SAPIENZA
UNIVERSITÀ DI ROMA

DOCTORAL THESIS

Towards Multi-Modal 3D Reconstruction: LiDAR-Camera Fusion for Surface and Radiance Field Modeling

Author:
Emanuele Giacomini

Advisor:
Prof. Giorgio Grisetti

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy in Engineering in Computer Science

in the

Department of Computer, Control and Management Engineering
"Antonio Ruberti" (DIAG)

June 2025

Thesis defended the 18th of September 2025.

Towards Multi-Modal 3D Reconstruction: LiDAR-Camera Fusion for Surface and Radiance Field Modeling
Ph.D. Thesis - Sapienza University of Rome
Advisor: Prof. Giorgio Grisetti

Reviewer: Prof. Jens Behley
Reviewer: Prof. Jose Luis Blanco Claraco

Author's email: giacomini@diag.uniroma1.it

Abstract

Accurate and efficient 3D scene representation is pivotal in spatial perception, from state estimation (Simultaneous Localization and Mapping and Structure from Motion) to higher-level scene understanding, such as panoptic segmentation. Recent advances in Radiance Fields have shown promising photorealistic and efficient appearance modeling results, yet these methods often struggle in texture-less, dynamic, or large-scale environments. While cameras are the de facto standard for appearance reconstruction, LiDARs could significantly mitigate these weaknesses. This thesis focuses on the strong linkage between the two sensors, highlighting parallels and fusion modalities. It first introduces a simple yet effective LiDAR-camera extrinsic calibration method using a planar target, typically used for vision systems. A direct formulation for Bundle Adjustment is shown, seamlessly integrating with LiDAR and camera pipelines, and this work presents solid claims highlighting the advantages of cross-sensor fusion for pose estimation. Sensor motion distortion, prevalent in LiDARs and rolling shutter (RS) cameras, is addressed through a novel de-skewing method that recovers intra-scan motion using spatiotemporal associations to compensate for distortion. The thesis also introduces a comprehensive multi-sensor benchmark dataset collected in Rome, which provides diverse, high-quality ground truth data for rigorous evaluation of multi-modal 3D reconstruction and localization methods. The central contribution lies in lifting Gaussian Splatting to a cross-modal context. Leveraging geometric consistency from LiDAR, the first LiDAR Odometry and Mapping pipeline is designed to rely on Gaussians as the sole scene representation, achieving state-of-the-art 3D reconstruction with a very low memory profile. Ultimately, the research journey expressed in this thesis highlights the complementarity of the two sensors and the strengths of fusing them for appearance and surface 3D reconstruction, while providing tools and datasets to advance robust multi-modal spatial perception.

Acknowledgments

Eccoci qui alla fine del mio percorso accademico, nel quale non ho solamente formato delle fondamenta solide (spero) a livello tecnico, ma sopratutto durante il quale ho avuto l'occasione di svilupparmi profondamente dal punto di vista umano. Ci tengo a voler ringraziare tutte ma proprio tutte le persone con cui ho avuto l'opportunità di parlare e condividere i miei pensieri e opinioni; anche quelle persone che, per diversi motivi, oramai sono uscite dalla mia vita.

Tra tutte le persone, cene sono alcune però che ritengo si meritino qualche parola in più.

In primis, Sabrina, che mi hai accompagnato durante tutto il mio percorso di dottorato, mi hai supportato nei momenti più difficili, e mi hai permesso di vivere tantissime meravigliose esperienze che ricorderò per sempre. Senza di te, non avrei potuto valutare questi anni come i migliori che abbia mai vissuto.

Un altro grazie va sicuramente a mamma Paola e papà Stefano, senza dei quali ovviamente non sarei qui e, che durante tutti questi anni hanno investito profondamente su di me, permettendomi di arrivare fin qui, senza mai avere preoccupazioni di alcun tipo a casa. Ovviamente ci tengo a ringraziare anche Stefania per aver accompagnato papà in questi anni e per i numerosi pranzi e cene deliziose che mi ha preparato.

Vorrei dedicare un altro ringraziamento a tutto il partito PPP con il quale ho condiviso praticamente tutto da oltre 5 *anni. Grazie per tutti i momenti che abbiamo passato insieme, dai più belli ai più tristi. In particolare vorrei ringraziare Antonio, Tobi, Marco e Luca. Ognuno di voi si meriterebbe almeno una pagina di ringraziamenti per tutto quello che ha fatto e condiviso con me negli ultimi* 10 *anni.*
Spero di non perdervi mai di vista nella vita.

Grazie anche a Francesco e Simone per avermi reso partecipe nei vostri percorsi universitari. Vi auguro veramente il meglio dalla vita. Velo meritate.

Vorrei anche ringraziare i Mitici 8, e in particolare Andrea, per avermi trasmesso dei fantastici spunti tecnici e umanistici (per questi ultimi, voglio ringraziare veramente tutto il gruppo).

Grazie a Francesco e Andrea per le innumerevoli risate durante tutti gli anni di università, sia in loco che online.

Grazie ad Andrea e Fabio per avermi accompagnato in mezzo alle montagne, ed avermi fatto scoprire questa passione per l'alpinismo e l'arrampicata.

Ovviamente non potevo non ringraziare i miei colleghi di dottorato e, in particolare vorrei ringraziare Luca, Leonardo, Omar, Simone e Lorenzo per tutte le magnifiche esperienze che abbiamo avuto modo di fare insieme durante questi 4 *anni. Siete persone meravigliose*

sotto tutti i punti di vista. Grazie, grazie e ancora grazie per tutto!

Un grande ringraziamento va sopratutto al prof Giorgio Grisetti, senza il quale non avrei potuto nemmeno vivere questa esperienza. Vorrei anche ringraziare il prof Martin R. Oswald per avermi ospitato ad Amsterdam durante il mio periodo all'estero.

Grazie ancora a tutti per avermi accompagnato in questo viaggio meraviglioso.

"So long and thanks for all the fishes."

Contents

List of Figures

List of Tables

Glossary

BA Bundle Adjustment

BRDF Bidirectional Reflectance Distribution Function

BSDF Bidirectional Scattering Distribution Function

DOF Degrees Of Freedom

FOV Field Of View

GD Gradient Descent

GN Gauss-Newton

ICP Iterative Closest Point

ILS Iterative Least-Squares

IMU Inertial Measurement Unit

IRLS Iterative Reweighed Least-Squares

LC Loop Closure

LiDAR Light Detection And Ranging

LM Levenberg-Marquardt

MC Monte Carlo

MLP Multi-Layer Perceptron

MVS Multi View Stereo

NCD Newer College Dataset

NDT Normal Distributed Transform

NeRF Neural Radiance Field

NeRFs Neural Radiance Fields

NVS Novel View Synthesis

PGO Pose-Graph Optimization

PnP Perspective-n-Point

PPS Pulse Per Second

RMSE Root Mean Square Error

RPE Relative Pose Error

SDF Signed Distance Function

SfM Structure from Motion

SLAM Simultaneous Localization and Mapping

SOTA State Of The Art

SVO Sparse Voxel Grid

TOF Time Of Flight

TSDF Truncated Signed Distance Function

VBR A Vision Benchmark in Rome

Chapter 1

Introduction

1.1 Motivation

Capturing and modeling the real world is a compelling challenge that finds diverse applications across different fields. Among these, cultural heritage preservation is particularly interesting. Artifacts, paintings, and historic buildings deteriorate over time, especially if kept under the exposure of sunlight or in contact with oxidizing agents. Accurate, millimeter-level reconstruction may lead to drastically reducing the deterioration of these works. Specifically, one could keep track of how external conditions or direct actions contribute to the object's degradation. Additionally, digital reconstruction facilitates broader access to such artifacts without contributing to their degradation[1].

Moreover, reconstructing the world becomes a requirement for autonomous systems which are becoming ever so popular within several critical fields such as automotive, search and rescue, logistics and, military.

Traditionally, 3D reconstruction methods rely on the use of RGB images captured from one or multiple cameras. The popularity of this sensor stems from two advantages: First, cameras are cheap and highly available (our smartphones typically have 2-3 or even more cameras) and second, they provide a visual-rich and dense snapshot of the world. Throughout the years, the scientific community pushed the boundaries of visual 3D reconstruction and as of today, using a smartphone, we can obtain photorealistic 3D representations that can even trick our eyes.

Nevertheless, RGB-based methods still face fundamental limitations, such as the inability to resolve scale ambiguities[2] and challenges with texture-less surfaces, which may lead to inaccurate results or even failures. Light Detection And Ranging (LiDAR) is another type of *active* optical sensors that emits laser pulses and accurately measures their Time Of Flight (TOF) to determine precise distances to surfaces. Something that is often repeated throughout this thesis is the fact that, remarkably, RGB cameras and LiDARs shows extremely orthogonal properties. Where cameras faces challenges with scale ambiguities and texture-less regions, LiDARs provide accurate range measurements, robust to surface texture[3]. Conversely, LiDAR measurements are typically sparse, whereas cameras yield dense

[1] Viewing a piece of art with our own eyes is definitely better! However, sometimes, it is important to keep the piece under isolation.

[2] Are we looking at a micro world from close up or a gigantic world from far away ? Apart from atmospheric effects, the images would be the same.

[3] Unless the surface is highly non-lambertian (i.e., mirrors), in which case both sensors may fail.

visual data. Despite LiDARs generally being more expensive, integrating both sensors can leverage their respective strengths, creating representations that are both visually appealing and geometrically accurate.

The orthogonality is also a key challenge to *tightly*-fuse[4] the two sensors. The first step in fusion is typically being able to find correspondences between the measurements. This is non-trivial when the nature of the sensors is so different[5]. Additionally, each sensor is affected by different physical processes that are not necessarily observable by both. A RGB camera may suffer from low-light condition that does not affect LiDARs. Moreover, as described in Chap. 5, LiDARs acquisition time is often not negligible and may lead to rolling-shutter effects that distort the measurements in case of high velocities.

Finally, finding a common scene representation is an additional challenge that should be tackled. Point-based representations have demonstrated powerful for online 3D reconstruction, however, it is important to account for the different measurement density of the sensors. Recently, with the advancements in Radiance fields, volumetric representations (described in Sec. 2.5.1 and used in Chap. 7 and Chap. 8) are emerging as an appealing representation for LiDAR-camera integration.

To conclude, LiDAR-camera fusion is a promising direction for robust and accurate 3D reconstruction by jointly leveraging geometric and appearance cues. This thesis investigates methods and representations for handling the two sensors in a unified manner.

1.2 Thesis Outline

Background. Chapter 2 presents the tools and required knowledge to grasp the following chapters. Specifically, it describes the mathematical tools to handle pose relationships, non-linear optimization problems and models to describe the sensor's measurements.

LiDAR-RGB Calibration using Common Markers. In Chapter 3, we focus on LiDAR and RGB sensors that exhibit complementary capabilities. Given the sparsity of the LiDAR measurements, current state-of-the-art calibration techniques often rely on complex or large calibration targets to resolve the relative pose estimation. As such, the geometric properties of the targets may hinder the calibration procedure in those cases where an ad-hoc environment cannot be guaranteed. This chapter addresses the problem of LiDAR-RGB calibration using common calibration patterns (i.e., A3 chessboard) with minimal human intervention. Our approach exploits the flatness of the target to find associations between the sensors' measurements, leading to robust features and retrieving the solution through non-linear optimization. The results of quantitative and comparative experiments with other state-of-the-art approaches show that our simple schema performs on par or better than existing methods that rely on complex calibration targets. We release an open-source implementation at `https://github.com/rvp-group/ca2lib`.

Photometric LiDAR-RGB Bundle Adjustment. Chapter 4 presents a unified Bundle Adjustment (BA) strategy that seamlessly integrates LiDAR and RGB sensors. The joint optimization of the sensor trajectory and 3D map is a crucial characteristic of Simultaneous

[4] Optimizing a problem by jointly considering both sensors. Alternatively, *loosely*-coupled methods often resolve sequential sub-problems (i.e., resolving a LiDAR-only problem first and then using its solution to complete a camera-only problem).

[5] Identifying which pixel in an image coincides with the surface point measured by the LiDAR

Localization and Mapping (SLAM) systems. To achieve this, the gold standard is BA. Modern 3D LiDARs retains higher resolutions that enable the creation of point cloud images resembling those taken by conventional cameras. Nevertheless, the typical effective global refinement techniques employed for RGB-D sensors are not widely applied to LiDARs. This work can be used on top of any SLAM/GNSS estimate to improve and refine the initial trajectory. The presented results show that this system performs on par or better compared to other state-of-the-art ad-hoc SLAM/BA strategies, free from data association and without making assumptions about the environment. In addition, the chapter presents the benefit of jointly using RGB-D and LiDAR within a unified method. We release an open-source CUDA/C++ implementation at https://github.com/rvp-group/ba-mdslam.

Compensating rolling-shutter effects of 2D-LiDARs. Most commercially available LiDARs measure the distances along a 2D section of the environment by sequentially sampling the free-range along directions centered at the sensor's origin. When the sensor moves during the acquisition, the measured ranges are affected by a phenomenon known as "skewing", which appears as a distortion in the acquired scan. Skewing potentially affects all systems that rely on LiDAR data, however it could be compensated if the position of the sensor were known each time a single range is measured. Most methods to de-skew LiDAR measurements are based on external sensors such as IMU or wheel odometry, to estimate these intermediate LiDAR positions. This chapter present a method that relies exclusively on range measurements to effectively estimate the robot velocities which are then used for de-skewing. This approach is suitable for low frequency LiDARs where the skewing is more evident. It can be seamlessly integrated into existing pipelines, enhancing their performance at negligible computational cost. We validated the proposed method with statistical experiments characterizing different operating conditions. We release an open-source C++ implementation at https://gitlab.com/srrg-software/srrg2_laser_deskewer.

Vision Benchmark in Rome. Chapter 6 presents a vision and perception research dataset collected in Rome, featuring RGB data, 3D point clouds, IMU, and GPS data. This work complements existing datasets by simultaneously addressing several issues, such as environment diversity, motion patterns, and sensor frequency. It uses up-to-date devices and presents effective procedures to accurately calibrate the intrinsic and extrinsic of the sensors while addressing temporal synchronization. During recording, we cover multi-floor buildings, gardens, urban and highway scenarios. Combining handheld and car-based data collections, our setup can simulate any robot (quadrupeds, quadrotors, autonomous vehicles). The dataset includes an accurate 6-Degrees Of Freedom (DOF) ground truth based on a novel methodology that refines the RTK-GPS estimate with LiDAR point clouds through BA. All sequences divided in training and testing are accessible at https://rvp-group.net/datasets/slam.html.

2D Gaussian Splatting for LiDAR. Chapter 7 introduces a formulation to enable 2D Gaussian Splatting on LiDAR-only systems. We build on recent advancements in Gaussian Splatting methods to develop a novel LiDAR odometry and mapping pipeline that exclusively relies on Gaussian primitives for its scene representation. Leveraging spherical projection, we drive the refinement of the primitives uniquely from LiDAR measurements. Experiments show that our approach matches the current registration performance, while achieving State Of The Art (SOTA) results for mapping tasks with minimal GPU requirements. This efficiency makes it a strong candidate for further exploration and potential adoption in real-time robotics estimation tasks. We release an open-source Python/CUDA implementation at

https://github.com/rvp-group/Splat-LOAM.

Hash-based GPU-Accelerated Sparse Voxel Grid for 3D Reconstruction. Efficient and scalable 3D surface reconstruction from range data remains a core challenge in computer graphics and vision, particularly in real-time and resource-constrained scenarios. Traditional volumetric methods based on fixed-resolution voxel grids or hierarchical structures like octrees often suffer from memory inefficiency, computational overhead, and poor GPU compatibility. Chapter 8 introduces a novel variance-adaptive, multi-resolution voxel grid that dynamically adjusts voxel size based on the local variance of Signed Distance Function (SDF) observations. Unlike prior multi-resolution approaches that rely on recursive octree structures, our method leverages a flat spatial hash table to store all voxel blocks, supporting constant-time access and full GPU parallelism. This design enables seamless integration, high memory efficiency, and real-time scalability. We further demonstrate how our representation supports GPU-accelerated rendering through a parallel quad-tree structure for Gaussian Splatting, enabling effective control over splat density. Our open-source CUDA/C++ implementation achieves up to 13× speed-up and 4× lower memory usage compared to fixed-resolution baselines, while maintaining on par results in terms of reconstruction accuracy, offering a practical and extensible solution for high-performance 3D reconstruction.

Conclusions. Chapter 9 sums the overall contributions and discusses interesting future direction to seamlessly fuse LiDAR and RGB measurements in a single 3D reconstruction framework.

Publications. The contents of the thesis are based on the publications [41, 29, 102, 12, 42]. Chapter 8 presents a work that, at the time of writing is submitted at SIGGRAPH Asia 2026.

Chapter 2

Background

This section presents the mathematical background required to grasp the following chapters. First, Section 2.2 introduces the fundamental instruments to model spatial relationship between entities. Next, Section 2.3 introduces the two fundamental sensors that this thesis mention, namely RGB and RGB-D cameras in Sec. 2.3.1 and LiDARs in Sec. 2.3.2. Section 2.4 introduces the tools to handle the non-linear optimization problems that will be encountered throughout this thesis. Moreover, Section 2.5 describes the problem of inverse rendering and provides a background on the currently established methods.

2.1 Homogeneous Coordinates

For transformations and especially for projective geometry in general, representing elements in a homogeneous form, often leads to symmetric and simpler forms than the Cartesian form. The real projective space $\mathbb{P}^3$ can be thought as the union of the real space $\mathbb{R}^3$ with additional points that are on a *plane* at infinity. The union is actually obtained by augmenting $\mathbb{R}^3$ vectors with an additional coordinate, so that a $\mathbb{P}^3$ element is a 4-dimensional vector

$$\mathbf{p} = (x_1, x_2, x_3, x_4)^T \in \mathbb{P}^3. \tag{2.1}$$

When $x_4 \neq 0$, then the point $\mathbf{p}$ represents the real vector $(x, y, z)^T \in \mathbb{R}^3$ with inhomogeneous coordinates:

$$x = x_1/x_4, \quad y = x_2/x_4, \quad z = x_3/x_4 \ , \tag{2.2}$$

while for $x_4 = 0$ it represents a point at infinity. To demonstrate the advantages of this representation, we briefly present how point and lines interact in $\mathbb{P}^2$. This is just an example, but similar properties can be found in other formulations in $\mathbb{P}^2$ and $\mathbb{P}^3$.

