92**	**0.91**		
	LPIPS↓	0.24	0.25	0.31	0.27	0.25	0.35	0.29	0.26	0.28		

MGSO demonstrates superior reconstruction quality in all evaluated datasets. On the Replica sequences, MGSO achieves a PSNR of 31.406 dB while maintaining a compact map size of 4.618 MB, outperforming Photo-SLAM in both metrics. The mobile variant of MGSO further improves these results, reaching 31.896 dB PSNR. The results for the Replica data set can be observed in Table 4.5.On the more challenging EuRoC dataset, MGSO widens this performance gap, achieving 22.10 dB PSNR and 0.80 SSIM compared to Photo-SLAM's 19.68 dB and 0.75 SSIM. This pattern of superior performance extends to the TUM dataset, where MGSO consistently maintains higher PSNR and SSIM scores. The reconstruction results for EuRoC MAV and TUM-RGBD are provided in Table 4.6 and Table 4.7.

Table 4.6: Reconstruction Results on EuRoC © 2025 IEEE

Method	**metric**	**MH**	**V1**	**V2**	**Avg.**	**Mem.**
Photo-SLAM	PSNR[dB] ↑	18.60	18.30	17.94	18.28	111.8
	SSIM ↑	0.65	0.73	0.65	0.68	
	LPIPS↓	0.39	0.44	0.53	0.46	
MGSO	PSNR[dB] ↑	**20.75**	**20.26**	**20.31**	**20.44**	**8.3**
	SSIM ↑	**0.72**	**0.79**	**0.75**	**0.76**	
	LPIPS↓	**0.36**	**0.39**	**0.39**	**0.38**	

Mem. is average map size in Mb

Table 4.7: Reconstruction Results on TUM's © 2025 IEEE

Method	**metric**	**fr1**	**fr2**	**fr3**	**Avg.**
Photo-SLAM	PSNR[dB] ↑	18.01	16.93	17.11	17.35
	SSIM ↑	0.65	0.60	0.62	0.63
	LPIPS↓	**0.41**	0.41	0.42	0.41
MGSO	PSNR[dB] ↑	**18.07**	**24.10**	**21.61**	**21.26**
	SSIM ↑	**0.66**	**0.80**	**0.75**	**0.74**
	LPIPS↓	0.45	**0.33**	**0.38**	**0.39**

The performance advantages shown in the tables stem from MGSO's efficient point

cloud generation strategy. By producing well-structured, dense point clouds during initialization, MGSO reduces the need for extensive post-processing operations such as cloning and pruning. This approach yields two key benefits: maps that are half the size of Photo-SLAM's and faster convergence with fewer artifacts. Qualitative observations reveal MGSO's superior handling of challenging scene elements, including improved flat surface rendering, reduced floating artifacts, and better preservation of both edges and fine details, which can be observed in Figure 4.7 and Figure 4.8. Additionally, MGSO's efficient initialization reduces reliance on extensive 3DGS training, enabling weaker mobile hardware to still achieve high-quality reconstruction.

Resource Efficiency and Real-Time Performance

MGSO demonstrates efficiency in both memory utilization and real-time processing. On the EuRoC dataset, MGSO operates with just 8.32 MB of memory, which is dramatically lower than Photo-SLAM's 109.73 MB usage. This efficiency extends to the TUM dataset, where MGSO requires only 2.85 MB compared to Photo-SLAM's 17 MB.

This superior memory efficiency stems from MGSO's structured point cloud generation, which minimizes redundant elements and produces compact maps without sacrificing reconstruction quality. In contrast, Photo-SLAM's approach necessitates larger maps and additional refinement steps. MGSO uniquely achieves this balance of real-time performance and compact map size, distinguishing it as the only system capable of simultaneous high-speed processing and efficient memory usage. A full comparison of MGSO's properties on the Replica dataset against other dense SLAM systems is shown in Table 4.8 and Figure 4.9.

4.6.2 Ablations

Robustness analysis investigated the system's sensitivity to cloning and pruning frequencies. The results demonstrate that higher cloning and pruning rates not only fail to improve reconstruction quality but also lower map compactness. In fact, when cloning and pruning rates exceed certain thresholds, the quality of the reconstruction decreases. The observed robustness of MGSO to changes in pruning and cloning settings indicates that MGSO generates spatially accurate point clouds. These point clouds effectively capture scene geometry, ensuring that MGSO does not require significant adjustments from cloning or pruning. The results of this ablation are shown in Table 4.9.



(a) Renders flat surfaces better

(b) Less floaters and artifacts

Figure 4.7: Comparison of difficult novel view renders between MGSO (top) and Photo-SLAM (bottom). Captions describe how MGSO performs better. Part 1. © 2025 IEEE



(a) Better edges on difficult scene

(b) Less floaters and artifacts

Figure 4.8: Comparison of difficult novel view renders between MGSO (top) and Photo-SLAM (bottom). Captions describe how MGSO performs better. Part 2. © 2025 IEEE

Table 4.8: Replica Aggregated Results © 2025 IEEE

Method	**PNSR[dB]**	**Map Size**	**FPS**	**GPU Usage**
GlORIE-SLAM (GlS)	31.04	114 Mb	0.23	15.22
Mono-GS (MGS)	31.22	6.8 Mb	0.32	14.62
Splat-SLAM (SpS)	**36.45**	6.8 Mb	1.24	17.57
IG-SLAM (IGS)	36.21	14.8 Mb	9.94	16.20
Photo-SLAM (PhS)	30.46	22.5 Mb	**>30***	**3.62**
MGSO (MGSO)	31.41	**4.3 Mb**	**30***	7.98

*System processed data as fast as inputted video stream

Table 4.9: Cloning and Pruning Interval Ablation © 2025 IEEE

Scene	**metric**	**1024**	**512**	**256**	**128**	**64**	**32**
o0	PSNR[dB]↑	37.06	37.10	37.02	37.15	37.40	36.11
	Memory(Mb)	6.4	7.2	9.0	13.0	21.4	38.1
r0	PSNR[dB]↑	28.84	28.73	28.86	28.76	28.49	27.73
	Memory(Mb)	4.5	5.3	7.7	12.4	22.2	38.5

Ablation Evaluations done on training images instead of test images
Values on top row indicate periods between cloning and pruning processes

Table 4.10: Reduced Input Points Ablation © 2025 IEEE

Dataset	**metric**	**o0**	**o1**	**o2**	**r0**	**r1**
Base	PSNR[dB]↑	37.06	38.37	29.81	28.84	30.68
	Memory(Mb)	6.4	4.2	6.0	4.5	5.3
Halved	PSNR[dB]↑	33.48	33.93	27.87	27.30	29.13
	Memory(Mb)	2.0	1.7	2.2	2.1	2.0

Ablation evaluations done on training images instead of test images

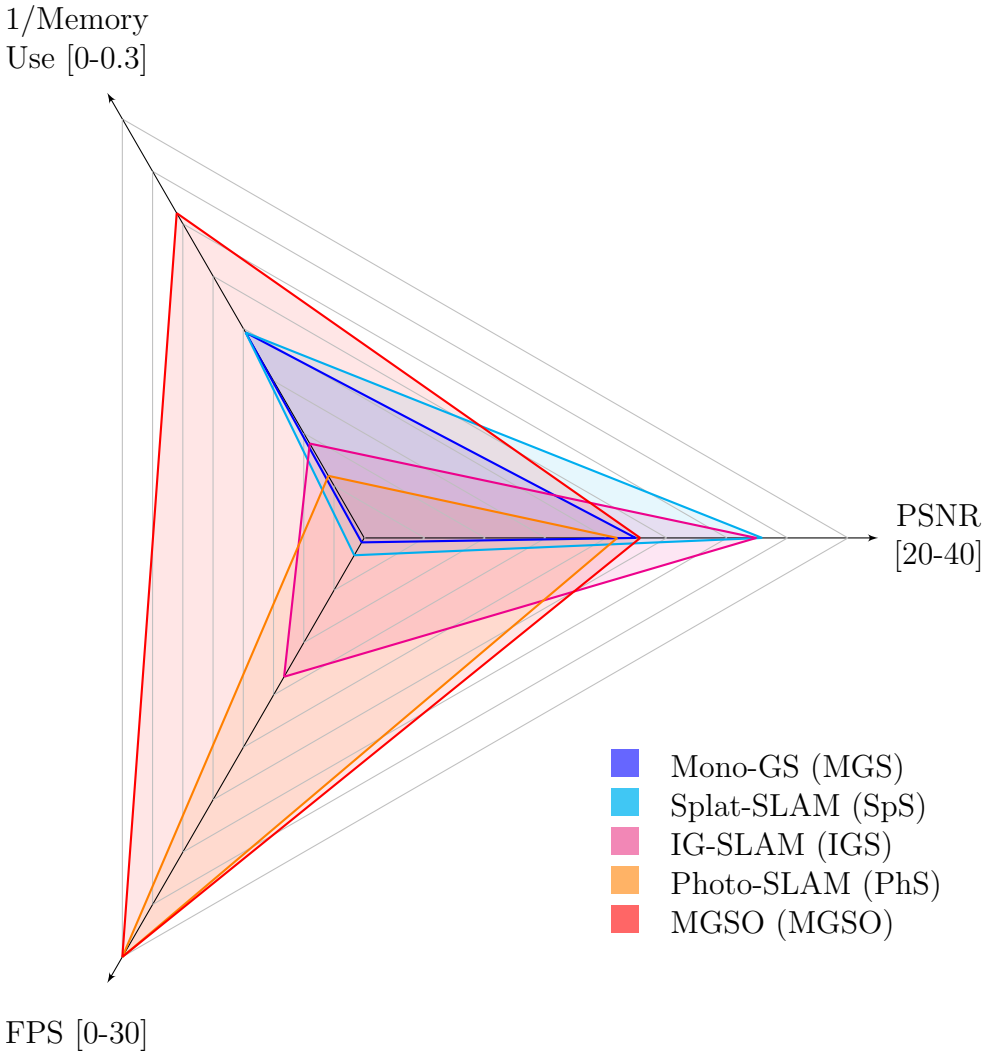


Figure 4.9: This radar plot visually compares the performance of various dense SLAM systems on the Replica dataset. In this plot, being closer to the outer edge generally indicates better performance. This radar plot shows performance of the listed dense SLAM systems on three metrics: rendering performance (PSNR), memory compactness (Memory Use), and speed (FPS). Peak Signal-to-Noise Ratio (PSNR) is measured in decibels (dB), while memory usage is recorded in megabytes (MB). Notably, MGSO demonstrates a well-balanced combination of all three properties compared to other systems.