Line definition: A line $\mathbf{l} \in \mathbb{P}^2$ can be defined by two points $\mathbf{x}$ and $\mathbf{x}'$ if it satisfies

$$\mathbf{x}^T\mathbf{l} = 0, \quad \mathbf{x}'^T\mathbf{l} = 0 \ . \tag{2.3}$$

Eq. (2.5) always admits a solution of the form $\mathbf{l} = \mathbf{x} \times \mathbf{x}' = (a, b, c)$. If $\mathbf{x} = \mathbf{x}'$, then $c = 0$, meaning that $\mathbf{l}$ does not exist in the real plane but rather lives at infinity.

Line Intersection: Let $\mathbf{l}$ and $\mathbf{l}'$ be two lines we wish to find the intersection of. Let $\mathbf{x}$ be the vector

$$\mathbf{x} = \mathbf{l} \times \mathbf{l}' = (a, b, c) \tag{2.4}$$

where $\times$ is the cross product. Then, it is also true that

$$\mathbf{l}^T\mathbf{x} = \mathbf{l}'^T\mathbf{x} = 0. \tag{2.5}$$

If $\mathbf{x}$ is thought to represent a point, then Eq. (2.5) indicates that it is the point that intersects $\mathbf{l}$ and $\mathbf{l}'$. Additionally, if $\mathbf{l}$ and $\mathbf{l}'$ are parallel lines, then $c = 0$, meaning that bx lives at the infinity and therefore, $\mathbf{l}$ and $\mathbf{l}'$ do intersect at the infinity.

2.2 Spatial relationships

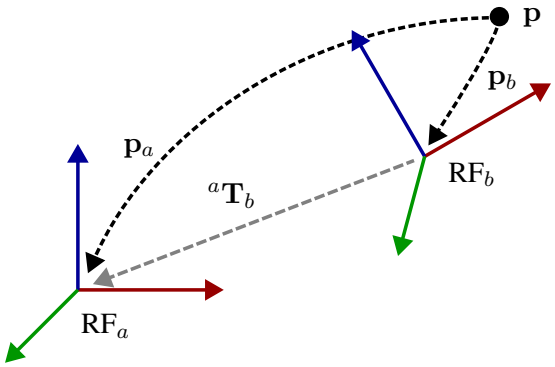


Figure 2.1. Relationship between two reference systems. The variable ${}^a\mathbf{T}_b$ describes how a point $\mathbf{p}_b$ expressed in RF_b is seen from RF_a. Or more formally, $\mathbf{p}_a = {}^a\mathbf{T}_b\mathbf{p}_b$

We are interested at modeling the spatial relationship between two or more entities. Such relationship are everywhere, from modeling the relative pose between two sensors, to object tracking, pose estimation and so forth. Specifically, we will assume such relationship to be *rigid* or rather *isometric*, meaning that it should preserve the Euclidean distance between every pair of points. Formally, we can express the position of a point $\mathbf{p}$ with respect to the coordinate frame $O{-}xyz$ as:

$$\mathbf{p} = \mathbf{p}_x\mathbf{x} + \mathbf{p}_y\mathbf{y} + \mathbf{p}_z\mathbf{z} \tag{2.6}$$

where $(\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z)$ are the components of the vector $\mathbf{p}$ along the frame axes. To describe the orientation, imagine linking an orthonormal frame to $\mathbf{p}$ such that $(\mathbf{x}', \mathbf{y}', \mathbf{z}')$ are the orthonormal axis forming the base whose origin lies on $\mathbf{p}$. We can describe the orientation of $\mathbf{p}$ with respect to the coordinate frame $O{-}xyz$ as

$$\begin{aligned} \mathbf{x}' &= \mathbf{x}'_x\mathbf{x} + \mathbf{x}'_y\mathbf{y} + \mathbf{x}'_z\mathbf{z} \\ \mathbf{y}' &= \mathbf{y}'_x\mathbf{x} + \mathbf{y}'_y\mathbf{y} + \mathbf{y}'_z\mathbf{z}, \\ \mathbf{z}' &= \mathbf{z}'_x\mathbf{x} + \mathbf{z}'_y\mathbf{y} + \mathbf{z}'_z\mathbf{z} \end{aligned} \tag{2.7}$$

where each vector is defined as the sum of the direction cosines between it and the reference basis. We can reformulate Eq. (2.7) that relates the orientation between two frames in its matrix form:

$$\mathbf{R} = \begin{pmatrix} \mathbf{x}' & \mathbf{y}' & \mathbf{z}' \end{pmatrix} = \begin{pmatrix} \mathbf{x}'_x & \mathbf{y}'_x & \mathbf{z}'_x \\ \mathbf{x}'_y & \mathbf{y}'_y & \mathbf{z}'_y \\ \mathbf{x}'_z & \mathbf{y}'_z & \mathbf{z}'_a \end{pmatrix} = \begin{pmatrix} \mathbf{x}'^T\mathbf{x} & \mathbf{y}'^T\mathbf{x} & \mathbf{z}'^T\mathbf{x} \\ \mathbf{x}'^T\mathbf{y} & \mathbf{y}'^T\mathbf{y} & \mathbf{z}'^T\mathbf{y} \\ \mathbf{x}'^T\mathbf{z} & \mathbf{y}'^T\mathbf{z} & \mathbf{z}'^T\mathbf{z} \end{pmatrix}. \tag{2.8}$$

Additional properties can be deduced from $\mathbf{R}$ (Additional proofs are found in Sec. A.1). Specifically, since its columns are mutually orthogonal and have unit norm, $\mathbf{R}$ is an *orthogonal* matrix, meaning:

$$\mathbf{R}^T\mathbf{R} = \mathbf{I}_3. \tag{2.9}$$

Furthermore, post multiplying each side by $\mathbf{R}^{-1}$, we reach

$$\mathbf{R}^T = \mathbf{R}^{-1}, \tag{2.10}$$

meaning that the transpose of a rotation matrix coincides with its inverse. This special case of matrix is part of the *Special Orthogonal* Lie group $\mathbb{SO}(3)$. Further details on how to update rotation matrices while preserving its properties are found in Sec. 2.4.3.

2.2.1 Orientation

Rotation matrices are quite handy for representing rotations, however, they provide a non-minimal set of parameters to express an orientation. Specifically, each matrix provides a set of 9 parameters related by 6 constraints due to orthogonality conditions previously described. At least in $\mathbb{R}^3$, this implies that only 3 parameters could suffice to minimally represent a single rotation. Historically, several representations were introduced. For the sake of simplicity, this thesis provides a description of only two of them.

Euler Angles

A minimal representation can be developed by using three angles $\boldsymbol{\phi} = (\alpha, \beta, \theta)^T$. If each parameter describes the rotation of a single axis (representable as a rotation matrix), then a generic rotation matrix can be obtained by composing an appropriate sequence[1] of elementary rotations. This allows to define up to 12 different set of angles and rotations that fall under the name of *Euler angles*. One of the 12 parametrizations will be now described, however, similar conclusions can be drawn for the others.

RPY Angles (ZYX): This representation if often encountered in the aeronautical field, and it is used to model the orientation of planes or vectors. Also known as *Roll-Pitch-Yaw angles* $\boldsymbol{\phi} = (\alpha, \beta, \theta)$, they represent a rotation defined with respect to a fixed frame, and are obtained by the following rules:

- Reference is rotated by θ about the $\mathbf{x}$ axis (yaw), as described by $\mathbf{R}_x(\gamma)$
- Reference is rotated by β about the $\mathbf{y}$ axis (pitch), as described by $\mathbf{R}_y(\beta)$
- Reference is rotated by α about the $\mathbf{z}$ axis (roll), as described by $\mathbf{R}_z(\alpha)$

To ease the notation, let $c_x = \cos(x)$ and $s_x = \sin(x)$, then, the final rotation matrix is

$$\begin{aligned}\mathbf{R}(\boldsymbol{\phi}) &= \mathbf{R}_z(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_x(\theta) \\ &= \begin{pmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\theta - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\theta + c_\alpha c_\gamma & s_\alpha s_\beta s_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\theta & c_\beta c_\gamma \end{pmatrix}.\end{aligned} \tag{2.11}$$

[1] A sequence that guarantees that two successive rotations are not made on parallel axes.

As for other Euler angles, it is also possible to find the inverse solution. Given a valid rotation matrix:

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{2.12}$$

then, a solution for $\alpha \in (-\pi/2, \pi/2)$ is:

$$\begin{aligned} \alpha &= \operatorname{atan2}(r_{21}, r_{11}) \\ \beta &= \operatorname{atan2}\left(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}\right), \\ \theta &= \operatorname{atan2}(r_{32}, r_{33}) \end{aligned} \tag{2.13}$$

where $\operatorname{atan2}$ is define as:

$$\operatorname{atan2}(y, x) = \begin{cases} \arctan(y/x) & x > 0, \\ \arctan(y/x) + \pi & x < 0 \wedge y \geq 0, \\ \arctan(y/x) - \pi & x < 0 \wedge y < 0, \\ +\frac{\pi}{2} & x = 0 \wedge y > 0, \\ -\frac{\pi}{2} & x = 0 \wedge y < 0, \\ \text{undefined} & x = 0 \wedge y = 0. \end{cases} \tag{2.14}$$

One major problem of Euler angles is known as *gimbal lock*, which describes the loss of one Degrees Of Freedom (DOF) in specific configurations that depends on the parametrization itself. This effect can be observed by setting $\beta = \pi/2$ in Eq. (2.11) and obtaining

$$\mathbf{R}(\alpha, \theta) = \begin{pmatrix} 0 & c_\alpha s_\theta - s_\alpha c_\gamma & c_\alpha c_\gamma + s_\alpha s_\gamma \\ 0 & s_\alpha s_\theta + c_\alpha c_\gamma & s_\alpha s_\gamma - c_\alpha c_\gamma \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & s_{\theta-\alpha} & c_{\gamma+\alpha} \\ 0 & -c_{\theta+\alpha} & s_{\gamma-\alpha} \\ -1 & 0 & 0 \end{pmatrix}, \tag{2.15}$$

where both angles α and θ contribute to the same rotation over the $\mathbf{z}$ axis. This makes Euler angles not particularly suited for modeling entities that may have full orientation freedom (i.e., robotic manipulators), as crossing the singularities may lead to undesired situations.

Quaternions

One established solution to avoid the gimbal lock is by using a non-minimal representation for orientation with 4 parameters. Unit quaternions are one possible representation defined as $\mathbf{q} = (w1 + x\mathbf{i} + y\mathbf{j} + z\mathbf{k})$ where $(1, \mathbf{i}, \mathbf{j}, \mathbf{k})$ forms a complex orthonormal basis. An alternative formulation required for further derivations is $\mathbf{q} = (w, \boldsymbol{\mu})$ where

$$\begin{aligned} w &= \cos\left(\frac{\theta}{2}\right) \\ \boldsymbol{\mu} &= \sin\left(\frac{\theta}{2}\right)\begin{pmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \end{pmatrix} \end{aligned}. \tag{2.16}$$

The "unit" prefix is included due to a normalization constraint over its parameters:

$$w^2 + x^2 + y^2 + z^2 = 1. \tag{2.17}$$

Quaternions have several useful properties for modeling rotations like having no singular configurations and uniqueness [2]. The rotation matrix form given a quaternion is

$$\mathbf{R}(\mathbf{q}) = \begin{pmatrix} 2\left(w^2+x^2\right)-1 & 2\left(xy-wz\right) & 2\left(xz+wy\right) \\ 2\left(xy-wz\right)-1 & \left(w^2+y^2\right)-1 & 2\left(yz-wx\right) \\ 2\left(xz+wy\right) & 2\left(yz-wx\right) & \left(w^2+z^2\right)-1 \end{pmatrix}. \tag{2.18}$$

Additionally, given a rotation matrix like Eq. (2.12), the corresponding quaternion $\mathbf{q}$ is computed as

$$\begin{aligned} w &= \frac{1}{2}\sqrt{r_{11}+r_{22}+r_{33}+1} \\ \boldsymbol{\mu} &= \frac{1}{2}\begin{pmatrix} \text{sign}(r_{32}-r_{23})\sqrt{r_{11}-r_{22}-r_{33}+1} \\ \text{sign}(r_{13}-r_{31})\sqrt{r_{22}-r_{33}-r_{11}+1} \\ \text{sign}(r_{21}-r_{12})\sqrt{r_{33}-r_{11}-r_{22}+1} \end{pmatrix}. \end{aligned} \tag{2.19}$$

Finally, it's possible to conclude that:

- $\mathbf{q}\left(\mathbf{I}_3\right) = (1, \mathbf{0}_3)$ represents the identity quaternion
- $\mathbf{q}\left(\mathbf{R}^{-1}\right) = (w, -\boldsymbol{\mu})$ represents the inverse quaternion.

2.2.2 Special Euclidean Group $\mathbb{SE}(3)$

As previously described, it's possible to fully express the spatial relationship between two oriented frames by expressing their relative orientation and position. As shown in Fig. 2.1, let $\mathbf{p}_b$ be the position of the point $\mathbf{p}$ with respect to RF$_b$, then the position $\mathbf{p}_a$ with respect to RF$_a$ is found as:

$$\mathbf{p}_a = {}^a\mathbf{R}_b\mathbf{p}_b + {}^a\mathbf{t}_b, \tag{2.20}$$

where ${}^a\mathbf{R}_b$ and ${}^a\mathbf{t}_b$ represents the rotation matrix and translation vector from RF$_b$ to RF$_a$. Additionally, $\mathbf{p}_b$ can be obtained by $\mathbf{p}_a$ by solving the inverse problem:

$$\mathbf{p}_b = {}^a\mathbf{R}_b^T\mathbf{p}_a - {}^a\mathbf{R}_b^T{}^a\mathbf{t}_b = {}^b\mathbf{R}_a\mathbf{p}_a - {}^b\mathbf{R}_a{}^a\mathbf{t}_b. \tag{2.21}$$

This representation can be compacted into a single matrix representation using homogeneous coordinates described in Sec. 2.1. Let the homogeneous point $\hat{\mathbf{p}}$ be defined as:

$$\hat{\mathbf{p}}_b = \left(\mathbf{p}_b, 1\right)^T, \tag{2.22}$$

then, Eq. (2.20) can be rewritten as:

$$\hat{\mathbf{p}}_a = \begin{pmatrix} {}^a\mathbf{R}_b & {}^a\mathbf{t}_b \\ \mathbf{0}_3 & 1 \end{pmatrix}\hat{\mathbf{p}}_b = {}^a\mathbf{T}_b\hat{\mathbf{p}}_b, \tag{2.23}$$

where $\mathbf{0}_3 = (0,0,0)^T$.
Similarly, Eq. (2.21) can be rewritten as:

$$\hat{\mathbf{p}}_b = \begin{pmatrix} {}^a\mathbf{R}_b^T & -{}^a\mathbf{R}_b^T{}^a\mathbf{t}_b \\ \mathbf{0}_3 & 1 \end{pmatrix}\hat{\mathbf{p}}_a = {}^b\mathbf{T}_a\hat{\mathbf{p}}_a. \tag{2.24}$$

Notice that, in contrast with $\mathbb{SO}(3)$ elements, ${}^a\mathbf{T}_b^T \neq {}^b\mathbf{T}_a$.

[2] Representations like axis-angle have different representation for the rotation $(\theta, \boldsymbol{\mu})$ and $(-\theta, -\boldsymbol{\mu})$

2.3 Sensors

The core idea of this thesis is to entangle two apparently different sensors under the same working methodology. This section introduces each sensor along with the core mathematical instruments required to operate over their measurement.

Concerning cameras, this thesis will introduce only the pinhole camera model.

2.3.1 Pinhole Cameras

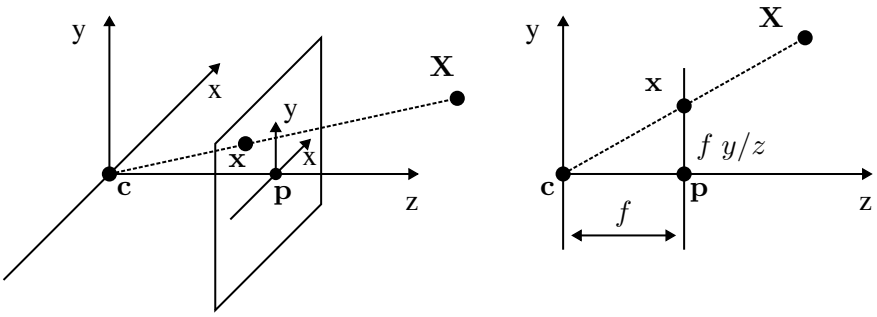


Figure 2.2. Pinhole camera geometry. $\mathbf{c}$ is the camera center, $\mathbf{p}$ is the principal point. The image plane is placed at a distance f from the camera center.

The most established way to capture our reality is through cameras. The principle behind cameras was discovered more than 2 thousands years ago. Given a dark room with a single tiny hole on one of its walls, it was possible to observe an inverted "projection" of the outside world onto the opposite wall. This concept was later (18-th to 19-th century) formalized and ultimately led to the development of the first photographic cameras.

To grasp the *pinhole* camera, we have to state that: (1) Light rays travel on straight lines and (2) When hitting a surface, a component of the ray is absorbed and the rest is reflected, carrying information concerning the color of the surface. Moreover, assuming the world to be composed by Lambertian surfaces[3], any given point in space would be reached by light rays emitted by every visible surface point. To allow a "projection" to occur, we need a dark room with a single tiny hole on one of its walls. The hole acts as a filter for the light rays, allowing only the ones that passes through it to reach inside the room. This effectively creates a unique mapping between light ray directions and points within the opposite side of the room (from now referred as the image plane). To better model this phenomenon, let us slightly refactor the geometric problem and introduce several notions.

Projection

Let the center of projection be the origin of a Euclidean reference frame and, consider the plane $z = f$ denoted as image plane, as shown in Fig. 2.2. Under the pinhole projection, a point $\mathbf{X} = (X, Y, Z)^T$ is mapped to the point $\mathbf{x}$ on the image plane where a line joints $\mathbf{X}$ with the projection center $\mathbf{c}$. The point $\mathbf{x}$ is computed as:

$$(X, Y, Z)^T \rightarrow (fX/Z, fY/Z)\,. \tag{2.25}$$

[3]Ideal surfaces that reflect light equally in all directions.