To validate the importance of point cloud density, ablation studies were conducted using two density reduction methods: halving tracked points and excluding untracked points from the SLAM system. Sparsification led to significant degradation in reconstruction quality, confirming that the input point cloud density and structure is crucial for optimal 3D Gaussian rendering. Results for this ablation are shown in Table 4.10

4.6.3 Insights

MGSO's contributions highlight two key insights about dense SLAM. First, 3DGS exhibits strong synergy with traditional SLAM methods. MGSO leverages high-quality initializations from SLAM to enhance 3DGS reconstruction quality and speed. This is enabled by the inherent compatibility between 3DGS and the point cloud maps used in traditional SLAM systems, which allows seamless adaptation of SLAM outputs for 3DGS integration. Second, photometric methods are critical for dense reconstruction. By maximizing information extraction from images, MGSO outperform alternatives like Photo-SLAM. MGSO's photometric SLAM system is able to feed richer data to 3DGS, significantly improving performance and speed. As reconstruction systems grow more complex and intelligent, the importance of photometric methods will increase because of the need for greater information extraction.

4.7 Extending MGSO

Although MGSO has demonstrated the successful integration of 3DGS with direct SLAM, it should be viewed as a first step toward more sophisticated systems that combine direct SLAM and 3DGS. The natural evolution of this work points toward a fully coupled 3DGS SLAM system, where the mapping and tracking components inform and improve each other.

The connection between 3DGS SLAM and direct SLAM methods extends beyond their shared use of photometric loss. The many years of research on direct SLAM have produced valuable insights and techniques that can be adapted for 3DGS SLAM systems. Particularly relevant are the semi-direct tracking strategies, which suggest an efficient approach to 3DGS SLAM optimization. Similarly to semi-dense direct SLAM systems, not every pixel in the 3DGS representation needs to be perfectly rendered for effective tracking. A system could first generate a semi-dense 3DGS map for tracking purposes while allowing the full mapping representation to slowly fill in holes in the representation. This approach

would enable the 3DGS SLAM system to perform rapid tracking on the already-rendered portions of the scene, potentially offering significant speed improvements over existing fully coupled 3DGS systems. Furthermore, established direct SLAM systems like DSO or SVO could serve as initializers for fully coupled 3DGS SLAM systems, as MGSO has already shown.

Given these synergies between direct SLAM and 3DGS, as shown by the promising results from MGSO, research should continue to bridge these approaches to build the next generation of real-time 3D reconstruction and SLAM systems. By combining traditional SLAM principles with newer radiance field technologies, it is possible to create more efficient, accurate, and practical SLAM systems.

Chapter 5

Conclusion

5.1 Summary and Contributions

This thesis demonstrates the synergy between traditional direct VSLAM techniques and emerging radiance field approaches. Through experimentation in combining newer 3D reconstruction methods with established SLAM techniques, I developed a novel dense SLAM system that synergistically integrates these approaches. By integrating DSO's strengths with 3DGS reconstruction, MGSO achieves enhanced performance.

Direct SLAM methods, particularly DSO, have a long history of development, with numerous extensions and improvements built upon the foundations set by the earliest Direct SLAM systems. This development has yielded significant breakthroughs in camera tracking precision and computational efficiency for the broader SLAM landscape. However, as SLAM rapidly evolves toward incorporating neural radiance fields, implicit representations, and deep learning architectures, the long-term relevance of traditional direct methods remains uncertain. My research demonstrates a possible approach on how established direct SLAM approaches integrate with emerging technologies to create hybrid systems that leverage their synergies and mitigate their respective limitations.

My research began with exploratory experiments combining direct SLAM with NeRFs. While this initial attempt did not yield a successful implementation, it provided crucial insights that guided the development of MGSO. The fundamental revelation from these early experiments was that the two approaches should not function as isolated systems but rather required thoughtful integration. The principle that direct SLAM and radiance field methods must be interconnected rather than operating independently became the guiding framework for MGSO's development.

MGSO is a novel dense VSLAM system designed for high-quality, real-time 3D reconstruction. The main contribution of MGSO is a novel architecture for dense SLAM that leverages the complementary strengths of photometric SLAM and 3DGS. MGSO's decoupled architecture uses photometric SLAM to effectively initialize a parallel 3DGS module. This is possible because unlike conventional feature-based SLAM methods that produce sparse point clouds, MGSO is engineered to track a pixels, thus generating a richer and more structured point cloud. This high-quality initialization accelerates 3DGS optimization, steering it toward more compact reconstructions with fewer artifacts and redundancies. Additionally, MGSO operates in real time using only a monocular camera. Consequently, experimental validation shows MGSO achieves an unmatched combination of speed, map quality, and memory efficiency compared to other dense SLAM approaches.

The success of MGSO leads to several important insights for future dense SLAM systems. The first main insight is that 3DGS is quite compatible with traditional SLAM. Traditional SLAM systems typically work with point cloud based-map representations. The Gaussian splats cloud used in 3DGS can be interperated as point cloud with points with additional parameters to represent the volumetric characteristics. In fact, splats with a volume of zero become points. The similarity of splat to point cloud leads to high compatibility and synergy between 3DGS and traditional SLAM.

The success of MGSO yields several insights for future dense SLAM systems. The first insight is the compatibility between 3DGS and traditional SLAM frameworks. While conventional SLAM systems typically operate with point cloud representations, the Gaussian splat formulation employed in 3DGS can be conceptualized as an enhanced point cloud where each element carries additional parameters that characterize volumetric properties. Gaussian splats with zero volume effectively reduce to traditional points. This fundamental similarity between splats and point clouds facilitates integration and creates synergies between 3DGS and traditional SLAM techniques, opening opportunities for hybrid mapping systems that leverage the strengths of both approaches. The second key insight is the effectiveness of photometric techniques for dense SLAM, as they maximize information extraction from each image frame. Modern reconstruction approaches like radiance fields and neural methods operate directly on image pixels rather than relying on trackable features, recognizing that dense reconstruction requires extensive image data. This principle enables MGSO's success: by extracting significantly more information, the photometric SLAM module generates high-quality dense point clouds that initialize the reconstructor, enhancing both the speed and accuracy of map creation. Both established and emerging photometric techniques will remain crucial for future development of dense SLAM.

5.2 Future Research

MGSO serves as a proof of concept for combining the rich history of direct SLAM techniques with modern radiance field methods. The diverse array of established SLAM approaches, each with its unique strengths, suggests numerous promising avenues for future integration. This suggests a promising path forward: rather than treating classical and contemporary methods as competing approaches, we can integrate their complementary strengths. MGSO represents just one approach to combining newer reconstruction techniques with established SLAM methodologies; more effective integration strategies likely exist and merit further investigation.

A promising direction for future research lies in integrating MGSO's concepts into a fully-coupled 3DGS SLAM system. The ideal dense SLAM framework should have fully-coupled tracking and mapping components, as it is easy to see that these processes inherently influence each other. Current fully-coupled 3DGS SLAM implementations typically rely on older and simpler photometric tracking techniques. Incorporating advancements from sophisticated photometric SLAM systems such as DSO and SVO could significantly enhance both speed and performance of existing coupled 3DGS frameworks. These techniques may help overcome a critical bottleneck: the requirement for high-quality map renderings before tracking can proceed. By adopting the sparse pixel utilization approach employed in DSO and SVO, where tracking depends on only a subset of image pixels, the system would need fewer well-mapped areas before effective tracking could commence. This suggests the possibility of a fully-coupled 3DGS SLAM system that incorporates DSO or SVO photometric tracking principles for fast tracking and map reconstruction, and is initialized by MGSO which supplies initial pixel depths and poses before high-quality renderings are available.

At the time of writing, the field of radiance field reconstruction remains in its early stages and attracts substantial research attention. The original 3DGS framework continues to evolve rapidly, with numerous extensions emerging that enhance its speed, reduce memory requirements, and improve overall performance. The uncoupled architecture of MGSO presents a significant advantage in this research landscape because advancements to the core 3DGS algorithm can be easily integrated into MGSO. This architectural flexibility ensures that MGSO can improve performance as the underlying technologies mature, potentially extending its utility. Furthermore, this adaptability positions MGSO as an ideal platform for testing various 3DGS improvements in SLAM contexts.

Looking ahead, this research contributes to the broader vision of developing real-time SLAM systems that provide robots with a comprehensive understanding of the environ-

ment. MGSO advances our understanding toward this goal, and it also illuminates promising directions for future research. By continuing to bridge the gap between established SLAM methodologies and emerging techniques, we are closer to creating perception systems that give robots the comprehensive spatial awareness they need.

References

[1] *GO-SLAM: Global Optimization for Consistent 3D Instant Reconstruction*, October 2023.

[2] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. In *2009 IEEE 12th International Conference on Computer Vision*, pages 72–79, 2009.

[3] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam): part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[4] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *ICCV*, 2021.

[5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023.

[6] P. Bergmann, R. Wang, and D. Cremers. Online photometric calibration of auto exposure video for realtime visual odometry and slam. *IEEE Robotics and Automation Letters (RA-L)*, 3:627–634, April 2018.

[7] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[8] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[9] Kin-Chung Chan, Jun Xiao, Hana Lebeta Goshu, and Kin-Man Lam. Point cloud densification for 3d gaussian splatting from sparse input views. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 896–904, New York, NY, USA, 2024. Association for Computing Machinery.