Using homogeneous coordinates, Eq. (2.25) can be rewritten as a mapping from a homogeneous 4-space vector $\mathbf{X}$ to a homogeneous 3-space vector $\mathbf{x}$ as

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.26}$$

Additionally, since the origin of the image plane might not coincide with the principal point $\mathbf{p}$ and focal lengths might vary between the x and y axes Eq. (2.26) can be generalized by including it as

$$\mathbf{x} = \begin{pmatrix} f_x & 0 & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \mathbf{K}\left[\mathbf{I}_3|\mathbf{0}\right]\mathbf{X} \tag{2.27}$$

where f_x and f_y represents per-axis focal lengths and $\mathbf{K}$ is referred as camera matrix [4]. If $\mathbf{X}$ is not expressed in the camera coordinate frame, we can further generalize the projection as

$$\mathbf{x} = \mathbf{K}\begin{pmatrix} {}^c\mathbf{R}_w & {}^c\mathbf{t}_w \end{pmatrix}\mathbf{X}_w = \mathbf{K}\left[{}^c\mathbf{R}_w|{}^c\mathbf{t}_w\right]\mathbf{X}_w = \mathbf{P}\mathbf{X}_w \tag{2.28}$$

where $({}^c\mathbf{R}_w|{}^c\mathbf{t}_w)$ is the first three rows of the homogeneous transformation matrix that maps world points into the camera reference frame and $\mathbf{P}$ is a 3×4 matrix referred as projection matrix. Additional generalizations of the pinhole camera model can be found in [47].

Throughout this work, we will refer to the pinhole projection function $\boldsymbol{\pi}_c : \mathbb{P}^3 \rightarrow \mathbb{P}^2$ as

$$\boldsymbol{\pi}_c(\mathbf{X}_c, \mathbf{K}) = \mathbf{K}\left[\mathbf{I}|\mathbf{0}\right]\mathbf{X}, \tag{2.29}$$

where the dependency to $\mathbf{K}$ will be omitted for ease of reading.

Back-Projection

In several chapters, we will encounter the concept of back-projection, which intuitively refers to the inverse projection operation. This operation is typically used to recover geometric cues of the projected scene, and it's often fundamental to solve multi-view problems. Starting from Eq. (2.25), notice that during the projection, the *Z depth* information is lost. Intuitively, this means that, given a point on image plane $\mathbf{x}$, through an inverse operation, we're only able to recover

$$\mathbf{X} = \boldsymbol{\pi}_c^{-1}\mathbf{x} = \mathbf{K}^{-1}\mathbf{x} = \begin{pmatrix} Xz \\ Yz \\ z \end{pmatrix} \; z \in \mathbb{R}. \tag{2.30}$$

The duality principle [47] states that points and lines share the same representation in homogeneous form. In the case of Eq. (2.30), $\mathbf{X}$ can be interpreted as the line that joins the camera center $\mathbf{c}$ with the point on the image plane $\mathbf{x}$ and thus, the family of points such that

$$\mathbf{X} = \boldsymbol{\pi}_c^{-1}(\mathbf{x}) \; : \; \left\{\mathbf{p} \in \mathbb{P}^3 : \boldsymbol{\pi}_c(\mathbf{p}) = \mathbf{x}\right\}. \tag{2.31}$$

[4] Real CCD cameras may have non-square pixels, leading to differences between the focal length on x and y axes.

Depth Estimation

The geometry of the scene from a vision perspective can be estimated either via the use of at least two-view or N-views [47] or via neural monocular depth priors. Apart from their differences, each method aims at estimating the per-pixel depth values. In the pinhole context, the depth is referred to the z-distance of the surface projected at pixel $\mathbf{x}$ from the camera center.

2.3.2 Light Detection And Ranging

An alternative technology often used in the context of 3D reconstruction is Light Detection And Ranging (LiDAR). Differently from optical cameras which relies on visual cues, LiDARs leverage laser Time Of Flight (TOF) technology to infer the geometry of the scene. After emitting IR laser pulses, a receiver circuit accurately measures the round trip time Δt it takes for the light to reach the surface and back to the sensor. By knowing the speed of light c, the distance d between the emitter-receiver pair and the collided surface is computed as

$$d = c\frac{\Delta t}{2}. \tag{2.32}$$

While the most simple form of LiDAR sensor is a single, static receiver-emitter pair, commercial ones contains several pairs to provide higher frequency or higher density measurements. Throughout this thesis, we will focus mostly on 3D mechanical rotating LiDARs which provide full $360°$ scans of the environment. These sensors mount a vertical tilted set of emitters (typically 16 to 128) that provides up to $90°$ of vertical Field Of View (FOV). Full panoramic measurements are obtained by mounting the set of emitters onto a spinning motor equipped with an encoder to measure angular positions. At the moment of writing, the fastest commercial LiDAR can provide measures at 20 Hz. Chap. 5 will introduce and discuss the problem of the skewing effect observed when LiDARs are subject to strong motions.

Spherical Projection

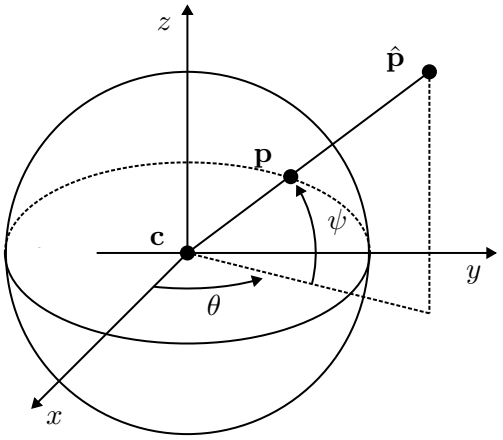


Figure 2.3. Spherical Coordinates. $\mathbf{c}$ is the center of projection, $\mathbf{p}$ is the normalized point and (θ, ψ) is the corresponding azimuthal and elevation coordinates.

This thesis focuses on parallels between cameras and LiDARs. This section introduces a formulation that allows to express the 3D point cloud of a mechanical LiDAR into a panoramic image-like representation. First, consider a set of points $\{\hat{\mathbf{p}}_i\}_{i=1}^{N}$ expressed in sensor frame and representing the measurement of a LiDAR. The point $\hat{\mathbf{p}} = (a, b, c)^T$ has its real projective $\mathbb{P}^3$ representation $\mathbf{p} = (x, y, z, 1)^T$ as

$$\begin{aligned} \mathbf{p} &= (da, db, dc, d)^T \\ d &= \sqrt{a^2 + b^2 + c^2}. \end{aligned} \tag{2.33}$$

We denote d as the *range* measurement, which, similarly to the *depth* for cameras, describes the distance between the point $\hat{\mathbf{p}}$ and the center of projection $\mathbf{c}$. As depicted in Fig. 2.3, let $\phi(\mathbf{p}) : \mathbb{R}^3 \rightarrow \mathbb{S}^2$ be the mapping from Cartesian to spherical coordinates defined as:

$$\phi(\mathbf{p}) = \begin{pmatrix} \operatorname{atan2}(y, x) \\ \operatorname{atan2}\left(z, \sqrt{x^2 + y^2}\right) \\ \sqrt{x^2 + y^2 + z^2} \end{pmatrix} = \begin{pmatrix} \theta \\ \psi \\ 1 \end{pmatrix}. \tag{2.34}$$

Under this formulation $\{\theta, \psi\} \in [-\pi, \pi)$ and we can unfold the sphere to achieve a Cartesian space with coordinate discontinuities at $\pm\pi$, as shown in Fig. 2.4. The discontinuity at $\psi \pm \pi$ is always disregarded due to the limited vertical FOV of commercial LiDARs, while the one at $\theta \pm \pi$ does not lead to any particular issue except for the reasons described in Chap. 7, in which it is explicitly handled. Finally, points can be transformed in image plane through an affine transformation

$$\mathbf{x} = \begin{pmatrix} f_\theta \theta + c_\theta \\ f_\psi \psi + c_\psi \\ 1 \end{pmatrix} = \begin{pmatrix} f_\theta & 0 & c_\theta \\ 0 & f_\psi & c_\psi \\ 0 & 0 & 1 \end{pmatrix} \phi(\mathbf{p}) = \mathbf{K}' \phi(\mathbf{p}), \tag{2.35}$$

where $\mathbf{K}'$ is fundamentally equal to the camera matrix described in Sec. 2.3.1.

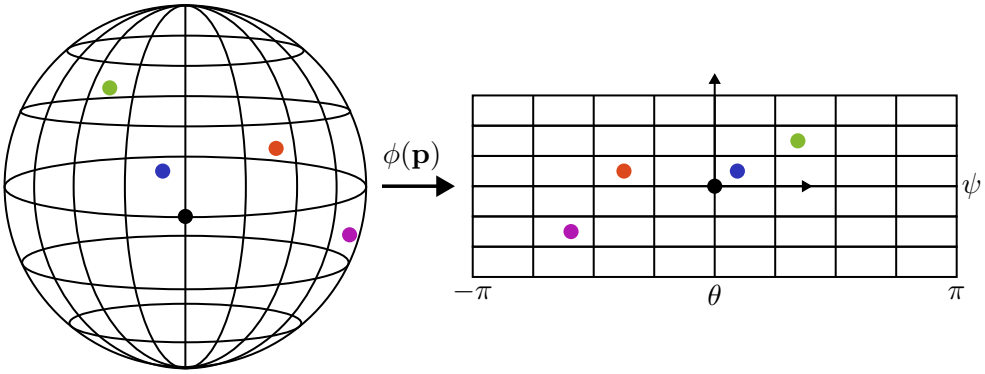


Figure 2.4. Spherical coordinates unfold. The operation causes distortions and discontinuities at $\pm\pi$ but provide a $\mathbb{R}^2$ panoramic snapshot of the surroundings.

Throughout this work, we will refer to the spherical projection function $\boldsymbol{\pi}_l : \mathbb{P}^3 \rightarrow \mathbb{P}^2$ as

$$\boldsymbol{\pi}_l(\mathbf{p}, H, W) = \mathbf{K}'(\{\mathbf{p}_i\}_{i=1}^{N}, H, W)\phi(\mathbf{p}). \tag{2.36}$$

Contrary to the camera matrix in which parameters model fixed physical quantities, $\mathbf{K}'$ can be estimated for each new point cloud and optimized to maximize the number of occupied pixels.

Estimating the LiDAR camera matrix

Using hard-coded field of views from the sensor's datasheet may lead to empty areas inside the image (i.e., when parts of the FOV contain no observable environment or, due to rounding errors). To solve these issues, we recompute the field of views and the principal direction for each incoming LiDAR measurement.

Let $\{\mathbf{v}_i = (\theta, \psi, 1)^T = \phi(\mathbf{p}_i)\}_{i=1}^N$ be the input set of points expressed in spherical coordinates. First, the bounds for each angle is computed as

$$\theta_m = \min_\theta \{\mathbf{v}_i\} \qquad \theta_M = \max_\theta \{\mathbf{v}_i\} \tag{2.37}$$

$$\psi_m = \min_\psi \{\mathbf{v}_i\} \qquad \psi_M = \max_\psi \{\mathbf{v}_i\}\,. \tag{2.38}$$

Furthermore, given an image size (H, W), the camera matrix $\mathbf{K}'$ is computed as

$$\mathbf{K}'\left(\{\mathbf{p}_i\}, H, W\right) = \begin{pmatrix} -\frac{W-1}{\theta_M - \theta_m} & 0 & \frac{W}{2}\left(1 + \frac{\theta_M + \theta_m}{\theta_M - \theta_m}\right) \\ 0 & -\frac{H-1}{\psi_M - \psi_m} & \frac{H}{2}\left(1 + \frac{\psi_M + \psi_m}{\psi_M - \psi_m}\right) \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.39}$$

2.4 Least Squares Optimization

In virtually every chapter, we will encounter a phase in which, given some measurements $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^M$, we are interested in estimating some unknown variables (state) $\mathbf{x}$ concerning our device or the external world. Albeit some of these problems admit a closed-form solution, the presence of incomplete or noisy input data can hinder this path. It is often more convenient to devise a common procedure that allows estimating the variables of interest with certainty. One particular approach is to devise a cost function $E(\mathbf{x}, \mathbf{Z})$ that capture the "fitness" of the estimate and, find an optimal solution by minimizing the sum of squared residuals $\mathbf{e}_k$ that changes based on the current state. The generic cost equation is

$$E(\mathbf{x}, \mathbf{Z}) = \sum_i \|\mathbf{e}_k(\mathbf{x})^T \mathbf{e}_k(\mathbf{x})\|_{\mathbf{\Omega}} = \sum_i \|\mathbf{e}_k(\mathbf{x})\|^2_{\mathbf{\Omega}}, \tag{2.40}$$

where $\mathbf{\Omega}$ is a symmetric positive-definite matrix and $\|\cdot\|_{\mathbf{\Omega}}$ denotes the Mahalanobis norm

$$\|\mathbf{x}\|_{\mathbf{\Omega}} = \sqrt{\mathbf{x}^T \mathbf{\Omega} \mathbf{x}}. \tag{2.41}$$

The Mahalanobis norm incorporates measurement uncertainty through the information matrix $\mathbf{\Omega}$. When $\mathbf{\Omega}$ is available (i.e., as the inverse of the measurement covariance), it allows down-weighting the noisier components of the measurements, leading to a statistically more consistent solution.

To be effective, the residual $\mathbf{e}_k$ should be minimal and have its minimum at the best state. Additionally, a strong requirement for the described approaches is that $\mathbf{e}_k$ should be differentiable with respect to $\mathbf{x}$.

When these conditions are satisfied, the optimal state $\mathbf{x}^*$ that minimizes $E(\mathbf{x}, \mathbf{Z})$ as

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} E(\mathbf{x}, \mathbf{Z}) \tag{2.42}$$

can be computed iteratively. In the next sections, several iterative optimization methods will be presented. Finally, Sec. 2.4.3 describes how to handle variables that live on manifolds.

2.4.1 Gradient Descent

The classic analogy used to describe Gradient Descent (GD) is as follows: "Suppose you are stuck in a foggy mountain, and you're trying to get back down. You can exploit the local information about the ground (make little steps around your current position) to find the fastest slope down. If you follow this direction, you will decrease your elevation and getting closer to safety. Still, you're not guaranteed that you will reach ground."

Since E is differentiable with respect to $\mathbf{x}$, a possible, first-order iterative solver to find an optimal state is GD. The idea is that, for each iteration k, we can exploit the gradient $\boldsymbol{\nabla}\mathbf{e}_k = \partial E_k/\partial\mathbf{x}$ to find the direction of steepest increase and, inverting it provides the direction of steepest descent. It follows that updating the state with

$$\mathbf{x}' = \mathbf{x} - \eta\boldsymbol{\nabla}\mathbf{e}_k, \tag{2.43}$$

with sufficiently small η, then guarantees that $\mathbf{e}_k(\mathbf{x}) > \mathbf{e}_k(\mathbf{x}')$. For certain conditions such that $\mathbf{e}_k$ is convex and $\boldsymbol{\nabla}\mathbf{e}_k$ is Lipschitz continuous, the GD guarantees convergence to a global minimum.

2.4.2 Gauss-Newton

Although easier, GD often requires serious consideration on the step size η, especially for situations in which the function has several local minima which can get the solver stuck. Another solution is the second-order Gauss-Newton (GN) solvers for Iterative Least-Squares (ILS) problems. These methods, iteratively optimize the cost function by looking for an infinitesimal $\boldsymbol{\Delta}\mathbf{x}$ that minimizes it. Since $\boldsymbol{\Delta}\mathbf{x}$ is infinitesimal, then the linear approximation with $\mathbf{J}_k = \partial\mathbf{e}_k/\partial\mathbf{x}$ holds

$$\mathbf{e}_k(\mathbf{x}+\boldsymbol{\Delta}\mathbf{x}) \approx \mathbf{e}_k(\mathbf{x}) + \mathbf{J}_k\boldsymbol{\Delta}\mathbf{x}. \tag{2.44}$$

By replacing Eq. (2.44) in Eq. (2.40), we obtain

$$\begin{aligned} E\left(\mathbf{x}+\boldsymbol{\Delta}\mathbf{x}\right) &= \sum_i \left\|\mathbf{e}_k(\mathbf{x}+\boldsymbol{\Delta}\mathbf{x})^T\mathbf{e}_k(\mathbf{x}+\boldsymbol{\Delta}\mathbf{x})\right\|^2_{\boldsymbol{\Omega}} \\ &\approx \sum_i \left\|\left(\mathbf{e}_k(\mathbf{x}) + \mathbf{J}_k\boldsymbol{\Delta}\mathbf{x}\right)^T\left(\mathbf{e}_k(\mathbf{x}) + \mathbf{J}\boldsymbol{\Delta}\mathbf{x}\right)\right\|^2_{\boldsymbol{\Omega}} \\ &= \sum_i \left\|\mathbf{e}_k(\mathbf{x})^T\mathbf{e}_k(\mathbf{x}) + 2\mathbf{e}_k(\mathbf{x})^T\mathbf{J}_k\boldsymbol{\Delta}\mathbf{x} + \boldsymbol{\Delta}\mathbf{x}^T\mathbf{J}_k^T\mathbf{J}_k\boldsymbol{\Delta}\mathbf{x}\right\|^2_{\boldsymbol{\Omega}} \\ &= \sum_i \boldsymbol{\Delta}\mathbf{x}^T\underbrace{\mathbf{J}_k^T\boldsymbol{\Omega}_k\mathbf{J}_k}_{\mathbf{H}}\boldsymbol{\Delta}\mathbf{x} + 2\underbrace{\mathbf{e}_k(\mathbf{x})^T\boldsymbol{\Omega}_k\mathbf{J}_k}_{\mathbf{b}^T}\boldsymbol{\Delta}\mathbf{x} + \underbrace{\mathbf{e}_k(\mathbf{x})^T\Omega\mathbf{e}_k(\mathbf{x})}_{\mathbf{c}} \end{aligned}. \tag{2.45}$$

This is a quadratic equation with respect to $\boldsymbol{\Delta}\mathbf{x}$ and can be minimized by finding the roots of its derivative $2\mathbf{b}^T + 2\left(\mathbf{H}\boldsymbol{\Delta}\mathbf{x}\right)^T = 0$ as

$$\begin{aligned} \mathbf{H}\boldsymbol{\Delta}\mathbf{x} &= -\mathbf{b} \\ \boldsymbol{\Delta}\mathbf{x} &= -\mathbf{H}^{-1}\mathbf{b}. \end{aligned} \tag{2.46}$$

Note that from Eq. (2.45), we derive that $2\mathbf{b}$ is equivalent to the gradient at point $\mathbf{x}$, while $\mathbf{H}$ is the GN approximation of the Hessian at $\mathbf{x}$.