[10] Kin-Chung Chan, Jun Xiao, Hana Lebeta Goshu, and Kin man Lam. Point cloud densification for 3d gaussian splatting from sparse input views. In *ACM Multimedia 2024*, 2024.

[11] Chi-Ming Chung, Yang-Che Tseng, Ya-Ching Hsu, Xiang-Qian Shi, Yun-Hung Hua, Jia-Fong Yeh, Wen-Chin Chen, Yi-Ting Chen, and Winston H. Hsu. Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9400–9406, 2023.

[12] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[13] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006.

[14] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *IEEE International Conference on Computer Vision (ICCV)*, Sydney, Australia, December 2013.

[15] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2018.

[16] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 834–849, Cham, 2014. Springer International Publishing.

[17] Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, Zhangyang Wang, and Yue Wang. Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds, 2024.

[18] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, 2014.

[19] X. Gao, R. Wang, N. Demmel, and D. Cremers. Ldso: Direct sparse odometry with loop closure. In *International Conference on Intelligent Robots and Systems (IROS)*, October 2018.

[20] Arren Glover, William Maddern, Michael Warren, Stephanie Reid, Michael Milford, and Gordon Wyeth. Openfabmap: An open source toolbox for appearance-based loop closure detection. In *2012 IEEE International Conference on Robotics and Automation*, pages 4730–4735, 2012.

[21] Stuart I. Granshaw. Laussedat bicentenary: origins of photogrammetry. *The Photogrammetric Record*, 34(166):128–147, 2019.

[22] Seongbo Ha, Jiung Yeon, and Hyeonwoo Yu. Rgbd gs-icp slam. *ArXiv*, abs/2403.12550, 2024.

[23] Seongbo Ha, Jiung Yeon, and Hyeonwoo Yu. Rgbd gs-icp slam, 2024.

[24] Javier Hidalgo-Carrió, Guillermo Gallego, and Davide Scaramuzza. Event-aided direct sparse odometry. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5771–5780, 2022.

[25] Jiarui Hu, Xianhao Chen, Boyin Feng, Guanglin Li, Liangjing Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Cg-slam: Efficient dense rgb-d slam in a consistent uncertainty-aware 3d gaussian field. *arXiv preprint arXiv:2403.16095*, 2024.

[26] Yan Song Hu. Integrating inertial data to a hybrid direct-indirect visual slam system. *Journal of Computational Vision and Imaging Systems*, 9(1):62–64, Apr. 2024.

[27] Yan Song Hu, Nicolas Abboud, Muhammad Qasim Ali, Adam Srebrnjak Yang, Imad Elhajj, Daniel Asmar, Yuhao Chen, and John S. Zelek. Mgso: Monocular real-time photometric slam with efficient 3d gaussian splatting, 2024.

[28] Yan Song Hu, Dayou Mao, Yuhao Chen, and John Zelek. Towards real-time gaussian splatting: Accelerating 3dgs through photometric slam, 2024.

[29] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024.

[30] Huajian Huang, Longwei Li, Cheng Hui, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and rgb-d cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[31] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011.

[32] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat track & map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21357–21366, 2024.

[33] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

[34] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2100–2106, 2013.

[35] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Jeff Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo, 2024.

[36] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. pages 225–234, 12 2007.

[37] S. H. Lee and Javier Civera. Loosely-coupled semi-direct monocular slam. *IEEE Robotics and Automation Letters*, 4:399–406, 2018.

[38] Huai-Jen Liang, Nitin J. Sanket, Cornelia Fermüller, and Yiannis Aloimonos. Salientdso: Bringing attention to direct sparse odometry. *IEEE Transactions on Automation Science and Engineering*, 16(4):1619–1626, 2019.

[39] Hansol Lim, Hanbeom Chang, Jongseong Brad Choi, and Chul Min Yeum. Lidar-3dgs: Lidar reinforced 3d gaussian splatting for multimodal radiance field rendering, 2024.

[40] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[41] Jiahao Lu, Yifan Zhang, Qiuhong Shen, Xinchao Wang, and Shuicheng Yan. Poison-splat: Computation cost attack on 3d gaussian splatting, 2024.

[42] Dongting Luo, Yan Zhuang, and Sen Wang. Hybrid sparse monocular visual odometry with online photometric calibration. *The International Journal of Robotics Research*, 41(11-12):993–1021, 2022.

[43] Mallick and Goel, Bernhard Kerbl, Francisco Vicente Carrasco, Markus Steinberger, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, 2024.

[44] H. Matsuki, L. von Stumberg, V. Usenko, J. Stueckler, and D. Cremers. Omni-directional dso: Direct sparse odometry with fisheye cameras. *IEEE Robotics and Automation Letters & Int. Conference on Intelligent Robots and Systems (IROS)*, 2018.

[45] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[46] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[47] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022.

[48] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[49] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. Dtam: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327, 2011.

[50] Janusch Patas. Gaussian splatting cuda. https://github.com/MrNeRF/gaussian-splatting-cuda, 2023.

[51] Zhexi Peng, Tianjia Shao, Yong Liu, Jingke Zhou, Yin Yang, Jingdong Wang, and Kun Zhou. Rtg-slam: Real-time 3d reconstruction at scale using gaussian splatting. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

[52] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[53] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696, 2020.

[54] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[55] Erik Sandström, Yue Li, Luc Van Gool, and Martin R. Oswald. Point-slam: Dense neural point cloud-based slam. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[56] Erik Sandström, Keisuke Tateno, Michael Oechsle, Michael Niemeyer, Luc Van Gool, Martin R. Oswald, and Federico Tombari. Splat-slam: Globally optimized rgb-only slam with 3d gaussians, 2024.

[57] F. Aykut Sarikamis and A. Aydin Alatan. Ig-slam: Instant gaussian slam, 2024.

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

[59] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[60] D. Schubert, N. Demmel, V. Usenko, J. Stueckler, and D. Cremers. Direct sparse odometry with rolling shutter. In *European Conference on Computer Vision (ECCV)*, September 2018.

[61] D. Schubert, N. Demmel, L. von Stumberg, V. Usenko, and D. Cremers. Rolling-shutter modelling for visual-inertial odometry. In *International Conference on Intelligent Robots and Systems (IROS)*, November 2019.

[62] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 134–144, 2019.

[63] Simon Seibt, Thomas Vincent Siu-Lung Chang, Bartosz von Rymon Lipinski, and Marc Erich Latoschik. Dense 3D Gaussian Splatting Initialization for Sparse Image Data. In Lingjie Liu and Melinos Averkiou, editors, *Eurographics 2024 - Posters*. The Eurographics Association, 2024.

[64] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[65] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[66] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew Davison. iMAP: Implicit mapping and positioning in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[67] Zachary Teed and Jia Deng. DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. *Advances in neural information processing systems*, 2021.

[68] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[69] L. von Stumberg and D. Cremers. DM-VIO: Delayed marginalization visual-inertial odometry. *IEEE Robotics and Automation Letters (RA-L) & International Conference on Robotics and Automation (ICRA)*, 7(2):1408–1415, 2022.

[70] L. von Stumberg, V. Usenko, and D. Cremers. Direct sparse visual-inertial odometry using dynamic marginalization. In *International Conference on Robotics and Automation (ICRA)*, May 2018.

[71] R. Wang, M. Schwörer, and D. Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *International Conference on Computer Vision (ICCV)*, Venice, Italy, October 2017.

[72] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy, 2024.

[73] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[74] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016.

[75] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *CVPR*, 2024.

[76] N. Yang, L. von Stumberg, R. Wang, and D. Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[77] N. Yang, R. Wang, J. Stueckler, and D. Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *European Conference on Computer Vision (ECCV)*, September 2018.

[78] Wenzhe Yang, Yan Zhuang, Senior Member, Dongting Luo, Wei Wang, and Hong Zhang. Vi-hso: Hybrid sparse monocular visual-inertial odometry. *IEEE ROBOTICS AND AUTOMATION LETTERS*, 8, 2023. High Performance SLAM system based on DSO.

[79] Xingrui Yang, Hai Li, Hongjia Zhai, Yuhang Ming, Yuqian Liu, and Guofeng Zhang. Vox-fusion: Dense tracking and mapping with voxel-based neural implicit representation. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 499–507, 2022.

[80] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details for 3d gaussian splatting, 2024.

[81] Georges Younes, Douaa Khalil, John Zelek, and Daniel Asmar. H-slam: Hybrid direct-indirect visual slam, 2023.

[82] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[83] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R. Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting, 2023.

[84] Ganlin Zhang, Erik Sandström, Youmin Zhang, Manthan Patel, Luc Van Gool, and Martin R Oswald. Glorie-slam: Globally optimized rgb-only implicit encoding point cloud slam. *arXiv preprint arXiv:2403.19549*, 2024.

[85] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[86] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. In *International Conference on 3D Vision (3DV)*, March 2024.

[87] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[88] Jon Zubizarreta, Iker Aguinaga, and Jose Maria Martinez Montiel. Direct sparse mapping. *IEEE Transactions on Robotics*, 36(4):1363–1370, 2020.

APPENDICES

Appendix A

Additional Information

A.1 A Brief Clarification on Visual Odometry versus SLAM

Referenced in subsection 1.2.2.

Visual navigation systems in research commonly fall into two categories: visual odometry and SLAM systems. Since researchers sometimes apply these terms inconsistently, this section establishes the precise definitions and conventions that will be used throughout this thesis. Historically, there has been a distinction between visual odometry and SLAM; however, the differences between them have always been blurry. Nonvisual odometry systems, such as wheel encoders, were able to estimate positional changes without generating any map. In contrast, visual odometry systems must construct at least a local 3D map representation to enable any localization.

Traditionally, VSLAM has been differentiated from visual odometry by the ability to leverage long-term information from systems such as loop closure. For a system to be classified as SLAM, it must maintain a globally consistent map that coherently integrates observations, rather than combining multiple local maps. This distinction has been challenged by the emergence of dense volumetric SLAM systems such as NICE-SLAM[87] and MonoGS[45] that lack global map consistency, but self-identify as SLAM systems due to their ability to still output a useful map of the scene. Because this thesis is focused on dense SLAM, it will use the more recent convention: **a SLAM system is any online system capable of localization and map creation, regardless of its specific long-term information processing strategies**.

VSLAM System		
Name	Publish Date	University
SVO	2014	University of Zurich
LSD-SLAM	2014	Technical University Munich
ORB-SLAM	2015	University of Zaragoza
VINS-Mono	2017	Hong Kong University of Science and Technology
Kimera	2019	Massachusetts Institute of Technology
Droid-SLAM	2021	Princeton University

Table A.1: Table of major modern SLAM systems. At the time of this writing, all of the shown systems are still being used as the foundation for VSLAM research

A.2 Important Modern VSLAM Systems

Referenced in section 2.2.

Table A.1 presents an overview of the pivotal SLAM systems that have shaped the field. Each of these foundational approaches has generated entire families of SLAM implementations.

SVO SVO by Forester, Pizzoli, and Scaramuzza [18] is a semi-direct monocular VSLAM system. SVO is the first fast, robust, and effective system to use photometric error instead of relying only on feature correspondences. It's significance is further expanded upon in Section 2.3.

LSD-SLAM LSD-SLAM, developed by Engel, Thomas, and Daniel [16], is a contemporary VSLAM system for SVO that also relies on photometric loss. Unlike SVO, it avoids feature detection entirely, using purely photometric methods. This system forms the foundation of a lineage of photometric SLAM frameworks that results in the work presented in the thesis. The significance of this SLAM system to this thesis cannot be understated, and more detail on this system is provided in Section 2.3.

ORB-SLAM ORB-SLAM by Mur-Artal et al. [48] is a groundbreaking VSLAM system that revolutionized feature-based SLAM. By centering the system around the ORB feature descriptor [54], it achieves unprecedented improvements in speed, precision, and robustness over its predecessors. Not only does ORB perform better than the older SIFT detector, it

also allows ORB-SLAM to share the same features for tracking, mapping, and loop closure, which significantly improves its efficiency.

VINS-Mono VINS-Mono by Qin, Li, and Shen [52] is an influential development in combining Inertial Measurement Units (IMU) with monocular cameras in SLAM. It created a tightly coupled, nonlinear optimization-based method to fuse pre-integrated IMU measurements and feature observations for effective visual-inertial SLAM.

Kimera Kimera, developed by Rosinol, Abate, Chang, and Carlone [53], represents a significant advance in metric-semantic SLAM. While traditional SLAM systems mainly focused on localization accuracy rather than map utility, the Kimera framework integrates multiple components to generate a semantically annotated 3D mesh representation of the environment.

Droid-SLAM Droid-SLAM is a deep learning-based SLAM system by Teed and Deng [67] that operates through iterative calculations of pose and pixel-wise depth using a dense bundle adjustment layer. By incorporating deep learning within the classical SLAM pipeline, Droid-SLAM demonstrates superior accuracy and robustness compared to traditional approaches, establishing itself as one of the leading deep learning-based SLAM frameworks.

A.3 Relevant Extensions of DSO

Referenced in section 2.3.3.

Several important extensions of DSO have been developed in recent research. This section reviews the key DSO extensions that are important or relevant to the current thesis work.

Additional Visual Data A natural evolution of DSO is the integration of additional visual information sources. Following DSO's initial release, researchers at the Technical University of Munich's Computer Vision Group expanded the framework's possible input sensors. Stereo DSO [71] introduced a methodology for integrating static stereo constraints into the DSO framework that effectively addresses the monocular scale ambiguity issue. Based on this foundation, deep virtual stereo odometer (DVSO) [77][76] leveraged deep

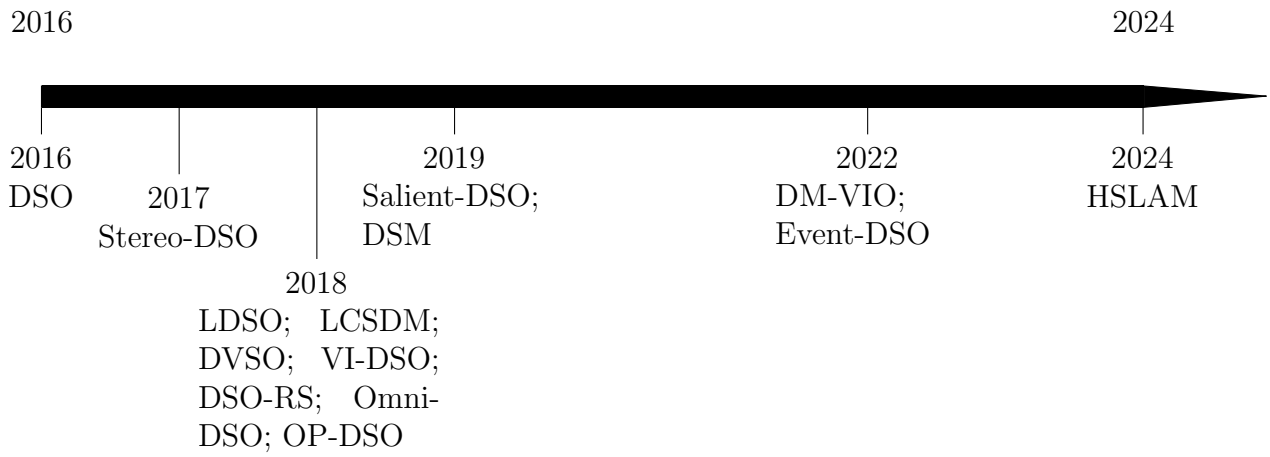


Figure A.1: Timeline of extensions for DSO

learning-based monocular depth estimation to further improve performance. The evolution of DSO extended beyond conventional cameras to encompass more specialized imaging systems. Omnidirectional DSO (Omni-DSO)[44] adapted the framework to work with fish-eye cameras, which have a wide field of view, and Event-Aided DSO (Event-DSO) [24] incorporating event cameras, sensors that detect brightness changes at the pixel level.

Adjustments for Camera Characteristics DSO's direct use of pixel intensities makes it particularly sensitive to subtle camera hardware distortions. Common fixed distortions, such as lens effects, vignetting, and gamma distortion, can be addressed by calibration. However, it is possible to eliminate the calibration burden using DSO with Online Photometric Calibration [6]. Another significant challenge arises from rolling shutter effects, which are present in most modern cameras that capture images by scanning line-by-line rather than instantaneously. This scanning process causes slight slanting in the image during motion, which DSO is sensitive to. DSO with rolling shutter correction (DSO-RS) [60][61] addresses this limitation by explicitly modeling and compensating for these distortions.

Integration of IMU Data The integration of acceleration and angular velocity measurements from an Inertial Measurement Unit (IMU) offers a cost-effective solution to enhance the robustness of monocular SLAM systems. Furthermore, by providing independent motion estimates, IMU data help resolve the fundamental scale ambiguity problem

in monocular SLAM. VI-DSO [70] incorporates an IMU by minimizing a combined objective function that considers both photometric error from visual data and motion residuals from the IMU. However, the absolute scale of the environment cannot be immediately determined from the IMU measurements. To address this challenge, VI-DSO implements a dynamic scale estimation process that initially delays full initialization until sufficient motion information is collected. Building on VI-DSO's foundation, DM-VIO [69] introduces a novel dual-graph optimization approach. While maintaining the primary optimization graph for real-time tracking, DM-VIO simultaneously manages a secondary optimization graph where marginalization is postponed. This enables the system to perform comprehensive optimization once an accurate scale estimate becomes available.

Hybrid Systems Hybrid SLAM systems combine the benefits of direct and indirect methods. Although direct methods excel at precise photometric tracking using pixel intensities, these pixel values are not reliable for long-term mapping, since they change significantly under different lighting conditions and viewing angles. To overcome this limitation, hybrid systems incorporate the loop closure capabilities of indirect SLAM by extracting feature points, which remain more consistent over time. These stable features, when paired with place recognition algorithms, enable the system to reliably detect when it returns to a previously visited location, allowing loop closure.

DSO would be extended with loop closure in systems such as DSO with Loop Closure (LDSO) [19] and Loosely-Coupled Semi-Direct Monocular SLAM (LCSDM) [37]. These approaches combine DSO's pixel points with indirect features at corresponding image positions, enabling both precise pixel-based tracking and the feature information needed for loop closure. However, this approach can lead to drift between the global and local maps due to their loose coupling. HSLAM[81] addresses this limitation by jointly optimizing both direct and indirect points, ensuring that the global map remains consistent with the local representation.

Other Systems DSO has inspired several novel architectural approaches in VSLAM. Direct Sparse Mapping (DSM) [88] replaces DSO's sliding-window optimization with a comprehensive photometric bundle adjustment across the entire video sequence. While this enables robust long-term relocalization, it demands significantly more computational resources. SalientDSO [38] improves DSO's sparse point cloud with semantic understanding by integrating machine learning-based visual saliency detection. This semantic awareness guides the system's localization by identifying and prioritizing visually significant features in the environment.

Appendix B

3D Gaussian Splatting Explanation

B.1 3DGS Rasterization and Optimization Process

Referenced in subsection 4.2.1.

3DGS [33] merges NeRF's differentiable volumetric rendering principles with an explicit scene representation using 3D Gaussian primitives. The technique transforms a scene composed of numerous overlapping, translucent Gaussian elements into 2D images through depth-sorted camera plane projection. The differentiable nature of this rasterization process enables optimization of each Gaussian's properties, which include position, shape, appearance, and opacity, via gradient descent, allowing the model to learn accurate scene representations.

3DGS renders scenes by projecting and blending volumetric Gaussian primitives in 3D space. The system employs depth-aware alpha blending for proper transparency handling, while utilizing an efficient tile-based rasterization approach that divides the image plane into chunks for optimized parallel processing.