Iterative Reweighted Least Squares: Sensor noise and bad data association can lead to "outliers" which shifts the local optimal solution to undesired areas. While consensus-based methods like RANSAC [35] remove these outliers by exploiting the idea that the inliers agree towards a common solution, ILS solvers over-account for outliers whose error is large. The root of this problem lies in the quadratic nature of the error term, therefore, a common solution is to solve a "damped" cost function that grows sub-quadratically with the error. By computing the $\mathcal{L}_1$ norm of the cost function and its derivative

$$\mathbf{u}_k(\mathbf{x}) = \sqrt{\mathbf{e}_k(\mathbf{x})^T \Omega_k \mathbf{e}_k(\mathbf{x})} \tag{2.47}$$

$$\frac{\partial \mathbf{u}_k(\mathbf{x})}{\mathbf{x}} = \frac{1}{\mathbf{u}_k(\mathbf{x})} \mathbf{e}_k(\mathbf{x})^T \mathbf{\Omega}_k \frac{\partial \mathbf{e}_k(\mathbf{x})}{\mathbf{x}}, \tag{2.48}$$

and by introducing a scalar weighting function $\rho(\mathbf{u})$, we obtain a generalized formulation

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i \rho\left(\mathbf{u}_k(\mathbf{x})\right). \tag{2.49}$$

Deriving Eq. (2.49) for $\mathbf{x}$ yields

$$\begin{aligned} \frac{\partial \rho\left(\mathbf{u}_k(\mathbf{x})\right)}{\partial \mathbf{x}} &= \left.\frac{\partial \rho\left(\mathbf{u}_k(\mathbf{x})\right)}{\mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}_k} \frac{\partial \mathbf{u}_k(\mathbf{x})}{\partial \mathbf{x}} \\ &= \left.\frac{\partial \rho\left(\mathbf{u}_k(\mathbf{x})\right)}{\mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}_k} \frac{1}{\mathbf{u}_k(\mathbf{x})} \mathbf{e}_k(\mathbf{x})^T \mathbf{\Omega}_k \frac{\partial \mathbf{e}_k(\mathbf{x})}{\mathbf{x}} \\ &= \gamma_k(\mathbf{x}) \mathbf{e}_k(\mathbf{x})^T \mathbf{\Omega}_k \frac{\partial \mathbf{e}_k(\mathbf{x})}{\partial \mathbf{x}} \end{aligned} \tag{2.50}$$

where

$$\gamma_k(\mathbf{x}) = \left.\frac{\partial \rho\left(\mathbf{u}_k(\mathbf{x})\right)}{\mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}_k} \frac{1}{\mathbf{u}_k(\mathbf{x})}. \tag{2.51}$$

Intuitively, the weighting function $\rho(\cdot)$ acts on the gradient by rescaling it with $\gamma_k(\mathbf{x})$. By comparing Eq. (2.50) with the gradient of Eq. (2.40)

$$\frac{\partial \|\mathbf{e}_k(\mathbf{x})\|^2_{\mathbf{\Omega}}}{\partial \mathbf{x}} = 2\mathbf{e}_k(\mathbf{x})^T \Omega_k \frac{\partial \mathbf{e}_k(\mathbf{x})}{\partial \mathbf{x}}, \tag{2.52}$$

we notice that the two differ only for the scalar term $\gamma_k(\mathbf{x})$. Furthermore, the term can be absorbed by the information matrix $\bar{\mathbf{\Omega}}_k$ as

$$\bar{\mathbf{\Omega}}_k(\mathbf{x}) = \gamma_k(\mathbf{x}) \mathbf{\Omega}_k, \tag{2.53}$$

and be propagated within the GN framework as described above. Through this formulation, based on the characteristics of the problem, a different weighting function may be employed to reduce the effect of outliers. Note that, the problem described in Eq. (2.40) can be obtained by using the weighting function

$$\rho(\mathbf{u}) = \frac{1}{2}\mathbf{u}^2. \tag{2.54}$$

Levenberg-Marquardt

Differently from the GD method, GN provides an update step based on a linearization of the local space. This does not guarantee to land in a state with a lower cost, especially for highly non-linear functions. Based on the state and cost function, this scenario may lead to state oscillations at best and divergence in worst cases. The Levenberg-Marquardt (LM) method takes the best of the two previously seen algorithms to provide fast, yet monotonically decreasing updates.

The intuition behind LM is that, when possible, GN should be used as it provides the best convergence. However, where GN fails, then we should progressively rely on a GD technique. Let $\lambda \geq 0$ be a damping factor, then the LM method defines the update step as

$$\boldsymbol{\Delta}\mathbf{x} = -\left(\mathbf{H} + \lambda\mathbf{I}\right)^{-1}\mathbf{b}. \tag{2.55}$$

If $\lambda = 0$, then Eq. (2.55) coincides with Eq. (2.46), while for $\lim_{\lambda\to\inf}$, the influence of $\mathbf{H}$ comes marginal and the update becomes $\boldsymbol{\Delta}\mathbf{x} \approx -\left(\lambda\mathbf{I}\right)^{-1}\mathbf{b} = -1/\lambda\mathbf{b}$. Since $\mathbf{b}$ is a scaled version of the cost gradient, the update step here follows the damped inverse direction of the gradient, mirroring the behavior of GD. LM is typically initialized with a low value of λ to immediately rely on GN updates. Each time an update step leads to a better state, λ is halved. If a new step leads to a bad spot, the step is retracted and lambda is doubled, favoring GD updates. This combination of approaches ensures that LM always converge to either a local minimum or to a saddle point.

2.4.3 Manifolds

In previous sections, we assumed the state to be represented by Euclidean entities. However, in the field of computer vision for instance, we are often interested at estimating a state containing one or more rotation matrices in $\mathbb{SO}(3)$ or unit vectors in $\mathbb{S}^2$ that lives on manifolds. Manifolds are curved spaces that locally resemble Euclidean space. An intuitive manifold is the Earth itself; We can locally treat the space as if it was Euclidean (i.e., using the Euclidean distance to measure the distance between two trees), however, globally the same rules do not apply (i.e., using the Euclidean distance to measure the distance between Rome and Tokyo). During the optimization phase, neither LM, GN nor GD handle the constraints required to maintain an element within its manifold. This section presents the mathematical tools required to integrate the manifolds within the ILS optimization frameworks.

A Lie group $\mathcal{G}$ admit a smooth[5] manifold $\mathcal{M}$ and a composition operation $(\mathcal{G}, \circ)$ such that

$$\begin{aligned} \text{Closure under ``}\circ\text{''} &: \mathcal{X} \circ \mathcal{Y} \in \mathcal{G} \\ \text{Identity } \mathcal{E} &: \mathcal{E} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{E} = \mathcal{X} \\ \text{Inverse } \mathcal{X}^{-1} &: \mathcal{X}^{-1} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{X}^{-1} = \mathcal{E} \\ \text{Associativity} &: (\mathcal{X} \circ \mathcal{Y}) \circ \mathcal{Z} = \mathcal{X} \circ (\mathcal{Y} \circ \mathcal{Z}). \end{aligned} \tag{2.56}$$

Since the space is smooth, it always admits a unique tangent space at each point, same everywhere and called *Lie algebra* $\mathfrak{m}$. The Lie algebra is a vector space and as such, its elements can be identified as vectors in $\mathbb{R}^m$ whose dimension m equals to the degrees of

[5]Topological space that locally resembles Euclidean space.

freedom of $\mathcal{M}$. To express a Cartesian vector $\boldsymbol{\tau} \in \mathbb{R}^M$ in the Lie algebra, let's define the *Hat* and *Vee* isomorphism operators

$$\begin{aligned} &\text{Hat}: \quad \mathbb{R}^M \to \mathfrak{m}; \quad \boldsymbol{\tau} \to \boldsymbol{\tau}^\wedge = \textstyle\sum_{i=1}^{M} \tau_i E_i \\ &\text{Vee}: \quad \mathfrak{m} \to \mathbb{R}^M; \quad \boldsymbol{\tau}^\wedge \to (\boldsymbol{\tau}^\wedge)^\vee = \boldsymbol{\tau} = \textstyle\sum_{i=1}^{M} \tau_i \mathbf{e}_i, \end{aligned} \tag{2.57}$$

where E_i are base elements or generators of $\mathfrak{m}$ and $\mathbf{e}_i$ are the corresponding vector form of $\mathbb{R}^M$ (such that $\mathbf{e}_i^\wedge = E_i$).

Let $\exp : \mathbb{R}^M \to \mathcal{M}$ be the operator that maps elements of the Lie algebra to the group and $\log : \mathcal{M} \to \mathbb{R}^M$ be the inverse operator

$$\mathcal{X} \triangleq \exp\left(\boldsymbol{\tau}^\wedge\right) \tag{2.58}$$

$$\boldsymbol{\tau} \triangleq \log(\mathcal{X})^\vee. \tag{2.59}$$

Finally, we can introduce the operators for addition $\boxplus$ and subtraction $\boxminus$ which allows us to formulate ILS problems with manifold updates. These operators are combination of a exp/log operator with one composition and, due to the commutativity of the latter, they are defined in right- and left- versions depending on the order of the operations. The right-operators are defined as:

$$\text{right-}\boxplus: \qquad \mathcal{Y} = \mathcal{X} \boxplus \boldsymbol{\tau} \triangleq \mathcal{X} \exp\left(\boldsymbol{\tau}^\wedge\right) \in \mathcal{M} \tag{2.60}$$

$$\text{right-}\boxminus: \qquad \boldsymbol{\tau} = \mathcal{Y} \boxminus \mathcal{X} \triangleq \log(\mathcal{X}^{-1} \circ \mathcal{Y})^\vee \in \mathbb{R}^M. \tag{2.61}$$

Conversely, the left- operators are defined as

$$\text{left-}\boxplus: \qquad \mathcal{Y} = \boldsymbol{\tau} \boxplus \mathcal{X} \triangleq \exp\left(\boldsymbol{\tau}^\wedge\right) \mathcal{X} \in \mathcal{M} \tag{2.62}$$

$$\text{left-}\boxminus: \qquad \boldsymbol{\tau} = \mathcal{Y} \boxminus \mathcal{X} \triangleq \log(\mathcal{Y}^{-1} \circ \mathcal{X})^\vee \in \mathbb{R}^M. \tag{2.63}$$

Due to the ambiguity of the notation and, provided that there are several differences that won't be discussed in this thesis but can be found in [45, 116], each chapter will provide explicitly a sub-definition of such operators.

We can now reformulate the linearization of GN in Eq. (2.44) for elements of the manifold $\hat{\mathbf{X}} \in \mathcal{M}$ as

$$\mathbf{e}_k(\hat{\mathbf{X}} \boxplus \boldsymbol{\Delta}\mathbf{x}) \approx \mathbf{e}_k(\hat{\mathbf{X}}) + \left.\frac{\partial \mathbf{e}_k(\hat{\mathbf{X}} \boxplus \boldsymbol{\Delta}\mathbf{x})}{\partial \boldsymbol{\Delta}\mathbf{x}}\right|_{\boldsymbol{\Delta}\mathbf{x}=0} \boldsymbol{\Delta}\mathbf{x}. \tag{2.64}$$

Similarly, after computing the update step, it can be added to the state as

$$\hat{\mathbf{X}} \leftarrow \hat{\mathbf{X}} \boxplus \boldsymbol{\Delta}\mathbf{x}. \tag{2.65}$$

In some cases, even the error function might lie on a manifold (i.e., pose-pose distance), therefore a local $\boxminus$ operator is also required to compute it as

$$\mathbf{e}_k(\mathbf{X}) = \mathbf{h}(\mathbf{X}) \boxminus \mathbf{Z}_k, \tag{2.66}$$

where $\mathbf{h}$ is a function of the state.

2.5 Inverse Rendering

A well known topic in the field of Computer Graphics is *rendering*, namely producing photorealistic images given a camera pose, a set of geometries, materials, and lights. Photorealistic rendering is solved by simulating light transport[6] in a 3D scene. Light transport is well modeled by the rendering equation denoted as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\omega}_o) = \mathcal{L}_e(\mathbf{x}, \boldsymbol{\omega}_o) + \int_{\mathcal{S}^2} f_r\left(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o\right) \mathcal{L}(\mathbf{x}, \boldsymbol{\omega}_i)(\boldsymbol{\omega}_i^T \mathbf{n}) d\boldsymbol{\omega}_i, \tag{2.67}$$

where $\mathcal{L}$ is the intensity of light measured at position $\mathbf{x}$ and coming from direction $\boldsymbol{\omega}_o$, $\mathcal{L}_e(\mathbf{x}, \boldsymbol{\omega}_o)$ is the emitted light from $\mathbf{x}$ towards $\boldsymbol{\omega}_o$, and $\mathbf{n}$ is the surface normal at $\mathbf{x}$. The surface integral $\int_{\mathcal{S}^2}$ requires the evaluation of directions in all the hemisphere $\mathcal{S}^2$.

The function $f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ is typically called Bidirectional Reflectance Distribution Function (BRDF)[7]and it models how light reflects from different types of materials.

Solving Eq. (2.67) is hard for two reasons. First, to estimate the light coming out at $\mathbf{x}$ from the direction $\boldsymbol{\omega}_o$, we need to integrate all the incoming light from all other possible directions in $\mathcal{S}^2$. Secondly, computing the incoming light from a direction $\boldsymbol{\omega}_i$, means recursively solving $\mathcal{L}(\mathbf{x}, \boldsymbol{\omega}_i)$ which depends on itself from other points and directions.

The recursive nature of the function hints that light should be resolved globally across the entire scene which is currently plausible only with ray-tracers or path-tracers. These Monte Carlo (MC) techniques are able to resolve Eq. (2.67) by efficiently sampling the integral and through several tricks that goes beyond the scope of this thesis.

Furthermore, *Inverse Rendering* is the inverse process of estimating the geometry, materials, and lights of a scene given one or many posed images. Intuitively, being able to estimate the 3D shape and appearance from just 2D images is of extreme interest to solve and ease many downstream tasks (i.e., Robotics, simulation, AR/VR, etc.) Still, robust and scalable joint estimation for geometry, material, and lighting is still not feasible, thus different methodologies have been developed to solve partial problems. Intuitively, a mandatory step is to establish a common reference frame to localize each camera. This task is impactful and is resolved through Structure from Motion (SfM) pipelines [105].

Furthermore, assuming no scene's light and ideal Lambertian surfaces, and no view-dependent surface appearance, Multi View Stereo (MVS) methods leverage per-pixel appearance correspondences to infer the geometry of the scene [108]. These approaches are very accurate at modeling textured[8] Lambertian surfaces and, as expected, tends to break in specific conditions: The presence of strong lights within the scene, or non-Lambertian surfaces creates strong view-dependent appearance changes within the scene, which hinders the process of data association and ultimately, of reconstruction in those areas.

In recent years, the use of *differentiable* rendering pipelines has emerged as an alternative to MVS. As the name implies, a differentiable renderer $f\left(\boldsymbol{\theta}, \mathbf{x}\right) \rightarrow \mathcal{I}$ allows computing an image $\mathcal{I}$ from a scene $\boldsymbol{\theta}$ and from position $\mathbf{x}$ while allowing the computation of the gradients $\partial f(\boldsymbol{\theta}, \mathbf{x}) / \partial \boldsymbol{\theta}$. By selecting an effective scene representation, one can tackle the inverse

[6]How light bounce and interacts with surfaces and materials.

[7]We omit the Bidirectional Scattering Distribution Function (BSDF) as we assume to handle only surfaces.

[8]Texture-less areas won't be mentioned directly here, since these surfaces are challenging for all the presented methodologies.

rendering problem as

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}\left(f(\boldsymbol{\theta}, \mathbf{x}_i), \mathcal{I}_i\right), \tag{2.68}$$

where $\mathcal{L}$ is an arbitrary cost function that depends on the differences between the synthetic image $f(\boldsymbol{\theta}, \mathbf{x}_i)$ and the measured one $\mathcal{I}$.

Current established methods that rely on differentiable rendering are able to either capture only the appearance of a scene [81, 61], both appearance and geometry [136, 46] or even material properties [143, 70]. Choosing an effective representation is key to achieve high quality results. Specifically, the rendering bottleneck for differentiability is the *visibility* property of a specific primitive. Imagine a scenario of triangle rasterization in which one primitive is partially occluding another. Per-pixel visibility inherently introduces non-continuity within the rendering function due to the sharp boundaries of the primitive itself[9]. Moreover, modern differentiable renderers are based on either one of these principles:

- **Volumetric representation**. Instead of relying on explicit primitives, let the scene be a continuous field for density and appearance. Using the notions described in Sec. 2.5.1, visibility and occlusion are straightforwardly differentiable.

- **Soft Rasterization.** Rather than defining hard boundaries, each pixel integrates data from neighboring primitives without introducing discontinuities, ensuring that gradients can flow. An example of soft rasterization is shown in Sec. 2.5.2.

2.5.1 Volumetric Rendering

A volumetric scene is a function $\mathcal{V}(\mathbf{x}, \boldsymbol{\omega})$ that maps, for each point $\mathbf{x} \in \mathbb{R}^3$ and for each view-direction $\boldsymbol{\omega} \in \mathbb{S}^2$, density $\sigma(\mathbf{x}) \in \mathbb{R}$ and view-dependent appearance $\mathbf{c}(\mathbf{x}, \boldsymbol{\omega}) \in \mathbb{R}^3$. As described in Sec. 2.4.3, we can map $\boldsymbol{\omega}$ using the lie algebra of $\mathbb{S}^2$ whose minimal representation is a 2D vector of angles $\hat{\boldsymbol{\omega}} \in \mathbb{R}^2$, therefore we model $\mathcal{V}$ as

$$\mathcal{V} : \mathbb{R}^5 \to \mathbb{R}^4. \tag{2.69}$$

To introduce how inverse rendering is performed, we need to explain how to traverse and measure volumetric data. Interpreting volumes from a probabilistic perspective offers a clear approach for sampling.

Let $\mathbf{r}(t) = \mathbf{x} + \hat{\boldsymbol{\omega}} t \ : \ t \in \mathbb{R}^+$ be a ray cast from its origin $\mathbf{x}$ and direction $\hat{\boldsymbol{\omega}}$ parametrized by t, and let $\sigma(t)$ represent the density of the volume at $\mathbf{r}(t)$. Suppose the density $\sigma(t)$ to quantify the infinitesimal number of particles that occupy the space at $\mathbf{r}(t)$. Moreover, we assume the ray to continue indefinitely within $\mathcal{V}$ unless it *hits* a particle. We can define this probability as

$$P\left(\text{hit at } t\right) = \sigma(t)dt \tag{2.70}$$

The probability is still incomplete because, we also need to consider that, to hit something at t, the ray should have been able to traverse the volume up to that point. Let

$$P\left(\text{no hits before } t\right) = T(t)dt, \tag{2.71}$$

[9] Each edge of the primitive (i.e., triangle) acts like a step function for the visibility.

where $T(t)$ is the volume's *transmittance*. Using Eq. (2.70) and Eq. (2.71) we can compute the probability of not hitting particles at $t + dt$ as

$$\begin{aligned} P\left(\text{no hits before } \ t+dt\right) &= P\left(\text{no hits before } \ t\right)\left(1 - P\left(\text{hit at } \ t\right)\right) \\ T(t+dt) &= T(t)(1-\sigma(t)dt). \end{aligned} \tag{2.72}$$

We further rearrange the terms and obtain a solution for $T(t)$ as

$$\begin{aligned} T(t) + T'(t)dt &= T(t) - T(t)\sigma(t)dt \\ \frac{T'(t)}{T(t)}dt &= -\sigma(t)dt \\ \log\left(T(t)\right) &= \int_{t_0}^{t} -\sigma(s)ds \\ T(t) &= \exp\left(-\int_{t_0}^{t}\sigma(s)ds\right). \end{aligned} \tag{2.73}$$

Since the integral in Eq. (2.73) can be solved numerically via quadrature, we can use α-blending to process samples from front to back. Specifically, let

$$\alpha(t_i) = (1 - \exp(\sigma(t_i))) \tag{2.74}$$

be the "opacity" of the volume at $\mathbf{r}(t_i)$, and

$$T(t_i) = \prod_{j=1}^{i-1}\left(1 - \alpha(t_j)\right) \tag{2.75}$$

be the cumulated volume transmittance for reaching t_i. Then, the cross-volume quantities such as view-dependent color appearance $\mathbf{c}(\boldsymbol{\omega}, \mathbf{x})$ can be computed by traversing samples and computing

$$\mathbf{c}(\boldsymbol{\omega}, \mathbf{x}) = \sum_{i=1}^{K}\alpha(t_i)\hat{\mathbf{c}}(t_i)T(t_i), \tag{2.76}$$

where K is the number of samples and $\hat{\mathbf{c}}(t)$ is the color property of the volume at t.

Having to sample multiple points per ray means that volumetric ray marching is a costly operation, especially for high pixel counts, meaning that sampling strategy plays a fundamental role in efficiency. For instance, a coarse sampling might skip relevant high density regions, while, on the other hand, dense sampling may lose too much time in integrating empty or almost empty regions.

A question that may arise is, how the volume $\mathcal{V}$ can be implemented efficiently?

Classic approaches leverage the limited sensor resolution to discretize the space into "voxels". Within each voxel, both density, color, or additional properties may be stored to model the underlying continuous field. One issue with voxel grids is that its memory constraints are $O(S^3)$ since they need to cover the whole scene. Given that within typical applications, most of the space is empty, efficient grids are only allocated near the surfaces, dropping the memory constraints to $O(S^2)$ [85, 89]. Additionally, since the required resolution for each voxel may change within the same scene[10], several approaches rely on

[10]Flat surfaces can be represented with larger voxels, while detailed regions may require smaller ones.

recursive space-partitioning structures to allocate them [85]. Chapter 8 contains a thorough analysis of Sparse Voxel Grid (SVO) representations.

In 2020, Neural Radiance Field (NeRF) [80] presented a methodology that relies on a single Multi-Layer Perceptron (MLP) to approximate the density and appearance volumetric fields. Since Eq. (2.76) is fully differentiable with respect to α and $\hat{\mathbf{c}}$, NeRF follows a gradient-based optimization approach that uses input posed images to train the MLP parameters. Specifically, each pixel of each image is back-projected and, samples are adaptively selected within the corresponding ray to capture high density regions. Then, each sample $\{\mathbf{x}_i, \boldsymbol{\omega}_i\}$ is used to query the volume and, via α-blending, synthetic images are generated. For each image, an error map is generated by computing the photometric error between the synthetic and input images and, via GD, the MLP are trained to minimize it. Remarkably, NeRFs representation is also extremely compact with respect to SVO while also yielding high fidelity results[11].

2.5.2 Gaussian Splatting

One major issue with NeRF is that, for each pixel of each training image, hundreds of queries are needed to optimize the underlying MLP to produce the synthetic counterparts. This translates in high computational demands for training and inference[12]. In 2023, Kerbl *et al.* [61] presented another solution for NVS that relies on "soft" primitives to achieve the same visual quality of NeRFs with much higher frame-rate[13]. To achieve this, the authors relied on two components:

- 3D Gaussian primitives.
- A tile-based differentiable rasterizer.

This section is going to describe these components thoroughly.

3D Gaussians: As mentioned in Sec. 2.5, differentiable triangle-based rasterization poses challenges due to the sharp boundaries on the edges of the primitives. One solution is to relax the bounds of the primitives to achieve a smoother transition. Additionally, triangles are defined by their vertices and from an optimization perspective, constraining the connectivity between neighboring triangles is non-trivial. 3D Gaussians offer a flexible and powerful alternative, especially for NVS. A 3D Gaussian is a spatial blob defined by a 3D covariance $\boldsymbol{\Sigma}_i \in \mathbb{R}^{3\times3}$ centered at $\boldsymbol{\mu}_i \in \mathbb{R}^3$, an opacity parameter o_i, and color $\bar{\mathbf{c}}_i(\boldsymbol{\omega})$[14]. Formally it is defined as

$$\mathcal{G}(\mathbf{x}) = \exp\left(-\frac{1}{2}\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}\right). \tag{2.77}$$

The primitive is rasterized via local affine approximation of the projective transformation $\boldsymbol{\pi}_c$, hence the 2D image-space covariance $\boldsymbol{\Sigma}'$ is computed as

$$\boldsymbol{\Sigma}' = \mathbf{J}^c\mathbf{T}_w\boldsymbol{\Sigma}^c\mathbf{T}_w^T\mathbf{J}^T, \tag{2.78}$$

where $\mathbf{J} = \partial\boldsymbol{\pi}_c(\mathbf{x})/\partial\mathbf{x}$. During the optimization phase, we can expect to handle the partial derivative $\partial\mathcal{L}/\partial\boldsymbol{\Sigma}$, however, following the optimization strategy of GD, could lead the

[11] At least for Novel View Synthesis (NVS).

[12] Often requiring ≥ 1 days to converge and up to $1-2$ Hz in inference.

[13] Over ≥ 60 Hz on 4K resolution.

[14] The color is parametrized using Spherical Harmonics to model view-dependent changes.

new $\mathbf{\Sigma}$ to violate its constraints[15]. The solution is to first decompose $\mathbf{\Sigma}$ into the equivalent representation

$$\mathbf{\Sigma} = \mathbf{R}\mathbf{S}\mathbf{S}^T\mathbf{R}^T, \tag{2.79}$$

where $\mathbf{R}$ is a rotation matrix and $\mathbf{S} = \text{diag}(s_x, s_y, s_z)$ is a diagonal matrix. These are further stored as a scaling vector $\mathbf{s} \in \mathbb{R}^{3+}$ and a quaternion $\mathbf{q}$, which is kept normalized at each optimization step. To summarize, a 3D Gaussian is parametrized as $\mathbf{G}_i = (\boldsymbol{\mu}_i, \mathbf{q}_i, \mathbf{s}_i, o_i, \bar{\mathbf{c}})$.

Differentiable Rasterizer: The task of the rasterizer is to produce an image given a camera view matrix ${}^c\mathbf{T}_w$, and a set of Gaussians $\mathcal{G} = \{\mathbf{G}_i\}_{i=1}^M$. Since Gaussians are semi-transparent primitives, they can be combined via α-blending. As outlined earlier, α-blending is a per-pixel process carried out in a front-to-back manner, requiring each pixel to itendify the set of contributing Gaussians as well as their depth ordering. To accelerate the process, instead of sorting Gaussians for each pixel, windows of 16×16 pixels are grouped and assigned the same set of ordered Gaussians. Since the full rasterization is carried on a GPU, the rasterizer is designed to be extremely parallel

After computing the image covariance $\mathbf{\Sigma}'_i$ centered at $\boldsymbol{\pi}_c(\boldsymbol{\mu}_i)$, the rasterizer generates, for each tile occupied by the Gaussian, a key-value pair. The 64-bit key $k_{i,j}$ is computed by combining the view-space depth $\boldsymbol{\mu}_{i,z} \in \mathbb{R}$ and the tile ID $t_j \in \mathbb{N}$ as

$$k_{i,j} = t_j \ll 32 \mid \text{int}(\boldsymbol{\mu}_{i,z}), \tag{2.80}$$

where $\ll$ represents the left bit-shift operation, while the corresponding value coincides with the Gaussian ID. After processing every primitive, key-value pairs are sorted via GPU Radix sort[16]. Moreover, since each tile is independent of one another, they can be processed in parallel to maximize throughput. The pixel $\mathbf{u}$ color $\mathbf{c}(\mathbf{u})$ is achieved by computing per-Gaussian opacity

$$\alpha(\mathbf{u}, i) = o_i \mathcal{G}_i(\mathbf{u}), \tag{2.81}$$

and view-dependent color contributes

$$\mathbf{c}_i({}^c\mathbf{T}_w) = \bar{\mathbf{c}}\left(\frac{\boldsymbol{\mu}_i - {}^c\mathbf{t}_w}{\|\boldsymbol{\mu}_i - {}^c\mathbf{t}_w\|}\right), \tag{2.82}$$

where ${}^c\mathbf{t}_w$ is the translation component of the transform. The final color is obtained as in Eq. (2.76). Since the whole process is once again differentiable, Gaussians can be optimized by minimizing the photometric error between synthetic and input images.

Adaptive Density Control: Differently from NeRFs, 3DGS may end in cases where, the results cannot improve because there are not enough primitives available. Conversely, it may also happen that one large Gaussian is over-reconstructing a portion of the scene, once again limiting the quality of the result. In both cases, per-Gaussian view-space positional gradients tend to remain high due to their positional instability[17]. Moreover, at every densification iteration, small Gaussians that are selected are duplicated, while larger Gaussians are split into two smaller ones. To limit the growth of primitives over time, opacity is reset every few hundred iterations. This guarantees that only a subset of Gaussians is going to become opaque, while the others are further pruned away.

[15] Symmetry and semi-positivity.

[16] The key defined in Eq. (2.80) allows to naturally sort keys such that, if multiple Gaussians affect the same tile, post sorting, the keys corresponding to that tile will be close and actually sorted by Gaussian depth. Additionally, through values, the corresponding Gaussian ID can be recovered.

[17] Since those areas are either under or over reconstructed, the optimizer keeps moving the primitives to find local minimums.

2D Gaussian Splatting

Albeit successful and very accurate for NVS, 3D Gaussian splatting suffers at modeling accurate scene geometries. Especially from planar surfaces, 3D Gaussian blobs faces several challenges. First, when one of the 3D Gaussian scaling parameters s_i tends to 0, the affine approximation method for projecting the covariance matrix becomes unstable and can lead to artifacts. Moreover, 3DGS does not naturally model surface normals and depth maps, both mandatory for high-quality surface reconstruction.

In 2D Gaussian Splatting [51], the Gaussian primitives are altered to be naturally expressed within a planar surface in $\mathbb{R}^3$. Specifically, the authors provide an alternative formulation for projecting the covariance $\mathbf{\Sigma}'$ and sampling the Gaussian when its third scaling parameter is by construction set to 0.

Let $\mathbf{R} = (\mathbf{t}_\alpha, \mathbf{t}_\beta, \mathbf{t}_n)$ and $\mathbf{S} = \text{diag}\,(s_\alpha, s_\beta, 0)$ be the rotation and scaling matrices. If the primitives live within a plane, let $(\mathbf{t}_\alpha, \mathbf{t}_\beta)$ be a basis for that space, so that, the Gaussian can be evaluated within this reference frame as:

$$\mathcal{G}_{2d}(\alpha, \beta) = \exp\left(-\frac{1}{2}(\alpha^2 + \beta^2)\right) \tag{2.83}$$

Let $\mathbf{H}$ be the mapping between the splat space and the world define as

$$\mathbf{H} = \begin{pmatrix} s_\alpha \mathbf{t}_\alpha & s_\beta \mathbf{t}_\beta & \mathbf{0} & \boldsymbol{\mu} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{RS} & \boldsymbol{\mu} \\ \mathbf{0} & 1 \end{pmatrix}. \tag{2.84}$$

Instead of projecting the covariance on the image plane, the authors employ a ray-splat intersection schema based on the intersection of three non-parallel planes, initially described in [113]. Let $\mathbf{u} = (u, v)$ be an image coordinate parametrized as the intersection of the homogeneous plane $\mathbf{h}_u = (-1, 0, 0, u)^T$ and $\mathbf{h}_v = (0, -1, 0v)^T$. Using Eq. (2.84), let $\mathbf{h}_\alpha(\mathbf{u}) = ({}^c\mathbf{T}_w\mathbf{H})^T\mathbf{h}_u$ and $\mathbf{h}_\beta = ({}^c\mathbf{T}_w\mathbf{H})^T\mathbf{h}_v$ be the planes expressed in the splat's reference frame[18]. A point on the 2D Gaussian plane can be represented as $(\alpha, \beta, 1, 1)$, therefore, a ray-splat intersection occurs if

$$\mathbf{h}_\alpha(\mathbf{u})\,(\alpha, \beta, 1, 1)^T = \mathbf{h}_\beta(\mathbf{u})\,(\alpha, \beta, 1, 1)^T = 0, \tag{2.85}$$

which leads to an efficient solution for the intersection point $(\hat{\alpha}, \hat{\beta})$:

$$\hat{\alpha} = \frac{\mathbf{h}_{\alpha,2}\mathbf{h}_{\beta,4} - \mathbf{h}_{\alpha,4}\mathbf{h}_{\beta,2}}{\mathbf{h}_{\alpha,1}\mathbf{h}_{\beta,2} - \mathbf{h}_{\alpha,2}\mathbf{h}_{\beta,1}} \quad \hat{\beta} = \frac{\mathbf{h}_{\alpha,4}\mathbf{h}_{\beta,1} - \mathbf{h}_{\alpha,1}\mathbf{h}_{\beta,4}}{\mathbf{h}_{\alpha,1}\mathbf{h}_{\beta,2} - \mathbf{h}_{\alpha,2}\mathbf{h}_{\beta,1}}. \tag{2.86}$$

Moreover, the same formulation can be solved in the opposite direction to obtain an accurate bounding box of the splat in image-space. Another good property is that, once the intersection point $(\hat{\alpha}, \hat{\beta})$ is known, an accurate depth measurement can be done as

$$d_i(\mathbf{u}) = \left(({}^c\mathbf{T}_w\mathbf{H}_i)\left(\hat{\alpha}(\mathbf{u}), \hat{\beta}(\mathbf{u}), 1, 1\right)^T\right)_z. \tag{2.87}$$

Having planar splats with accurate depths, enable accurate depth and normal maps as

$$\mathbf{D}(\mathbf{u}) = \sum_{i=1}^{K} \alpha_i d_i(\mathbf{u}) T_i \tag{2.88}$$

$$\mathbf{N}(\mathbf{u}) = \sum_{i=1}^{K} \alpha_i\,({}^c\mathbf{R}_w\mathbf{t}_{n,i})\,T_i. \tag{2.89}$$

[18] $\mathbf{H}$ is pre-multiplied by ${}^c\mathbf{T}_w$ to map camera points in splat reference, while the transpose operation is equivalent to transforming points on a plane given the transformation matrix ${}^c\mathbf{T}_w\mathbf{H}$ [9].

Another difference with respect to 3DGS is found in the training stage for the representation. To enforce better surfaces, two additional regularization terms are included.

Depth Distortion: A pixel depth $d_i(\mathbf{u})$ can be generated by an arbitrary combination of weights and intersection distances. This is obviously not ideal for surface reconstruction as we, ideally, would like to have contributing splats spatially close. Inspired by [5], the regularizer guides the splats to concentrate in space, ensuring better surface alignment. The regularization term is defined as

$$\mathcal{L}_{dd} = \sum_{i,j} w_i w_j |d_i - d_j|, \tag{2.90}$$

where $w_i = \alpha_i T_i$ and $w_j = \alpha_j T_j$ be the blending weights for the i-th and j-th intersections. Intuitively, this regularizer act as a spring between the intersection points, trying to pull them closer to match the nature of a surface.

Normal Consistency: To correctly model a surface, the splat's normals should be aligned with the surface normal. The latter can be estimated directly from the depth-map $\mathbf{D}$ by computing the gradient with finite differences as

$$\hat{\mathbf{N}}(\mathbf{u}) = \frac{\boldsymbol{\nabla}_u \mathbf{p}_s \times \boldsymbol{\nabla}_v \mathbf{p}_s}{\|\boldsymbol{\nabla}_u \mathbf{p}_s \times \boldsymbol{\nabla}_v \mathbf{p}_s\|}, \tag{2.91}$$

where $\mathbf{p}_s = \boldsymbol{\pi}_c^{-1}(\mathbf{u}) \cdot d(\mathbf{u})$ is the estimated back-projected surface point. The regularizer defined as

$$\mathcal{L}_n = \sum_{\mathbf{u}} \left(1 - \mathbf{N}(\mathbf{u})^T \hat{\mathbf{N}}(\mathbf{u})\right), \tag{2.92}$$

guides the splat's normals towards the surface ones, ensuring a well-constrained problem for surface reconstruction.