The tile-based rasterization method used is described below.

1. Split the image into 16 by 16 pixel tiles.
2. Cull all 3D Gaussian that will not appear on the camera or are too far or close to the camera.
3. For each tile, sort the 3D Gaussians that appear in them by depth.

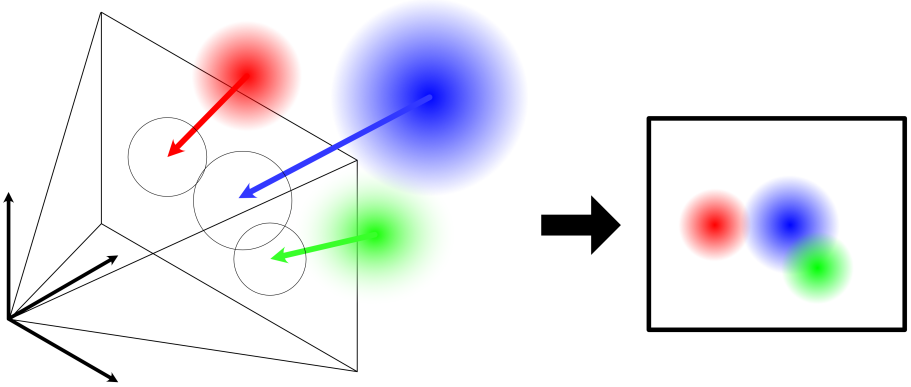


Figure B.1: Diagram showing how 3DGS operates. The Gaussians are rasterized by sorting them by depth and projecting them onto the camera plane. This is the same diagram shown in section 4.2.1 and has been copied here for the convenience of the reader.

4. Each image tile is processed by a dedicated thread that transforms 3D Gaussian primitives into their 2D screen-space projections. These projections are sorted by depth and blended sequentially from front to back.

The parameters of each Gaussian are optimized using gradient descent to minimize the photometric loss L:

$$L = |I_r - I_{gt}|(1 - \lambda) + SSIM(I_r, I_{gt})\lambda \tag{B.1}$$

Where $|I_r - I_{gt}|$ is the difference between the rendered image I_r and ground truth image I_{gt} and $SSIM(I_r, I_{gt})$ is a normalization term using the SSIM. For every pose, a loss is calculated using the equation provided above.

The Gaussian splats are parameterized in a way to allow for differentiation so that gradient descent can be done. Each Gaussian is centered at a point μ has a overall transparency α and color. To model the cloud-like geometry of the Gaussian it is parameterized using this equation:

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

Where Σ represents the covariance of the Gaussian. The covariance defines the Gaussian's size, shape, and orientation. In order to be differentiable, the covariance must be represented using this equation:

$$\Sigma = RSS^TR^T \tag{B.3}$$

Where R represents a rotation matrix and S represents a scaling matrix. This parametrization splits the covariance matrix into a rotation and scaling value, which are optimized separately. Each Gaussian will have it's center (μ), transparency (α), color, rotation (R), and scale (S) values optimized using the losses calculated from the differences between the rendered and ground truth images.

However, the creation and deletion of individual Gaussians are not differentiable and must be handled by a separate parallel process known as adaptive density control. This process relies on heuristics to regulate the density of Gaussians throughout the scene. Beginning with the initial point cloud provided by SfM or SLAM, the adaptive density control process periodically adds or removes Gaussians after a set number of training steps, which is 100 by default.

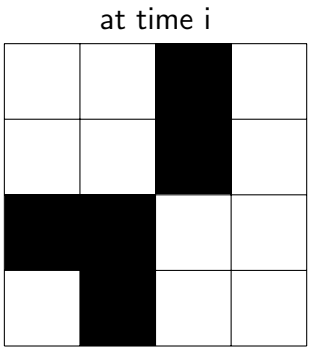
The primary objective of this process is to refine the representation by introducing Gaussians in areas lacking features or appearing empty while eliminating redundant ones. To achieve this, regions experiencing significant losses are enhanced through Gaussian addition, which can be implemented via either Gaussian cloning or splitting. When the Gaussians in the area are small, cloning is applied. Conversely, if particularly large Gaussians with high variance are present, splitting is employed instead. Furthermore, Gaussians with excessively low transparency values are removed for pruned from the representation. While this heuristic approach is effective in practice, it remains an area for potential improvement in the original 3DGS algorithm.

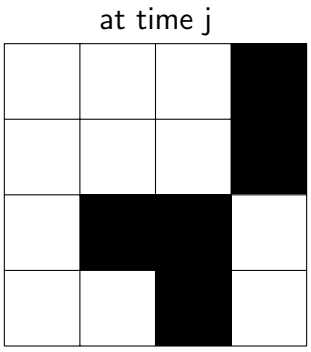
Appendix C

DSO Explained

C.1 Photometric Loss Example

Referenced in subsection 1.2.3 and C.2.2





0	0	1	-1
0	0	1	-1
1	0	-1	0
0	1	-1	0

The photometric loss between the grid between times i and j will be 8. If we assume that the grid at time i is shifted one right, the photometric loss will be 0.

This simplified example uses 2D planar motion and orthographic projection, whereas real scenarios involve 3D motion and perspective projection.

C.2 Overview

The appendix serves as an operational manual for DSO, extending the original paper with detailed implementation details and mathematical derivations of the gradients used

in optimization. It offers step-by-step explanations of how the codebase implements the visual odometry pipeline of DSO. Although the appendix provides valuable supplementary information, it should be used alongside rather than in place of the original paper, as it builds upon rather than repeats the core mathematical foundations, core implementation details, and reasons for design decisions presented in the main text.

C.2.1 Pixel Selection

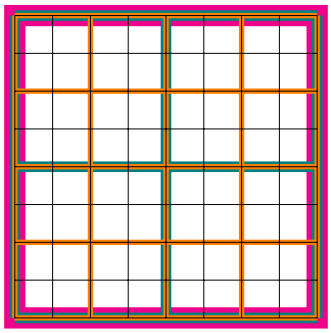


Figure C.1: A diagram of the hierarchical selection grid used by the pixel selector assuming the lowest grid size (potential) is two. The orange is the lowest level grid, the teal is the second level, and the pink is the largest grid. The black grids represent pixels.

The DSO pixel selector is designed to select pixels in areas of high gradient while maintaining good distributions of pixels.

The algorithm operates through a hierarchical grid-based approach. The image is divided into grids at three hierarchical levels. In each hierarchy, the grid size doubles compared to the last level. Within each grid cell, it selects pixels with the strongest gradients. To prevent clustering, pixel selection is excluded from a grid if a pixel has already been selected in any smaller grid cell contained within at a lower hierarchy. The gradient threshold for pixel selection scaled down by a multiplier as the grid size increases, allowing the algorithm to capture both strong local features and subtler gradients in less textured regions.

Gradient thresholding is based on a threshold table generated by dividing the histogram-equalized gradient image into N×N pixel zones. The threshold for each zone is determined by calculating the median gradient value and adding a fixed minimum threshold. After creation, the threshold table is smoothed using a box kernel.

The density of the pixel selection is controlled by the initial block size, which is referred to as the potential in the code.

A recursive process is done to obtain the desired density of selected points. A threshold table is assumed to be made before this operation starts.

1. Select pixels using the algorithm discussed above.
2. Calculate the ratio (quota) between the number of pixels desired versus the number of pixels selected.
3. Calculate the ideal grid size, which is called the potential:

$$p_i = \sqrt{q} * (p_c - 1) - 1$$

   Where p_i is the ideal potential, p_c is the current potential, and p_c is the quota. The smallest ideal potential is one because it is not possible to be smaller than one pixel.
4. This process is repeated until the number of points is within 1.25 to 0.25 of the target density.
   (a) If the ideal potential is greater than the current potential (too little points), recurse with potential minus one
   (b) If the ideal potential is less than the current potential (too many points), recurse with potential plus one

C.2.2 Tracking

The tracking process begins with each new input frame. First, the system verifies and, if necessary, updates the reference frame, which is typically the latest keyframe. In photometric tracking, pixels from the reference frame are projected onto the current frame to calculate loss and gradient values, as demonstrated in Figure C.1.

Rather than tracking map points directly, DSO projects them onto the reference frame to create a reference pixel map. This reference map is then compared to the incoming frames for tracking. The reference pixel map is generated through the following process:

1. For all active keyframes, project all the center pixel of all pixel points that would be valid in the reference frame into the reference keyframe. This will create a reference pixel map with known pixel depths on the first pyramid level.

2. Create pixel maps for every other coarser pyramid level by doing a 2 by 2 box kernel convolution on the finer pyramid level.

3. For every pyramid level, interpolate depth values to empty spaces by averaging the depth values from surrounding pixels.

The reference pixel map is generated across multiple pyramid levels of the image. Tracking begins at the coarsest pyramid level and progressively moves to finer levels, with each coarser level's results initializing the tracking at the next finer level.

After confirming the reference frame, the system generates initial motion estimates for optimization using these heuristics.

- Constant velocity from last position
- Half or double constant velocity from last position
- Zero motion from last position or keyframe
- 27 evenly distributed rotations around a sphere, centered at the previous position

The system performs tracking optimization for each initial guess using multiple iterations of the Gauss-Newton algorithm. The optimization progresses from coarse to fine pyramid levels, maintaining constant pixel depth values while optimizing pose and affine parameters. The result of each coarser level initializes the optimization at the next finer level, gradually improving the photometric precision.

Furthermore, the tracking process outputs statistical variables for mapping and calculates the loss energy per point. A track is considered successful if this energy falls below a specified threshold. The system selects the track with the lowest loss energy out of all of the initial guesses. Upon successful tracking, the system updates to the calculated values and continues. If tracking fails but energy remains reasonable, the system assumes a constant velocity motion. However, if the energy loss is excessive, the system declares complete tracking failure and a full system restart is done.