2.6 Summary

In summary, this chapter has introduced the fundamental concepts and mathematical tools that serve as the foundation for the remaider of this thesis. We first reviewed the principles required to model spatial relationships, followed by an overview of the RGB, RGB-D, and LiDAR sensing modalities. Moreover, we discussed the optimization frameworks that underpin the estimation problems addressed in later chapters. Finally, we provided an introduction to inverse rendering and the related methodologies.

Building on this foundation, Chap. 3 focuses on a fundamental problem for multi-sensor fusion.

Chapter 3

Ca^{2}Lib: Simple and Accurate LiDAR-RGB Calibration using Small Common Markers

3.1 Introduction

The first work presented in this thesis concerns the extrinsic[1] calibration of LiDAR-camera pairs, which is a major requirement for sensor fusion. In this context, the challenge stems from the need to align and synchronize data streams with different modalities. In this chapter, we focus on estimating the extrinsics of the two sensors using cues extracted from their measurements. Currently, LiDAR-RGB calibration can be solved using a calibration target or marker, as done already for uni-modal sensor extrinsic estimation (i.e., multi-RGB [11]). The calibration target represents a unique object whose geometric and visual properties are known and, through tailored detection algorithms, can be measured precisely. Identifying common elements from both viewpoints is typically sufficient for determining spatial correlations between sensors. However, discerning which elements are relevant in this scenario remains problematic. Due to the sensors' varying resolutions and visual patterns, traditional markers such as checkerboard corners are not viable. This necessitates the exploration of more complex features. One example could be the use of holes of known dimensions in the target: LiDARs would be able to infer the center of the hole by measuring the points on the border, while cameras could estimate the same point using typical circle detection algorithms [53]. Although these features are proven effective for extrinsic estimation, the requirement of ad-hoc calibration targets poses more practical and often economic problems (these specific markers are often realized using CNC printers). Moreover, calibration is typically carried out directly on-site, so the target's portability and size are other problems that must be considered.

This chapter introduces a method for extracting robust planar features readily obtainable from sensor data. We relax the requirements for calibration targets to typical patterns already used for RGB calibrations (i.e., Checkerboards or ChAruCO [39]) of sizes down to A3/A4 dimensions for portability. Leveraging a direct non-linear formulation, we can achieve a highly accurate relative pose estimate even with a minimum of three observations per sensor.

[1] Relative position and orientation.

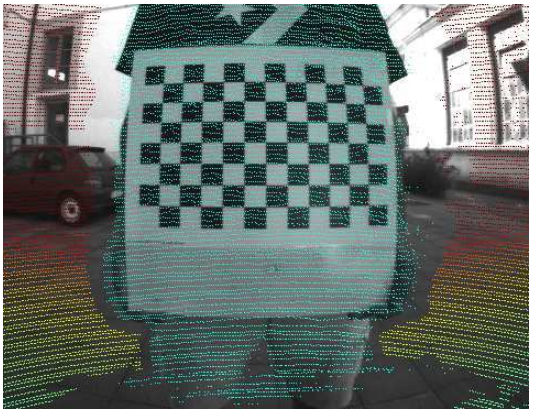


Figure 3.1. Qualitative LiDAR projection onto a fisheye RGB-camera. Shadows are caused by the parallax between the sensors.

3.2 Related Works

This section delves into LiDAR-RGB calibration and explores the two main classes of approaches: *target-based* and *target-less* calibration. As the name suggests, target-based approaches require the user to place artificial markers that both the camera and LiDAR can easily detect. This contrasts with target-less methods that free the user from this task. The core idea of calibration is common in the two approaches: computing common features between heterogeneous measurements and estimating the transformation that minimizes the distance between corresponding features.

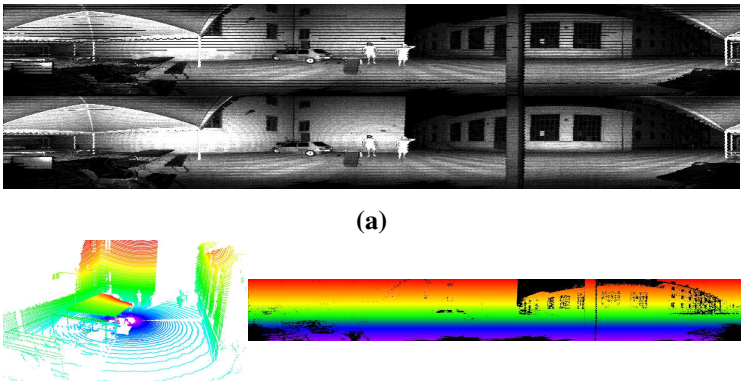
Target-less methods: Pandey *et al.* [94] presents an automatic data-driven approach based upon the maximization of mutual information between the sensor-measured surface intensities. Using different calibration parameters, the authors exploit the correlation coefficient for many scan-image pairs' reflectivity and intensity values. However, shadows of objects or colored surfaces that completely absorb infrared lights might result in a weaker correlation between scan-image pairs. Yoon *et al.* [148] proposes a calibration method using region-based object pose estimation. Objects are segmented in both measurements, and then a 3D mesh is generated from the LiDAR measurements, while images are used to reconstruct the scene using Structure from Motion (SfM). The two models are then registered together to acquire an initial guess on the relative pose. The final solution is obtained iteratively by finding correspondences between the reconstructed objects from both measurements. Bai *et al.* [4] introduces an approach for calibration that relies on parallel lines commonly visible in urban environments (i.e., edges of buildings). The relative pose estimate is obtained by aligning the directions of the lines observed by the two sensors to find the relative orientation and then by solving a set of linear point-on-line constraints to find the relative translation. In recent years, the development of learning-based methods has also spanned in this field: Lv *et al.* [75] proposes a real-time self-calibration network that predicts the extrinsic parameters by constructing a cost volume between RGB and LiDAR features, while Sun *et al.* [121] first estimates an initial guess by solving a hand-eye calibration method. Moreover, the

guess is fine-tuned by segmenting the image-cloud pair and aligning the distances between centroids. The advantage of the target-less method is that it can be used without preparing the environment. This comes at the cost of lower accuracy and robustness when compared to their target-based counterpart.

Target-based methods: These methods estimate the relative pose using an observed known structure. Given the difference in resolution for the two sensors, it is highly unlikely that correspondences within the measurements can be established directly. For this reason, *point-to-point* methods tend to process LiDAR measurements to implicitly obtain virtual points[2] easily detectable from a RGB sensor. For instance, Park *et al.* [95] utilizes a specially designed polygonal planar calibration board with known lengths of adjacent sides. By estimating the 3D corresponding points from the LiDAR, vertices of the board can be determined as the meeting points of two projected sides. The vertices, along with the corresponding points detected from the color image, are used for calibration. Pusztai *et al.* [97, 96] introduces a methodology that utilizes cubic objects with predetermined side lengths. The corners of the cubes are estimated by initially detecting each side of the box and subsequently determining their intersection points. Furthermore, the corners and their corresponding RGB image are employed to calibrate the system by solving Iterative Closest Point (ICP). Zhou *et al.* [161] propose a single-shot calibration method requiring a checkerboard. The target is detected both in the RGB image and LiDAR measurement, using RANSAC[35] for the latter. Furthermore, the four edges of the checkerboard are estimated and aligned to compute the relative offset between the two sensors. Grammatikopoulos *et al.* uses a custom-made retro-reflective target paired with an AprilTag [91] fiducial marker to establish correspondences in the center of the target [44]. The relative pose is optimized by solving a Perspective-n-Point (PnP) problem. Tóth *et al.* introduces a fully automatic calibration technique that leverages the utilization of spheres, enabling accurate detection in both point clouds and camera images [126]. Upon successful detection, the algorithm aligns the set of sphere centers using SVD. Beltrán *et al.* [7] presents a methodology that utilizes a custom calibration target with four holes and AruCO markers specifically designed for monocular detection. The methodology employs a set of techniques for each sensor to estimate the center points of the holes. Subsequently, the relative offset between sensors is determined by aligning the set of centers obtained from each sensor, Li *et al.* [69] adopt a similar approach while using a checkerboard with four holes. Fan *et al.* [32] propose a two-stage calibration method using an auxiliary device with distinctive geometric features. The method extracts lines from images and LiDAR point clouds, providing an initial estimation of the external parameters. Nonlinear optimization is then applied to refine these parameters. In the work of Singandhupe *et al.* [114], the authors first extract planar information from RGB and LiDAR measurements, then, two grids of points are extracted from the computed planar patches and aligned using a customized ICP algorithm. Albeit these approaches provide accurate results with few measurements, care should be taken during the estimation of virtual correspondences, as they can cause significant errors in the estimation step. Moreover, these custom targets often require precise construction or expensive manufacturing.

Another group of approaches does not directly solve the calibration problem using point-to-point correspondences but rather exploits the flatness of the target to reduce the

[2]Points that are not explicitly detected, but estimated from the measurement.



(a)

(b)

Figure 3.2. Comparison between spherical projection and projection by ID. (a) shows a comparison between the spherical projection (top) described in Sec. 2.3.2 and the projection by ID (bottom) used for this work. (b) shows the ring information before (left) and after (right) the projection.

feasible set of solutions using *plane-to-plane* constraints. Mirzaei *et al.* addresses the challenge of accurate initial estimates by dividing the problem into two sub-problems and analytically solving each to obtain precise initial estimates [82]. The authors then refine these estimates through iterative minimization. They also discuss the identifiability conditions for accurate parameter estimation. Finally, in a method similar to our proposal, Kim *et al.* combine observed normals first to estimate the relative orientation with SVD and then iteratively estimate an initial guess of the relative translation by minimizing the pairwise planar distances between measurements [63]. Finally, the translation is refined using a non-linear optimization problem using Levenberg-Marquardt (LM). Despite its simplicity, this method decouples the estimation of orientation and translation, thus leading to potential losses in accuracy while also increasing the minimum number of measurements for an acceptable solution. In conclusion, compared to the State Of The Art (SOTA), we propose:

- a formulation for joint nonlinear optimization that couples relative rotation and translation using a plane-to-plane metric;
- an extensible framework that decouples the optimization from target detection. Currently, we supports Checkerboard and *ChARuCO* patterns of typical A3-A4 sizes, easily obtainable from commercial printers;
- the possibility to handle different camera models and distortion;
- a open-source implementation.

3.3 Method

This section first introduces several mathematical tools required to define our approach. Moreover, a detailed description of each component of the calibration pipeline is provided.

Plane Representation: Let $\boldsymbol{\pi} = (\mathbf{n}, d)$ be a 3D plane, where $\mathbf{n} \in \mathbb{S}^2$ represents its normal and $d \in \mathbb{R}$ is its orthogonal distance from the origin. Applying a transform $\mathbf{X} = (\mathbf{R}, \mathbf{t}) \in \mathbb{SE}(3)$ to a plane $\boldsymbol{\pi}$ yields new coefficients $\boldsymbol{\pi}'$ defined as

$$\mathbf{X}\boldsymbol{\pi} = \begin{cases} \mathbf{R}\mathbf{n} \\ d + (\mathbf{R}\mathbf{n})^T \mathbf{t}. \end{cases} \tag{3.1}$$

Since our goal is to iteratively refine $\mathbf{X}$ to match the relative pose between the sensors, we are interested at analyzing how Eq. (3.1) changes under small manifold updates, as described in Sec. 2.4.3. Let $\boldsymbol{\Delta}\mathbf{x}$ be the Lie algebra $\mathfrak{se}(3)$ associated with the manifold group $\mathbb{SE}(3)$, parametrized as $\boldsymbol{\Delta}\mathbf{x} = [\boldsymbol{\Delta}\mathbf{t}, \boldsymbol{\Delta}\mathbf{r}]^T \in \mathbb{R}^6$, where $\boldsymbol{\Delta}\mathbf{t} \in \mathbb{R}^3$ describes the translation and $\boldsymbol{\Delta}\mathbf{r} \in \mathbb{R}^3$ describes the rotation expressed in angle-axis representation. Following Sec. 2.4.3, we use the left-$\boxplus$ operator for applying the manifold update. We can thus rewrite the plane $\boldsymbol{\pi}$ subject to the transform $\mathbf{X} \boxplus \boldsymbol{\Delta}\mathbf{x}$ as

$$(\mathbf{X} \boxplus \boldsymbol{\Delta}\mathbf{x})\, \boldsymbol{\pi} = \begin{cases} \exp(\boldsymbol{\Delta}\mathbf{r}^\wedge)\mathbf{R}\mathbf{n} \\ d + (\mathbf{R}\mathbf{n})^T \mathbf{t} + \mathbf{n}^T \mathbf{R}^T \exp(\boldsymbol{\Delta}\mathbf{r}^\wedge)^T \boldsymbol{\Delta}\mathbf{t} \end{cases}. \tag{3.2}$$

Deriving the result for $\boldsymbol{\Delta}\mathbf{x}$ leads to the following Jacobian

$$\left.\frac{\partial(\mathbf{X}\ \exp(\boldsymbol{\Delta}\mathbf{x}))\boldsymbol{\pi}}{\partial \boldsymbol{\Delta}\mathbf{x}}\right|_{\boldsymbol{\Delta}\mathbf{x}=0} = \begin{bmatrix} \mathbf{0}_{3\times3} & -\lfloor \mathbf{R}\mathbf{n} \rfloor_\times \\ \mathbf{n}^T\mathbf{R}^T & \mathbf{0}_{1\times3} \end{bmatrix}_{4\times6}, \tag{3.3}$$

where $\lfloor \mathbf{v} \rfloor_\times$ maps the vector $\mathbf{v}$ into a skew-symmetric matrix defined as follows:

$$\lfloor \mathbf{v} \rfloor_\times = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & v_x \\ -v_y & v_x & 0 \end{bmatrix}. \tag{3.4}$$

The distance between two planes depends on the difference between their normals and the signed distance of the planes from the origin. These quantities can be captured by a 4D error vector $\mathbf{e}_p$ expressing the *plane-to-plane* error metric:

$$\mathbf{p}(\boldsymbol{\pi}_k) = -\mathbf{n}_k d_k \tag{3.5}$$

$$\mathbf{e}_p(\boldsymbol{\pi}_i, \boldsymbol{\pi}_j) = \begin{bmatrix} \mathbf{n}_i - \mathbf{n}_j \\ \mathbf{n}_j^T(\mathbf{p}(\boldsymbol{\pi}_j) - \mathbf{p}(\boldsymbol{\pi}_i)) \end{bmatrix}. \tag{3.6}$$

Here $\mathbf{p}(\boldsymbol{\pi}_k)$ is the point on the plane closest to the origin of the reference system, and it is obtained by taking a point along the normal direction $\mathbf{n}$ at a distance d.

Projection by ID (LiDAR): Differently from the spherical projection described in Sec. 2.3.2, this chapter leverages a different, non-differentiable projection function. Let $\mathbf{p}$ be a point expressed in the LiDAR frame. Its projection is computed as:

$$\boldsymbol{\pi}_\mathrm{l}(\mathbf{p}) = \mathbf{A}\psi(\mathbf{p}) \tag{3.7}$$

$$\mathbf{A} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & 1 & 0 \end{bmatrix} \tag{3.8}$$

$$\psi(\mathbf{v}) = \begin{bmatrix} \mathrm{atan2}(v_y, v_x) \\ \mathit{ring}(\mathbf{v}) \\ 1 \end{bmatrix} \tag{3.9}$$

where f_x represent the azimuth resolution of the LiDAR, while c_x denotes the offset in pixels. The $ring(\mathbf{v})$ function is typically provided by the LiDAR sensors and represents, for each point, the index of the vertical receiver that performed the measurement. In cases in which this information is unavailable but the cloud is ordered[3], the $ring(\mathbf{v})$ it is obtainable by dividing the point index by the number of horizontal measurements N_h of the sensor as

$$ring(\mathbf{v}_i) = \left\lfloor \frac{i}{N_h} \right\rfloor . \tag{3.10}$$

Fig. 3.2a shows the comparison with the classical spherical projection. We remark that, compared to the spherical projection described in Sec. 2.3.2, the projection by ID presents non differentiable properties, and doesn't allow back-projection, however, it is preferred for this applications due to the absence of empty pixels.

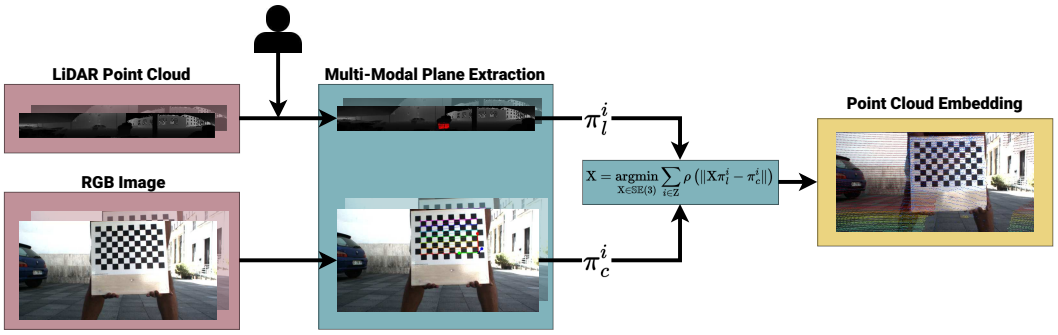


Figure 3.3. Diagram of our calibration pipeline. Measurements are acquired, and calibration target detection is performed (LiDAR planar detection is performed via human intervention). The set of planes is used to solve the non-linear optimization problem, leading to the optimal relative pose between the sensors.

Data preprocessing: First, we process the incoming raw LiDAR and RGB measurements for target detection and plane extraction. Assuming the scene remains static throughout the acquisition of a single joint measurement, the LiDAR cloud is projected via Eq. (3.7). Moreover, the system awaits the user interaction to guess the position of the calibration target on the LiDAR image. A parametric circular patch around the user's selection is used to estimate a plane using RANSAC, and concurrently, the calibration target detection is performed on the RGB image using a target-dependent method (i.e., OpenCV[11]). Once the target is detected, the RGB plane is computed by solving a PnP problem with OpenCV. If the user is satisfied with both LiDAR and RGB planes, they are stored for processing.

Optimization: Whereas a straightforward rank analysis of the Jacobians reveals that just three measurements are sufficient to constrain a solution, it is well known from the estimation theory that the accuracy grows with the number of measurements[4]. Once the set of measurements is acquired, we jointly estimate the relative orientation and translation of the

[3]The measurements are provided as an array of points, ordered by vertical receiver.
[4]From now on, a measurement represents a synchronized pair of LiDAR RGB measurements

Table 3.1. Average translation error. The error is expressed in millimeters with different noise levels and number of measurements *N*. Best results are highligthed as **first**, **second**, and **third**.

	$\sigma_l = 0$ $\sigma_c = 0$		$\sigma_l = 8e^{-3}$ $\sigma_c = 7e^{-3}$		$\sigma_l = 16e^{-3}$ $\sigma_c = 14e^{-3}$	
N	mean	stdev	mean	stdev	mean	stdev
3	41.761	104.362	20.790	25.124	57.849	112.365
4	10.872	17.941	12.206	12.363	14.940	11.681
5	6.492	7.997	8.350	9.076	9.115	5.675
10	4.591	3.458	5.759	4.974	5.849	1.989
20	2.575	1.981	3.646	2.564	4.123	1.139
30	2.673	1.263	2.867	1.659	3.735	0.878
39	**2.091**	**0.883**	**2.666**	**1.206**	**3.261**	**0.413**

LiDAR to the RGB sensor $\mathbf{X} \in \mathbb{SE}(3)$ by solving

$$\mathbf{X} = \underset{\mathbf{X} \in \mathbb{SE}(3)}{\operatorname{argmin}} \sum_{i \in \mathbf{Z}} \| \underbrace{\mathbf{X}\boldsymbol{\pi}_l^i - \boldsymbol{\pi}_c^i}_{\mathbf{e}_p} \|^2, \tag{3.11}$$

where $\mathbf{e}_p$ represent the plane-to-plane error.

During acquisition, the user may accept one or more wrongly estimated measurements. As described in Sec. 2.4.2, the quadratic nature of the error terms may over-account the outlier's contribution, resulting in wrong estimations. We employ the Huber *M-estimator* $\rho(\cdot)$ that treats different measurements based on their error to account for this factor. We rewrite Eq. (3.11) as

$$\mathbf{X} = \underset{\mathbf{X} \in \mathbb{SE}(3)}{\operatorname{argmin}} \sum_{i \in \mathbf{Z}} \rho \left(\| \mathbf{X}\pi_l^i - \pi_c^i \|^2 \right). \tag{3.12}$$

To resolve Eq. (3.12), we employ the Gauss-Newton (GN) algorithm.

3.4 Experiments

In this section, we describe the experiments we conducted to establish the quality of our calibration toolbox. We perform quantitative experiments in the simulated environment provided by [7] to compare our estimates with ground truth while we also conduct qualitative and quantitive experiments on real scenarios using our acquisition system. We directly compare our results with [63], as it is the work closest to ours. In addition, we compare to [7] that produce accurate results relying on a very complex target (CNC printed).

3.4.1 Synthetic Case

We conducted experiments on *Gazebo* simulator [66] to evaluate the accuracy and robustness of our approach, injecting different noise figures into the sensor measurements. We also experiment with how the number of observations affects the final results. The scene setup includes a Velodyne HDL-64 LiDAR, a BlackFly-S RGB sensor, and a 6×8 checkerboard

Method	e_t (cm)	e_r (10^{-2} rad)
Beltrán et al. [7]	0.82	**0.24**
Kim et al. [63]	10.2	129.56
Ours	**0.11**	0.25

Table 3.2. Quantitative results on synthetic data. The experiment is achieved through calibration using $N = 3$ measurements. We choose this measurement count for parity with the methodology proposed in [7]. In [7], a single measurement is deemed sufficient for calibration determination, with 3 measurements considered the optimal scenario. Beyond 3 measurements, accuracy does not improve significantly. Both for our study and in alignment with the findings in [63], 3 measurements represent the minimum requirement for solution determination, and an increase in this count is expected to result in more precise outcomes. Our results show that with our minimum number of measurements, we perform on par with [7] in rotation while outperforming all methods on translation, using small commercial tags.

target with a corner size of 0.2 meters. We randomly generate and acquire 53 valid[5] measurements.

To quantify the impact of the number of measurements on the accuracy of our approach, we run the calibration procedure with an increasing number of measurements $w_s = [3 \dots 39]$ and at three different LiDAR noise levels σ_l (0 mm, 7 mm, and 14 mm). For every w_s, we sample 40 sets of measurements.

From Tab. 3.1, we observe a steady decrease of error for every noise level, reaching an average of 2.6 mm translation error in the intermediate noise case. In the case of three measurements, the high uncertainty is due to the potentially poorly conditioned system when using planes with similar normals. Nonetheless, we compare our best result with 3 measurements against the best results of the methods presented in [7] and [63]. Tab. 3.2 shows the results.

3.4.2 Real Case

In this section, we describe the experiments conducted on real measurements. We perform a quantitative test on our acquisition system shown in Fig. 3.4 that is equipped with an Ouster OS0-128 LiDAR with a resolution of 128×1024, a RealSense T-265 stereo camera and two Manta-G145 RGB cameras arranged in a wide horizontal stereo configuration.

Since no ground truth information is available, we use the stereo extrinsic to estimate the calibration error. The offset between multiple cameras is measured using optical calibration procedures, which typically reach subpixel precision.

In the first experiment, we consider the LiDAR and the Realsense T-265 sensor, which provides two wide field-of-view fish-eye cameras with factory-calibrated intrinsic/extrinsic parameters. The task of the experiment is to demonstrate the accuracy of the calibrator in real-world scenarios and to verify how the number of measurements considered affects the quality of the solution. As for the synthetic case, we first acquire a set of 17 cloud-image LiDAR-RGB measurements for both cameras. Moreover we perform 40 calibrations with w_s randomly selected measurements with $w_s \in \{3, 15\}$. Finally, for every w_s, we combine the computed extrinsic for both cameras to obtain an estimated stereo transform. Assuming

[5] A valid measurement for which both LiDAR and RGB sensor can detect the target

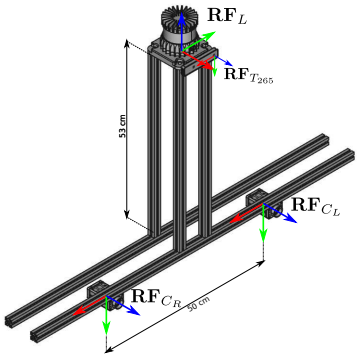


Figure 3.4. Acquisition system used for the Real Case experiments. We report nominal measurements between the sensors. The Realsense T265 ($180°$ horizontal field of view) is installed closer to the LiDAR, while the two Manta cameras ($90°$ horizontal field of view) are mounted in a further wide horizontal stereo baseline.

approximately symmetrical errors in the two cameras, Fig. 3.5a shows the results of this experiment. We obtained, at best, an average error of 7.1 mm in translation and 0.01 rads in orientation.

The second experiment is conducted using the wide stereo setup for which we also calibrate the intrinsics and extrinsics of the cameras, providing expected results in a typical scenario. The large parallax between the LiDAR and each camera and a smaller field of view allows us to evaluate our approach in a stressful scenario. The acquisition procedure is the same as in the first experiment, and Fig. 3.5b shows the experimental result, where we obtain the best solution with 4.6 mm and 0.2×10^{-2} rads in orientation.

Moreover, Fig. 3.1 and Fig. 3.6 show the reprojection onto the right camera, respectively of the fisheye and wide baseline RGB sensor. In the latter, the large parallax between the sensors leads to strong occlusion effects that have been mitigated with a hidden point removal algorithm [58].

Our evaluation indicates that our method can generate extrinsic estimates comparable or superior to those obtained using other SOTA approaches. It is important to note that careful consideration is required when selecting a minimal number of measurements. However, our experiments demonstrate that the accuracy of these estimates improves as the number of measurements increases.

3.5 Conclusions

The experiments show that planar features are a valid alternative to existing solutions for LiDAR-RGB calibrations due to resiliency to LiDAR inherent noise. In particular, Tab. 3.1 shows that similar translation error occurs across different noise levels of the sensors. Moreover, real-case experiments support our claim concerning the dimensions of the calibration target, brought down to A3/A4 dimensions, along with the seamless integration of different camera models (Kannala-Brandt [57] for T-265 and Radial-Tangential for Manta G-145). We suggest using our methodology in situations where calibration should be performed on-site, where the ad-hoc environment for calibration is not guaranteed, or where bringing more specialized calibration targets is not feasible. An important note regards the

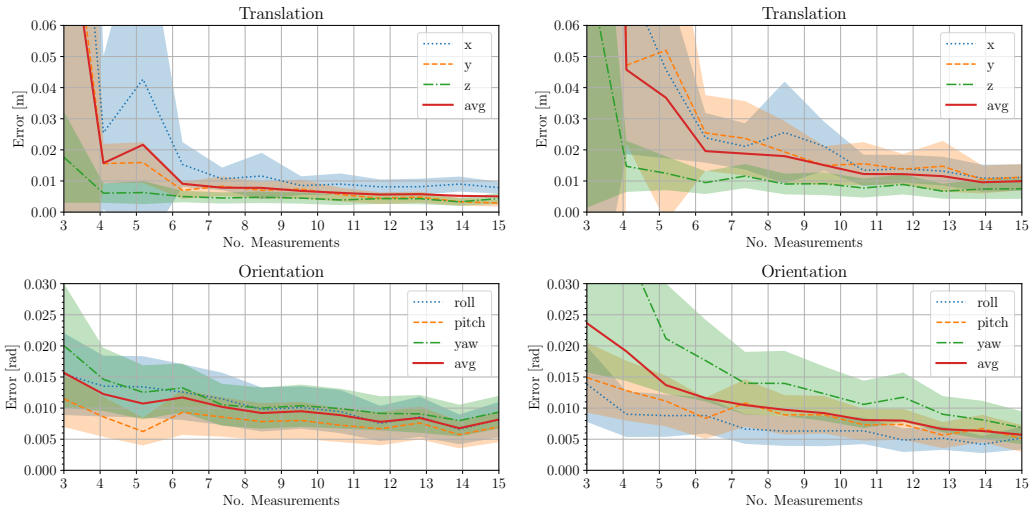


Figure 3.5. Quantitative pose error for real scenarios.(a) Average camera-wise calibration error in the LiDAR-T265, wide fov case. (b) shows the average camera-wise calibration error in the LiDAR-Manta case while

sensors' configuration and shared field of view. Ensuring a correct result requires multiple views of the calibration target from both perspectives. In those cases where the shared field of view is small, a single-shot calibration approach might produce better results in terms of accuracy. Finally, concerning situations where the calibration target is not static during the acquisition, caution should be taken on temporal synchronization of the measurements. Our system assumes input measurements to be synchronized; moreover, even small temporal offsets may worsen the calibration accuracy. We remark on the difficulty of synchronizing these two sensors due to their different nature. Specifically, the acquisition time for rotating LiDARs is typically much higher compared to the exposure time of a commerical RGB sensor. To lessen the synchronization issues, we suggest to trigger the capture of RGB images when the LiDAR is scanning the overlapping Field Of View (FOV).

In conclusion, in this chapter, we introduced a simple and effective method for accurately estimating extrinsic parameters between LiDARs and RGB sensors. By leveraging the inherent planar shape of standard calibration patterns, we establish common observations between these sensors, greatly simplifying the calibration procedure. Our experiments show that planar features mitigate the LiDAR noise, leading to accurate results even with common A3/A4 calibration patterns.

Figure 3.6. Qualitative examples of LiDAR cloud projection on RGB images. The recording setting is the one sketched in Fig. 3.4, where the parallax between the sensor is approximately 66 cm.

Chapter 4

Photometric LiDAR and RGB-D Bundle Adjustment

4.1 Introduction

Simultaneous Localization and Mapping (SLAM) has gained significant popularity in the field of robotics, and thanks to approximately three decades of research, there are now effective solutions available. As SLAM is widely used in various sectors, such as autonomous driving, augmented reality, and space exploration, it continues to attract significant attention from academia and industry. Modern SLAM systems typically comprise two key components: a front-end that estimates sensor odometry in real-time and a back-end that optimizes previous sensor poses and, eventually, the 3D map. The gold standard for the back-end optimization is Bundle Adjustment (BA) [127]. Photometric (or direct) approaches have been used by the computer vision community to tackle the SLAM or Structure from Motion (SfM) problems. The direct techniques address registration by minimizing the pixel-wise error between image pairs. By not relying on specific features and having the potential of operating at subpixel resolution on the entire image, direct approaches do not require explicit data association and boost the registration accuracy [62, 109]. Although these methods have been successfully employed on monocular, stereo, or RGB-D sensors, their use on 3D LiDAR data has not been widely exploited. Della Corte *et al.* [19] presented a general photometric registration methodology that extends direct approaches to different projective models and enhances the optimization robustness by considering additional channels such as normals. Nowadays, 3D LiDARs offer up to 128 channels of resolution, measuring more than 5M points per second. The enhancement of density, the increase in vertical resolution, and the capacity to measure the reflectivity of a surface make it logical to render a scan onto a panoramic image. This consideration makes photometric approaches attractive for 3D LiDARs.

The main contribution described in this chapter is a unified photometric BA strategy that works for both RGB-D and LiDAR. Our method aims to refine the trajectory coming from a SLAM/GNSS system to maximize its photometric consistency. Our approach implicitly addresses the data association and straightforwardly supports multiple heterogeneous sensors. We performed a comparative evaluation of benchmark data concerning State Of The Art (SOTA) sensor-specific refinement strategies and SLAM algorithms. Results show that our simple optimization schema is very effective, performing on par or better than methods

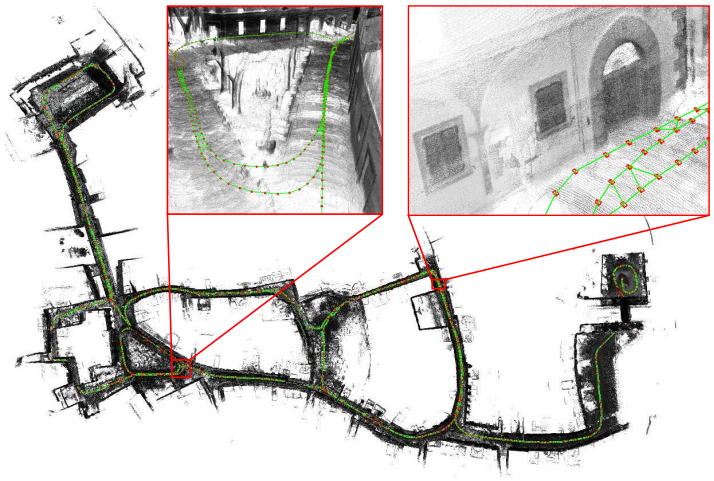


Figure 4.1. Reconstruction of Viterbo city-center (Italy) using our data recorded with an OS0-128. The trajectory, which is about 2 km long, has been estimated first with MD-SLAM [28] and then refined with our photometric BA strategy. This image highlights both the global and local consistencies. We show reconstruction details with multiple scans of the same places acquired over time.

specialized for RGB-D and LiDAR data. We also demonstrate how our photometric BA strategy can be improved by fusing 3D LiDAR and RGB-D. We release an open-source CUDA/C++ implementation of this work. Fig. 4.1 shows a reconstruction of the Viterbo city-center (Italy) using our self-recorded data. From the detailed views, it is possible to appreciate the fine map resolution after performing our BA strategy.

4.2 Related Works

Approaches for global refinement such as BA are widely used in Visual SLAM and SfM systems but are less common for LiDAR. In this work, we provide a unified photometric global registration method for both RGB-D and LiDAR that can improve the accuracy of the trajectory and hence, the map obtained by standard SLAM systems that rely on Pose-Graph Optimization (PGO). PGO formulation reduces the optimization problem's size but approximates the original problem by marginalizing the projective observations.

In the RGB-D SLAM field, BA can be classified into two categories: *direct* and *indirect*. In the following, we will explain the difference between these two paradigms and review the SOTA of both approaches. We finally discuss BA applications in LiDAR SLAM. Indirect SLAM methods estimate camera poses and 3D structure by minimizing reprojection error of feature points, making them faster and less sensitive to calibration and synchronization issues [109]. SOTA implementations perform windowed BA to refine the map in a neighborhood of the sensor location, and global BA is invoked upon loop closures on the entire trajectory. To keep the problem tractable, the trajectory is subsampled in a set of keyframes, and only some salient feature points are considered in the optimization [84, 65].

Direct methods directly minimize the photometric error between overlapping images

without relying on feature detection and matching. The photometric error is calculated as the difference between the measured and predicted pixel intensities based on an estimate of the 3D structure and camera parameters. Delaunoy and Pollefeys propose an offline dense 3D reconstruction methodology that refines the scene and camera parameters to minimize photometric error. The scene is represented by triangular meshes, which require frequent remeshing, making the approach computationally demanding [24]. Goldlücke *et al.* [43] present a variational framework that estimates a super-resolution curved surface to precisely represent the 3D scene, leading to improved estimates and high-quality texture output. Slavcheva *et al.* [115] perform pairwise alignment by registering two consecutive signed distance functions (SDFs) generated from the depth images, which is used as global refinement. The direct methods presented are computationally heavy, being mainly used for offline 3D reconstructions.

To enhance the computation, some researchers exploited the decomposability of the problem and investigated data reduction strategies to reduce the computation of BA in SfM applications. Alismail *et al.* [2] consider only pixels with reasonable gradient in the error function to reduce the problem's size. Eriksson *et al.* [31] propose a consensus-based optimization to parallelize BA for large-scale problems. Demmel *et al.* [26] propose a novel solution in Distributed BA by breaking down the problem into smaller subparts using the k-means clustering method, and structuring the resulting subgraphs into well-constrained connected segments for more efficient processing.