C.2.3 Mapping

Successfully tracked frames are passed to the mapping process, which runs parallel to tracking. The mapping process serves two purposes: to calculate new depth values for the

pixel points and to optimize the full pose graph for all keyframes and their variables within the window. This comprehensive mapping only occurs when a new keyframe is inserted.

Using the statistics calculated in tracking, the mapping system calculated whether a keyframe is needed using these metric:

- The field of view changes significantly. The field of view change is measured by the mean optical flow.
- If motion causes occlusions or dis-occlusions. This is measured by the mean translation flow.
- If lighting conditions are significantly changed. This is measured by the relative brightness change between two frames.
- If the residual is too high.
- If max time between keyframes is surpassed.

Immature Points

At keyframe creation, DSO establishes a large set of candidate points, named immature points, that can be converted into fully tracked points as needed. This approach ensures that DSO maintains an adequate number of active points without needing to repeatably call the pixel selector. It is important to note that the number of immature points is fixed at keyframe creation and cannot increase afterwards because the immature point selection process occurs only once.

In keyframe creation, the pixel selector is run on the base pyramid level to identify suitable pixels for new immature points. Once the pixels are selected, candidate points are initialized at these locations with the default parameters. Each candidate point is assigned initial depth bounds, with the minimum depth set to zero and the maximum depth set to infinity.

The immature points undergo continuous tracking on each frame input. This tracking serves two purposes: providing an initial estimated depth value once activated and filtering out low-quality candidates. The tracking process is simplified by using known motions from the tracking system, which provide epipolar lines for efficient search operations. The tracking procedure consists of the following steps:

1. First, the system checks if a candidate point lies within valid image boundaries. This boundary check projects the candidate point onto the current frame at both its minimum and maximum depths. For candidate points with invalid depth values (such as during initialization), the projection uses a predetermined fixed projection length. If the projected point falls outside the image boundaries at any depth, it is marked as out-of-bounds and removed from further tracking.

2. When the projected distance between the minimum and maximum depth positions falls below a specified threshold, the system retains the existing depth values and skips the tracking step. This optimization prevents unnecessary tracking operations when the depth uncertainty is already sufficiently small.

3. If the depth uncertainty is not small, the system performs a discrete search along the epipolar line between the projected minimum and maximum depth positions. This search identifies the optimal depth value by locating the point with minimal photometric error along the search line.

4. Using the starting point found in the previous step, a Gauss-Newton optimization for a fixed number of iterations is done to further optimize the point

5. The system updates the depth bounds (minimum and maximum) using results from Gauss-Newton optimization and incorporating an uncertainty term. The uncertainty is derived from the alignment between the pixel gradient and observed image motion because when gradients are poorly aligned with motion, tracking reliability decreases. This alignment-based uncertainty directly influences the range of depth bounds.

During the tracking process, the system maintains a set of statistical metrics that inform decisions about the activation of immature points. These metrics help to evaluate the quality and reliability of each candidate point. Immature point will be rejected under these conditions:

- If the candidate point is projected to out-of-bound at any depth.
- If the change in the Z direction is too large compared to the depth.
- If the pixel derivative is aligned extremely poorly with the movement. When a pixel's gradient is poorly aligned with the direction of motion, gradient-based tracking methods can produce degenerate results.
- If the Gauss Newton optimization returns a value with too high energy.
- If the final depth values are not finite or less than zero.

Non-Keyframe Creation

When no keyframe is needed, the system performs only maintenance tasks. The immature points are traced using the calculated motion of the tacking, and the active points remain unchanged.

Keyframe Creation

When a new keyframe is created, the full optimization process and all supporting operations are performed. This process is described in the following steps.

1. Set the pose and other important variables of the new keyframe
2. Trace the existing immature points
3. Flag frames for marginalization using these heuristics:
    - Always keep the two newest keyframes.
    - Marginalize frames where the number of active pixel points falls below a specified threshold.
    - When the keyframe window reaches capacity, the system marginalizes the frames to maintain an even spatial distribution of viewpoints in the 3D space. This distribution is calculated using a distance-based scoring metric that evaluates spatial coverage.
4. Add the new frame to the keyframe window and the energy function.
5. Iterates through existing active pixel points in previously established keyframes to compute and incorporate new residual terms between these points and the newly created keyframe.
6. To maintain optimal point density, the system activates select candidate points from existing keyframes. Activation is restricted to candidates that maintain sufficient spatial separation from already active points, ensuring even distribution across the keyframe.
7. Perform a comprehensive optimization that jointly refines all active pixel points and keyframes within the current window.

8. Prunes point identified as outliers during the optimization process, removing them from the active pixel point set.
9. Set the new reference frame for tracking
10. Remove poorly conditioned points. These include active pixel points with these characteristics:
    - The point has an invalid depth
    - The point is no longer viewable in any keyframe in the window excluding it's host, which means it no longer has any residuals
    - The point is out of bounds
    - The point is in a frame that is flagged for marginalization
11. Create immature points for the new keyframe.
12. Marginalize all frames flagged for marginalization from the optimization system.

C.2.4 Initialization

The system initialization establishes the initial camera motion by analyzing the photometric changes between two frames. This initialization phase presents unique challenges compared to standard tracking because the depths of the pixel points are completely unknown at this stage. Consequently, techniques must be used to estimate the initial depth values.

When processing the first frame, the pixel selector operates across all pyramid levels to identify suitable points. At each point selected by the pixel selector, the pixel points are initialized with a default depth value of one. The system then identifies two sets of neighboring pixels for each point: the n-nearest neighbors in the same pyramid level and the closest points in adjacent pyramid levels (both above and below). This neighborhood structure enables depth interpolation from neighboring pixels, providing a method to estimate initial depth estimation.

After establishing the first frame, DSO attempts to create a tracking relationship between it and each subsequent frame using the standard photometric error function, enhanced with point interpolation. The tracking process occurs hierarchically across all pyramid levels, initializing with a zero-motion assumption for optimization. Before each optimization iteration, pixels with no depth try to inherit initial depth estimates from their

corresponding pixels in the higher pyramid level if needed. Throughout the tracking process, depth values are progressively refined through neighborhood averaging. The process concludes with an upward depth propagation, interpolating values from lower to higher pyramid levels if needed.

The initialization process continues frame-by-frame, using accumulated information from previous frames until the photometric error falls below a specified threshold. If the system fails to establish a satisfactory track within a maximum time window, it triggers a complete reset. Upon achieving a successful track, the system finalizes initialization by adding the successfully tracked frame to the set of active keyframes. Since initial tracking operates with a higher point density than standard tracking, the system then randomly subsamples the tracked points to achieve the desired point density for normal operation. Once initialization is complete, DSO transitions to its standard tracking and mapping pipeline.

C.3 Mathematics

C.4 Residual Calculation

The full optimization equation between frames i and j is this:

$$E_{\boldsymbol{p}_j} := \sum_{\boldsymbol{p} \in N_{\boldsymbol{p}}} w_{\boldsymbol{p}} \left\| (I_j[\boldsymbol{p'}] - b_j) - \frac{s_j e^{a_j}}{s_i e^{a_i}} (I_i[\boldsymbol{p}] - b_i) \right\|_\gamma \tag{C.1}$$

For each point:

$$E = \left\| (I_j[\boldsymbol{p'}] - b_j) - \frac{s_j e^{a_j}}{s_i e^{a_i}} (I_i[\boldsymbol{p}] - b_i) \right\|_\gamma \tag{C.2}$$

Defintion of variables:

- u and v are the location of the point in pixel coordinates
- a and b are photometric variables. a represents a contrast adjustment. b represents a brightness adjustment.
- s is the exposure time.

- $d_{\boldsymbol{p}}$ is the inverse depth
- $\boldsymbol{R}$ is a rotation matrix
- $\vec{t}$ is a translation vector
- $\boldsymbol{K_c}$ is a camera calibration matrix with variables $\boldsymbol{c} = [f_x, f_y, c_x, c_y]$
- γ represents the Huber norm

$\boldsymbol{p}$ is defined as a 2D coordinate in pixel space. For tracked pixels, it is paired with an inverse depth value $d_{\mathbf{p}}$. The a and b values in the photometric loss equation act as an affine brightness transfer function to correct for photometric changes if exposure times are not known. a and b are pulled to zero with an additional prior if exposure times are fully known.

Where the change between $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ is calculated by:

$$\mathbf{p}' = \Pi_c(\mathbf{R}\Pi_c^{-1}(\mathbf{p}, 1) + d_{\mathbf{p}}\mathbf{t}) \tag{C.3}$$

Transformations for rotation and translation are done separately. Rotations are not affected by inverse depth because rotation is the same for all depth values. The translation is scaled by the inverse depth value. The $\boldsymbol{p}$ is converted to the 3D coordinate value before the transformations using the inverse of the intrinsic matrix $\Pi_c^{-1}(\boldsymbol{p}, 1)$.

$$\Pi_c^{-1}(\boldsymbol{p}, 1) = \begin{bmatrix} 1/f_x & 0 & -c_x/f_x \\ 0 & 1/f_y & -c_y/f_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} (u - c_x)/fx \\ (v - c_y)/fy \\ 1 \end{bmatrix}$$

Therefore

$$\mathbf{R}\Pi_c^{-1}(\mathbf{p}, 1) + d_{\mathbf{p}}\mathbf{t} = \begin{bmatrix} \mathbf{R}_{1,1} & \mathbf{R}_{1,2} & \mathbf{R}_{1,3} \\ \mathbf{R}_{2,1} & \mathbf{R}_{2,2} & \mathbf{R}_{2,3} \\ \mathbf{R}_{3,1} & \mathbf{R}_{3,2} & \mathbf{R}_{3,3} \end{bmatrix} \begin{bmatrix} (u - c_x)/fx \\ (v - c_y)/fy \\ 1 \end{bmatrix} + d_{\mathbf{p}} \begin{bmatrix} \mathbf{t}_x \\ \mathbf{t}_y \\ \mathbf{t}_z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Thus

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{1,1}(u - c_x)/f_x + \mathbf{R}_{1,2}(v - c_y)/f_y + \mathbf{R}_{1,3} + d_{\mathbf{p}}\mathbf{t}_x \\ \mathbf{R}_{2,1}(u - c_x)/f_x + \mathbf{R}_{2,2}(v - c_y)/f_y + \mathbf{R}_{2,3} + d_{\mathbf{p}}\mathbf{t}_y \\ \mathbf{R}_{3,1}(u - c_x)/f_x + \mathbf{R}_{3,2}(v - c_y)/f_y + \mathbf{R}_{3,3} + d_{\mathbf{p}}\mathbf{t}_z \end{bmatrix} = \begin{bmatrix} r_x + d_{\mathbf{p}}\mathbf{t}_x \\ r_y + d_{\mathbf{p}}\mathbf{t}_y \\ r_z + d_{\mathbf{p}}\mathbf{t}_z \end{bmatrix}$$

Where r_x,r_y, and r_z represent the rotation aspect

These values are then transformed back to pixel space using the intrinsic matrix Π_c.

$$\Pi_c(x,y,z) = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} xf_x + zc_x \\ yf_y + zc_y \\ z \end{bmatrix} = \begin{bmatrix} xf_x/z + c_x \\ yf_y/z + c_y \\ 1 \end{bmatrix}$$

$$P_u = xf_x/z + c_x \tag{C.4}$$

$$P_v = yf_y/z + c_y \tag{C.5}$$

With the new inverse depth of the pixel being scaled by z

$$d_{\mathbf{p}'} = d_{\mathbf{p}}/z \tag{C.6}$$

The Huber loss is used for the photometric loss optimization to make the regression more robust. Furthermore, $w_{\mathbf{p}}$ is used as a gradient-dependent weighing to dow-weigh pixels with high gradients.

$$w_{\mathbf{p}} = \frac{c^2}{c^2 + \|\nabla I_j[\mathbf{p}]\|_2^2} \tag{C.7}$$

C.5 Jacobian Derivative Calculations

The optimization is done through iterative non-linear Gauss-Newton.

$$H = J^T W J \quad \text{and} \quad b = -J^T W r \tag{C.8}$$

Where W are the weights, r are the residuals, and J is the Jacobian of r. H and b are the values calculated by the Gauss-Newton optimization.

$$x' \leftarrow \delta + x \quad \text{and} \quad \delta = H^{-1}b \tag{C.9}$$

Here x' is the new state and δ is the increment calculated by Gauss-Newton.

The Jacobian has three parts:

$$J_k = [J_I, J_g, J_p] \tag{C.10}$$

- J_I is the partial derivative of the pixel intensity function I on the location of the pixel point $\boldsymbol{p'}$.
- Where J_g is the partial derivative of $\boldsymbol{p'}$ on the geometric parameters $\vec{t}, \boldsymbol{R}, d_{\boldsymbol{p}}, \boldsymbol{c}$.
- Where J_p is the partial derivative of I over a_i, a_j, b_i, b_j.

The derivative of $[J_I, J_g, J_p]$ are derived here. All derivatives are calculated disregarding the Huber Norm.

For J_I, which consists of the derivatives of the residual over the new position $\boldsymbol{p'} = (P_u, P_v)$:

$$\partial \left((I_j[\boldsymbol{p'}] - b_j) - \frac{s_j e^{a_j}}{s_i e^{a_i}} (I_i[\boldsymbol{p}] - b_i) \right) \Big/ \partial \boldsymbol{p'} \tag{C.11}$$

$$\nabla I_u \frac{\partial P_u}{\partial \boldsymbol{p'}} + \nabla I_v \frac{\partial P_v}{\partial \boldsymbol{p'}} \tag{C.12}$$

Where ∇I is the pixel gradient in the specified direction. ∇I_u is the horizontal pixel gradient and ∇I_v is the vertical pixel gradient.

For J_g, which consists of the derivatives of $\boldsymbol{p'} = (P_u, P_v)$ over geometric parameters $\vec{t}, \boldsymbol{R}, d_{\boldsymbol{p}}, \boldsymbol{c}$:

For $d_{\boldsymbol{p}}$:

$$\frac{\partial P_u}{\partial d_{\boldsymbol{p}}} = \frac{\partial}{\partial d_{\boldsymbol{p}}} \left(f_x \frac{x}{z} + c_x \right) = \frac{\partial}{\partial d_{\boldsymbol{p}}} \left(f_x \frac{r_x + d_{\mathbf{p}} \mathbf{t}_x}{r_z + d_{\mathbf{p}} \mathbf{t}_z} + c_x \right) \tag{C.13}$$

$$\frac{\partial P_u}{\partial d_{\boldsymbol{p}}} = f_x \frac{(r_z + d_{\mathbf{p}} \mathbf{t}_z)\mathbf{t}_x - (r_x + d_{\mathbf{p}} \mathbf{t}_x)\mathbf{t}_z}{(r_z + d_{\mathbf{p}} \mathbf{t}_z)^2} \tag{C.14}$$

$$\frac{\partial P_u}{\partial d_{\boldsymbol{p}}} = \frac{f_x}{r_z + d_{\mathbf{p}} \mathbf{t}_z} \left(\mathbf{t}_x - \mathbf{t}_z \frac{r_x + d_{\mathbf{p}} \mathbf{t}_x}{r_z + d_{\mathbf{p}} \mathbf{t}_z} \right) = \frac{f_x}{z} \left(\mathbf{t}_x - \mathbf{t}_z \frac{x}{z} \right) \tag{C.15}$$

Similarly:

$$\frac{\partial P_v}{\partial d_{\boldsymbol{p}}} = \frac{f_y}{z}\left(\mathbf{t}_y - \mathbf{t}_z\frac{y}{z}\right) \tag{C.16}$$

For the values of the calibration matrix:

For c_x:

$$\frac{\partial P_u}{\partial c_x} = \frac{\partial}{\partial c_x}\left(f_x\frac{x}{z} + c_x\right) \tag{C.17}$$

$$\frac{\partial P_u}{\partial c_x} = \frac{\partial}{\partial c_x}\left(f_x\frac{\mathbf{R}_{1,1}(u-c_x)/f_x + \mathbf{R}_{1,2}(v-c_y)/f_y + \mathbf{R}_{1,3} + d_{\mathbf{p}}\mathbf{t}_x}{\mathbf{R}_{3,1}(u-c_x)/f_x + \mathbf{R}_{3,2}(v-c_y)/f_y + \mathbf{R}_{3,3} + d_{\mathbf{p}}\mathbf{t}_z} + c_x\right) \tag{C.18}$$

$$\frac{\partial P_u}{\partial c_x} = f_x\frac{-\mathbf{R}_{1,1}z/f_x + \mathbf{R}_{3,1}x/f_x}{z^2} + 1 \tag{C.19}$$

$$\frac{\partial P_u}{\partial c_x} = \frac{1}{z}\left(-\mathbf{R}_{1,1} + \frac{\mathbf{R}_{3,1}x}{z}\right) + 1 \tag{C.20}$$

For c_y:

$$\frac{\partial P_u}{\partial c_y} = \frac{\partial}{\partial c_y}\left(f_x\frac{x}{z} + c_x\right) \tag{C.21}$$

$$\frac{\partial P_u}{\partial c_y} = \frac{\partial}{\partial c_y}\left(f_x\frac{\mathbf{R}_{1,1}(u-c_x)/f_x + \mathbf{R}_{1,2}(v-c_y)/f_y + \mathbf{R}_{1,3} + d_{\mathbf{p}}\mathbf{t}_x}{\mathbf{R}_{3,1}(u-c_x)/f_x + \mathbf{R}_{3,2}(v-c_y)/f_y + \mathbf{R}_{3,3} + d_{\mathbf{p}}\mathbf{t}_z} + c_x\right) \tag{C.22}$$

$$\frac{\partial P_u}{\partial c_y} = f_x\frac{-\mathbf{R}_{1,2}z/f_y + \mathbf{R}_{3,2}x/f_y}{z^2} \tag{C.23}$$

$$\frac{\partial P_u}{\partial c_y} = \frac{f_x}{zf_y}\left(-\mathbf{R}_{1,2} + \frac{\mathbf{R}_{3,2}x}{z}\right) \tag{C.24}$$

For f_x:

$$\frac{\partial P_u}{\partial f_x} = \frac{\partial}{\partial f_x}\left(f_x \frac{x}{z} + c_x\right) \tag{C.25}$$

$$\frac{\partial P_u}{\partial f_x} = \frac{\partial}{\partial f_x}\left(f_x \frac{\mathbf{R}_{1,1}(u-c_x)/f_x + \mathbf{R}_{1,2}(v-c_y)/f_y + \mathbf{R}_{1,3} + d_{\mathbf{p}}\mathbf{t}_x}{\mathbf{R}_{3,1}(u-c_x)/f_x + \mathbf{R}_{3,2}(v-c_y)/f_y + \mathbf{R}_{3,3} + d_{\mathbf{p}}\mathbf{t}_z} + c_x\right) \tag{C.26}$$

$$\frac{\partial P_u}{\partial f_x} = f_x \frac{-\mathbf{R}_{1,1} z(u-c_x)/f_x^2 + \mathbf{R}_{3,1} x(u-c_x)/f_x^2}{z^2} + 1\frac{x}{z} \tag{C.27}$$

$$\frac{\partial P_u}{\partial f_x} = \frac{1}{z}\frac{u-c_x}{f_x}\left(-\mathbf{R}_{1,1} + \frac{\mathbf{R}_{3,1}x}{z}\right) + \frac{x}{z} \tag{C.28}$$

For f_y:

$$\frac{\partial P_u}{\partial f_y} = \frac{\partial}{\partial f_y}\left(f_x \frac{x}{z} + c_x\right) \tag{C.29}$$

$$\frac{\partial P_u}{\partial f_y} = \frac{\partial}{\partial f_y}\left(f_x \frac{\mathbf{R}_{1,1}(u-c_x)/f_x + \mathbf{R}_{1,2}(v-c_y)/f_y + \mathbf{R}_{1,3} + d_{\mathbf{p}}\mathbf{t}_x}{\mathbf{R}_{3,1}(u-c_x)/f_x + \mathbf{R}_{3,2}(v-c_y)/f_y + \mathbf{R}_{3,3} + d_{\mathbf{p}}\mathbf{t}_z} + c_x\right) \tag{C.30}$$

$$\frac{\partial P_u}{\partial f_y} = f_x \frac{-\mathbf{R}_{1,2} z(v-c_y)/f_y^2 + \mathbf{R}_{3,2} x(v-c_y)/f_y^2}{z^2} \tag{C.31}$$

$$\frac{\partial P_u}{\partial f_y} = \frac{f_x}{z f_y}\frac{v-c_y}{f_y}\left(-\mathbf{R}_{1,2} + \frac{\mathbf{R}_{3,2}x}{z}\right) \tag{C.32}$$

Similarly:

$$\frac{\partial P_v}{\partial c_x} = \frac{f_y}{z f_x}\left(-\mathbf{R}_{2,1} + \frac{\mathbf{R}_{3,1}y}{z}\right) \tag{C.33}$$

$$\frac{\partial P_v}{\partial c_y} = \frac{1}{z}\left(-\mathbf{R}_{2,2} + \frac{\mathbf{R}_{3,2}y}{z}\right) + 1 \tag{C.34}$$

$$\frac{\partial P_v}{\partial f_x} = \frac{f_y}{z f_x} \frac{v - c_y}{f_y} \frac{u - c_x}{f_x} \left(-\mathbf{R}_{2,1} + \frac{\mathbf{R}_{3,1} y}{z} \right) \tag{C.35}$$

$$\frac{\partial P_v}{\partial f_y} = \frac{1}{z} \frac{v - c_y}{f_y} \left(-\mathbf{R}_{2,2} + \frac{\mathbf{R}_{3,2} y}{z} \right) + \frac{y}{z} \tag{C.36}$$

For the values of the translation values:

For $\mathbf{t}_x$:

$$\frac{\partial P_u}{\partial \mathbf{t}_x} = \frac{\partial}{\partial \mathbf{t}_x} \left(f_x \frac{x}{z} + c_x \right) = \frac{\partial}{\partial \mathbf{t}_x} \left(f_x \frac{r_x + d_\mathbf{p} \mathbf{t}_x}{r_z + d_\mathbf{p} \mathbf{t}_z} + c_x \right) \tag{C.37}$$

$$\frac{\partial P_u}{\partial \mathbf{t}_x} = f_x \frac{d_\mathbf{p}}{z} \tag{C.38}$$

For $\mathbf{t}_y$:

$$\frac{\partial P_u}{\partial \mathbf{t}_y} = \frac{\partial}{\partial \mathbf{t}_y} \left(f_x \frac{x}{z} + c_x \right) = \frac{\partial}{\partial \mathbf{t}_y} \left(f_x \frac{r_x + d_\mathbf{p} \mathbf{t}_x}{r_z + d_\mathbf{p} \mathbf{t}_z} + c_x \right) \tag{C.39}$$

$$\frac{\partial P_u}{\partial \mathbf{t}_y} = 0 \tag{C.40}$$

For $\mathbf{t}_z$:

$$\frac{\partial P_u}{\partial \mathbf{t}_z} = \frac{\partial}{\partial \mathbf{t}_z} \left(f_x \frac{x}{z} + c_x \right) = \frac{\partial}{\partial \mathbf{t}_z} \left(f_x \frac{r_x + d_\mathbf{p} \mathbf{t}_x}{r_z + d_\mathbf{p} \mathbf{t}_z} + c_x \right) \tag{C.41}$$

$$\frac{\partial P_u}{\partial \mathbf{t}_z} = -f_x \frac{d_\mathbf{p} (r_x + d_\mathbf{p} \mathbf{t}_x)}{(r_z + d_\mathbf{p} \mathbf{t}_z)^2} = -f_x \frac{d_\mathbf{p}}{z} \frac{x}{z} \tag{C.42}$$

Similarly:

$$\frac{\partial P_v}{\partial \mathbf{t}_x} = 0 \tag{C.43}$$

$$\frac{\partial P_v}{\partial \mathbf{t}_y} = f_y \frac{d_\mathbf{p}}{z} \tag{C.44}$$

$$\frac{\partial P_v}{\partial \mathbf{t}_z} = -f_y \frac{d_{\mathbf{p}}}{z} \frac{y}{z} \tag{C.45}$$

For the rotation values:

$$\frac{\partial \boldsymbol{p'}}{\partial \mathbf{R}} = \frac{\partial \boldsymbol{p'}}{\partial \mathbf{X}} \frac{\partial \boldsymbol{X}}{\partial \mathbf{R}} \tag{C.46}$$

Where $\mathbf{X}$ are the 3D position values:

$$\mathbf{X} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{C.47}$$

Thus:

$$\frac{\partial \boldsymbol{p'}}{\partial \mathbf{R}} = \partial \begin{bmatrix} x f_x/z + c_x \\ y f_y/z + c_y \end{bmatrix} / \partial \begin{bmatrix} x \\ y \\ z \end{bmatrix} \frac{\partial \boldsymbol{X}}{\partial \mathbf{R}} \tag{C.48}$$

$$\frac{\partial \boldsymbol{p'}}{\partial \mathbf{R}} = \partial \begin{bmatrix} f_x/z & 0 & -x f_x/z^2 \\ 0 & f_y/z & -y f_y/z^2 \end{bmatrix} \frac{\partial \boldsymbol{X}}{\partial \mathbf{R}} \tag{C.49}$$

$\frac{\partial \boldsymbol{X}}{\partial \mathbf{R}}$ is the $\mathbf{SO(3)}$ Lie Algebra.

$$\frac{\partial \boldsymbol{X}}{\partial \mathbf{R}} = \begin{bmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{bmatrix} \tag{C.50}$$

Thus:

$$\frac{\partial \boldsymbol{p'}}{\partial \mathbf{R}} = \begin{bmatrix} f_x/z & 0 & -x f_x/z^2 \\ 0 & f_y/z & -y f_y/z^2 \end{bmatrix} \begin{bmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{bmatrix} \tag{C.51}$$

$$\frac{\partial \boldsymbol{p'}}{\partial \mathbf{R}} = \begin{bmatrix} -xy f_x/z^2 & f_x + x^2 f_x/z^2 & -y f_x/z \\ -f_y - y^2 f_y/z^2 & xy f_y/x^2 & x f_y/z \end{bmatrix} \tag{C.52}$$

Therefore:

$$\frac{\partial P_u}{\partial \mathbf{R}} = \begin{bmatrix} -xyf_x/z^2 & f_x + x^2 f_x/z^2 & -yf_x/z \end{bmatrix} \tag{C.53}$$

$$\frac{\partial P_v}{\partial \mathbf{R}} = \begin{bmatrix} -f_y - y^2 f_y/z^2 & xyf_y/x^2 & xf_y/z \end{bmatrix} \tag{C.54}$$

For J_p, the partial derivative of the photometric parameters. Where p consists of a_j and b_j.

For a_j:

$$\partial \left((I_j[\boldsymbol{p'}] - b_j) - \frac{s_j e^{a_j}}{s_i e^{a_i}} (I_i[\boldsymbol{p}] - b_i) \right) \Big/ \partial a_j \tag{C.55}$$

$$= -\frac{s_j e^{a_j}}{s_i e^{a_i}} (I_i[\boldsymbol{p}] - b_i) \tag{C.56}$$

For b_j:

$$\partial \left((I_j[\boldsymbol{p'}] - b_j) - \frac{s_j e^{a_j}}{s_i e^{a_i}} (I_i[\boldsymbol{p}] - b_i) \right) \Big/ \partial a_j \tag{C.57}$$

$$= -1 \tag{C.58}$$

Glossary

Bayesian Estimation A statistical method that uses prior evidence to estimate an unknown population parameter. 4

camera exposure The amount of light that reaches a camera's sensor when a photo is taken. 12

CUDA An application programming interface (API) that allows programmers develop GPU code for general-purpose processing. 38

Delaunay Triangulation A computational technique that subdivides the convex hulls of a set of points into triangles whose circumcircles do not contain any of the points. It is used to organize points into triangular meshes. 37

Extended Kalman Filters A Kalman Filter is a mathematical algorithm that combines inaccurate and noisy measurements to produce estimates of unknown variables that tend to be more accurate than those based on a single measurement. The extended Kalman Filter is the nonlinear version of the Kalman filter. 4, 7, 8

gamma The nonlinear relationship in how light is captured and how it's perceived by humans. 12

Gauss-Newton An algorithm used to solve non-linear least squares problems. Equivalent to minimizing a sum of squared function values 15

loop closure A sub-algorithm of Simultaneous Localization and Mapping (SLAM) that corrects drift by recognizing when the observer returns to a place it has visited before. 4, 16

online operation Operations that process input data in real-time as it arrives. In contrast, offline operations processes all the collected data after after operations have finished. 1, 29

Particle Filters A nonlinear filtering algorithm that uses Monte Carlo sampling to find approximate solutions for the probability density of a system's state. 4, 7

photogrammetry The science of using photographs to create 3D models of objects and environments. 2, 8

radiance field A radiance field is a mathematical representation of how light interacts with a scene and the surfaces it encounters. 2, 22, 50, 52

rasterization The process of converting a mathematical image representation into a raster image, which is made up of pixels. 30, 69

scale ambiguity The inherent uncertainty in the absolute scale of a 3D scene when reconstruction is done from monocular 2D images. 5

signed distance field A map where each point represents the minimum distance to a boundary or surface. 20

vignetting The reduction in brightness on the edges of an image. 12

voxel A 3D counterpart to a pixel. It represents a section on a regular cubic grid in a 3D space. 20