In between the class of direct and indirect methods, we find the work of Forster *et al.* [37], which proposes a hybrid method to estimate the camera's motion. First, an initial guess of the sensor location is computed by minimizing the reprojection error of the world points. Then the estimate is refined by minimizing the photometric error of the patches around the feature points. The final map refinement step is done by performing direct BA.

Recently, the community has focused on embedding BA refinement in SLAM applications. Hybrid approaches have gained traction to combine the accuracy of direct methods with the robustness of feature-based ones by mixing direct and indirect error terms in optimization strategies. One example is Bundle Fusion [22], which interleaves feature-based and photometric BA to refine the global estimate. The photometric refinement considers only camera poses and not the structure. Similarly, BAD-SLAM [109] minimizes a cost function that accounts for geometric and photometric errors in motion estimation and global refinement processes, with three main steps: refining the 3D scene modeled by surfel, optimizing camera poses with a fixed model, and refining the camera's intrinsics.

In parallel, the community approached LiDAR-based SLAM by seeking alternative representations for the dense 3D point clouds. Given the accuracy of these measurements, the robotics community addressed the problem of building a map incrementally registering new scans.

Many registration techniques have been exploited using LiDAR data. These include 3D salient features [154], subsampled clouds [128] or Normal Distributed Transform (NDT) [118]. Nowadays, LiDAR Odometry and Mapping (LOAM) is perhaps one of the most popular methods for LiDAR odometry [154, 155]. It extracts distinct features corresponding to surfaces and corners, then used to determine point-to-plane and point-to-line distances to a voxel grid-based map representation. A revised approach (Lego-LOAM) has been suggested [111], which takes advantage of a ground surface in its segmentation and optimization steps. Odometry estimation techniques, or more generally 3D point clouds registration routines, coupled with place recognition and PGO show satisfying results within LiDAR SLAM

[111, 28]. This makes PGO the gold standard optimization method in LiDAR community. A pose-graph represents the trajectory, and observations between pairs of poses along the path are computed by registering the two overlapping clouds. Hence, an observation is a relative transform and potentially a covariance matrix. With this approximation, the constraints between the poses can be represented in a relatively compact manner. The optimum of a PGO is the configuration of poses that is maximally consistent with the transforms in the measurements. Albeit efficient, PGO approaches operate on approximating the original problem since pairwise measurements are computed once during the SLAM phase and never revised. Unavoidable drifts will accumulate, and wrong behavior of the place recognition might lead to inconsistent graphs as shown in [153].

In order to remove these inconsistencies, recently Liu and Zhang presented a global geometrical optimization methodology that considers cloud measurements [73]. In particular, it formulates a cost function based on LOAM features (i.e., edges and planes) and globally optimizes the trajectory to maximize the features' overlap. This approach performs a static data association based on the co-visibility of landmarks; thus, it requires a good initial guess to operate.

In recent years, the vertical resolution of modern 3D LiDARs increased, and these devices provide scans that resemble more and more dense panoramic images. This allows us to transfer results about direct global refinement from vision to LiDAR. In this chapter, we present a unified BA strategy that works independently for LiDAR and RGB-D in the same way. By operating directly on images, our method constantly refines the data association during the optimization; hence it is less sensitive to poor initial guesses. Our algorithm's capabilities are demonstrated through quantitative and qualitative analyses, consistently improving the initial estimates provided by any SLAM systems.

4.3 Method

The goal of our approach is to compute the set of sensor poses $\{\mathbf{X}_i\}_{i=1}^{N} \; : \; \mathbf{X}_i \in \mathbb{SE}(3)$ whose photometric error between overlapping frames is minimized. The input of our system is a set of triplets $\left\langle \hat{\mathbf{X}}_i, \mathcal{I}^{\mathrm{g}}, \mathcal{I}^{\mathrm{d}} \right\rangle$, containing the initial guess of the sensor pose, the grayscale[1]/intensity image $\mathcal{I}^{\mathrm{g}}$ and the depth image $\mathcal{I}^{\mathrm{d}}$. The workflow of our method is illustrated in Fig. 4.2. The first step is to generate an augmented image pyramid from each pair of images in a triplet (Sec. 4.3.1). The second step is determining which pairs of images observe a common structure, which is discussed in Sec. 4.3.2. Finally, these pairs will be used to instantiate a photometric optimization problem presented in Sec. 4.3.3. Without using geometrical error functions or structure optimization, our approach is entirely free from any feature association. This makes our method simple, compatible with multiple depth sensors, and competitive with ad-hoc SOTA approaches.

4.3.1 Input Preprocessing

The input of our photometric refinement method is a pair of intensity $\mathcal{I}^{\mathrm{g}}$ and depth $\mathcal{I}^{\mathrm{d}}$ images for each sensor pose. The output of the preprocessing step is a five-channel image pyramid for each input pair. The first two channels of an image in the pyramid are intensity and depth, while the other three channels encode the surface normals. To calculate the normal at

[1]We focus on grayscale images to maintain consistency between LiDAR and RGB-D measure dimensionality.

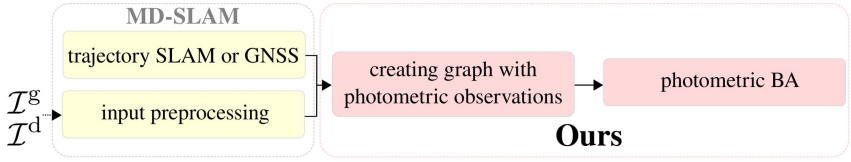


Figure 4.2. The flow of our approach. The input of our system contains the initial guess of the sensor pose, the intensity image $\mathcal{I}^{\mathrm{g}}$ and the depth image $\mathcal{I}^{\mathrm{d}}$. MD-SLAM already provides the input in the correct format since the preprocessing step is embedded in the SLAM system. The core of our refinement strategy, highlighted in red in this graph, consists in creating a graph with photometric observations (Sec. 4.3.2) and running a global optimization on the set of poses $\mathbf{X}_{1:N}$, as detailed in Sec. 4.3.3.

pixel $\mathbf{u}$ we back-project[2] the pixels in the neighborhood $\mathcal{U} = \{\mathbf{u}' : \|\mathbf{u} - \mathbf{u}'\| < \tau_{\mathbf{u}}\}$ of a radius $\tau_{\mathbf{u}}$ inversely proportional to the range at the pixel $\mathcal{I}^{\mathrm{d}}(\mathbf{u})$. The normal $\mathbf{n}_{\mathbf{u}}$ is the one of the plane that best fits the back-projected points from the set $\mathcal{U}$, and is oriented towards the observer. All valid normals are assembled in a normal image $\mathcal{I}^{\mathrm{n}}$, so that $\mathcal{I}^{\mathrm{n}}(\mathbf{u}) = \mathbf{n}_{\mathbf{u}}$. Hence, the final five-channel image is obtained by stacking together $\mathcal{I}^{\mathrm{g}}$, $\mathcal{I}^{\mathrm{d}}$, and $\mathcal{I}^{\mathrm{n}}$. In the remainder, we will refer to the generic channel as a *cue* $\mathcal{I}^{\mathrm{c}}$.

Photometric approaches perform an implicit data association at a pixel level. Whereas attractive for their accuracy, these methods suffer from relatively small convergence basins that decrease with the image's resolution: the higher the image resolution, the narrower the convergence basin will be. To lessen this effect, we generate multiple copies of the same image at decreasing resolution to form a pyramid. The optimization will proceed from the coarser to the finest level (Fig. 4.4).

Each level of a pyramid consists of a multi-cue image generated from $\mathcal{I}^{\mathrm{g}}$, $\mathcal{I}^{\mathrm{d}}$ and $\mathcal{I}^{\mathrm{n}}$, by downscaling at user-selected resolutions. In our experiments, we typically use three scaling levels, each half the resolution of the previous level.

4.3.2 Graph Construction

Our system takes as input a collection of triplets $\left\langle \mathbf{X}_i, \mathcal{I}^{\mathrm{g}}, \mathcal{I}^{\mathrm{d}} \right\rangle$, that consist of the initial estimate of the sensor's pose and pairs of images for both intensity and depth. To instantiate Eq. (4.3) we need to determine the matching pairs $\langle i, j \rangle$. The problem can be visualized as an undirected graph, where each node is a triplet, and an edge between two nodes encodes a potential match.

To compute the matches, we start from the initial assignment of poses $\{\mathbf{X}_n\}$ and add edges to the graph based on the input data. To this extent, we use a straightforward criterion that generates a matching pair if two poses $\mathbf{X}_i$, $\mathbf{X}_j$ are close in space, and their orientations are similar. More specifically, we create a pair between $\mathbf{X}_i$, $\mathbf{X}_j$ if all these conditions are satisfied:

1. the angle between the poses is below a threshold (typically $30\,\mathrm{deg}$);
2. the translation between the poses is below a threshold (typically below 1 meter);
3. the ratio of reprojected valid points from $\mathbf{X}_i$ onto $\mathbf{X}_j$ is sufficiently high (typically $1/3$).

[2] See Sec. 2.3.1

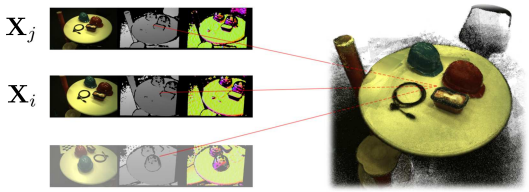


Figure 4.3. Example of associated pose-pairs. Our BA strategy relies on the association of $\mathbf{X}_i$ and $\mathbf{X}_j$ if they share observations. In the picture above, the input images with depth and normals are on the left, and on the right is the reconstructed model using our methodology. Note that in reality, inputs of our BA are pyramids (as discussed in Sec. 4.3.1, i.e., images of different resolutions). Here, we show just one level for simplicity. The data used is from ETH3D dataset [109].

In addition to these criteria, if the data come from a sequential acquisition, we add matches between subsequent triplets to model odometry-like constraints. An example of pose-pair associated is shown in Fig. 4.3.

4.3.3 Photometric Error Minimization

Our method seeks to find the set of transformations $\mathbf{X}^*_{1:N} \in \mathbb{SE}(3)^N$ that minimizes the photometric error between each candidate pair of sensor poses that observe a common portion of the environment. Let $\mathcal{I}_i$ and $\mathcal{I}_j$ be two images acquired from poses $\mathbf{X}_i$ and $\mathbf{X}_j$ that form a matching pair. Let $\mathcal{I}(\mathbf{u})$ be the value of the pixel $\mathbf{u}$ in the image $\mathcal{I}$.

The photometric error at image coordinates $\mathbf{u}$ in the matching pair is the difference between $\mathcal{I}^{\mathrm{g}}_i(\mathbf{u})$ and the pixel $\mathcal{I}^{\mathrm{g}}_j(\mathbf{u}')$ of the second image. The evaluation point $\mathbf{u}'$ is computed by back-projecting the pixel $\mathbf{u}$ from $\mathcal{I}^{\mathrm{g}}_i$ onto the image plane of $\mathcal{I}^{\mathrm{g}}_j$. This accounts for the relative transform $\mathbf{X}_{j,i} = \mathbf{X}_j^{-1}\mathbf{X}_i$ between the two frames, as follows:

$$\mathbf{u}' = \pi\left(\mathbf{R}_j^T\left(\mathbf{R}_i\pi^{-1}\left(\mathbf{u}, d\right) + \mathbf{t}_i - \mathbf{t}_j\right)\right), \tag{4.1}$$

where π is the pinhole projection $\boldsymbol{\pi}_c$ (Sec. 2.3.1) in case of RGB-D measurements and $\boldsymbol{\pi}_l$ (Sec. 2.3.2) for LiDAR ones. To carry out this operation, the depth at the pixel $d = \mathcal{I}^{\mathrm{d}}(\mathbf{u})$ needs to be known. Standard photometric optimization seeks to find the following minimum:

$$\mathbf{X}^*_{1:N} = \underset{\mathbf{X}_{1:N}}{\operatorname{argmin}} \sum_{\langle i,j\rangle}\sum_{\mathbf{u}} \|\mathcal{I}^{\mathrm{g}}_i(\mathbf{u}) - \mathcal{I}^{\mathrm{g}}_j(\mathbf{u}')\|^2. \tag{4.2}$$

Here the inner summation computes the photometric error of a matching pair $\langle i, j\rangle$ as the squared norm of the error of all pixels $\mathbf{u}$ while the outer summation spans over all matching image pairs.

Eq. (4.2) models classical photometric error minimization assuming that the cues are unaffected by $\mathbf{X}_{j,i}$. Whereas this is true when operating with pure intensity/grayscale values, normals, and depths change when mapped from the frame $\mathbf{X}_i$ to the frame $\mathbf{X}_j$. As in in [19] these mappings can be encapsulated by the function $\zeta^{\mathrm{c}}(\mathbf{X}_{j,i}, \mathcal{I}^{\mathrm{c}}_i(\mathbf{u}))$ that calculates the *pixel* value of the c^{th} cue after applying the transform $\mathbf{X}_{j,i}$ to the original channel value $\mathcal{I}^{\mathrm{c}}_i(\mathbf{u})$.

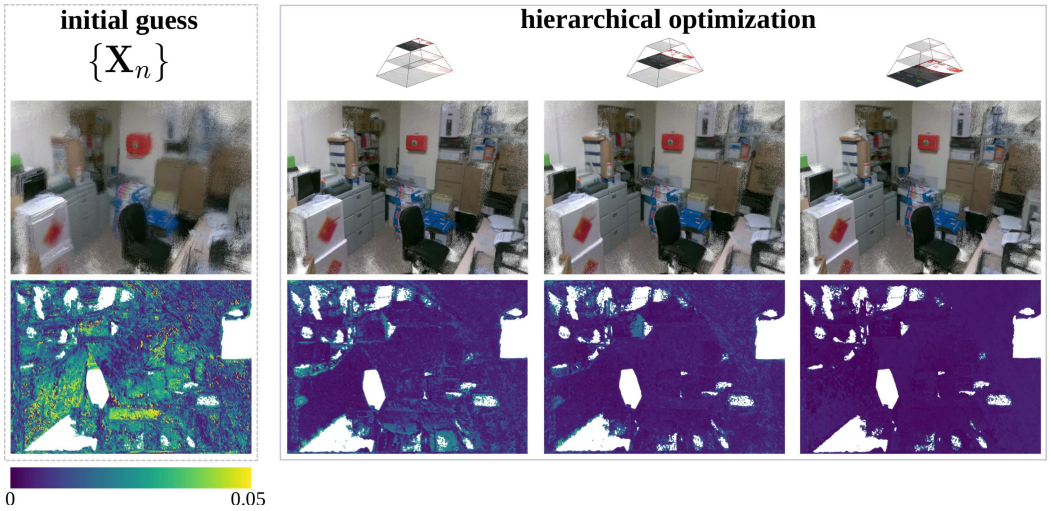


Figure 4.4. The effect of hierarchical optimization when performing BA. From left to right, we show how the quality of the estimate improves, starting from a bad initial guess. Starting from coarser, each level bootstraps the succesive one by estimating a better estimate, avoiding local minima solutions. The heatmap is normalized between 0 and 0.005 [m] for better visualization. The data is self-recorded using an Intel D455.

By extension, let $\hat{\mathcal{I}}_i^{\mathrm{c}} = \zeta^{\mathrm{c}}(\mathbf{X}_{j,i}, \mathcal{I}_i^{\mathrm{c}})$ be the image obtained by remapping $\mathcal{I}_i^{\mathrm{c}}$, according to $\mathbf{X}_{j,i}$. We can thus rewrite a more general form of Eq. (4.2) that accounts for all cues and captures this effect as follows:

$$F(\mathbf{X}_{1:N}) = \sum_{i,j} \sum_{\mathbf{u}} \rho \left(\left\| \sum_{\mathrm{c}} \hat{\mathcal{I}}_i^{\mathrm{c}}(\mathbf{u}) - \mathcal{I}_j^{\mathrm{c}}(\mathbf{u}') \right\|_{\mathbf{\Omega}^{\mathrm{c}}}^2 \right) \tag{4.3}$$

$$\mathbf{X}_{1:N}^* = \underset{\mathbf{X}_{1:N}}{\operatorname{argmin}} F(\mathbf{X}_{1:N}) \tag{4.4}$$

where ρ is a Huber robust M-estimator. The squared Mahalanobis distance $\| \cdot \|_{\mathbf{\Omega}^{\mathrm{c}}}^2$ is used to weight the different cues. In our experiment we typically set $\mathbf{\Omega}^{\mathrm{g}} = 0.6$, $\mathbf{\Omega}^{\mathrm{n}} = 0.8$ and $\mathbf{\Omega}^{\mathrm{d}} = 1$, giving more importance to depth and surface normals compared to grayscale channel. To carry out the minimization in Eq. (4.4) we employ the Levenberg-Marquardt (LM) algorithm implemented in the `srrg2_solver` [45].

At each iteration, we suppress the occluded portions of the images before evaluating Eq. (4.3). The optimization proceeds by seeking the optimum of all poses, starting from the coarser level. Once convergence is reached at one level, our system switches to the next finer one, and the optimization proceeds by choosing the solution computed so far as an initial guess. Fig. 4.4 shows the effect of this hierarchical approach applied to a BA problem. Generally, the worse the initial guess, the more the required levels.

4.4 Experiments

In this section, we report the results of our method on different public benchmark datasets. To the best of our knowledge, our approach is the only open-source photometric BA strategy that can deal with RGB-D and LiDAR in a unified manner. Therefore, to evaluate our

system, we compare it with SOTA SLAM and BA packages developed specifically for each of these sensor types.

To run the experiments, we used a PC with an Intel Core i7-7700K CPU @ 4.20GHz, 32GB of RAM and a Zotac Geforce GTX 1070 X 8G. Our BA schema is implemented on the GPU using CUDA 11. Since this work is focused on global consistency, we perform our quantitative evaluation using the RMSE on the absolute trajectory error (ATE) with $\mathbb{SE}(3)$ alignment. The metric's alignment is computed using the Horn method [50], and the timestamps are used to determine the associations. Then, we calculate the RMSE of the translational differences between all matched poses. In Sec. 4.4.1, we discuss the approaches and the datasets used for comparison with RGB-D sensors, while in Sec. 4.4.2 we present the results for LiDAR data. Since our implementation can run both on GPU and CPU, we report the runtimes of our algorithm for various pyramid resolutions using different commercial GPUs and our processor (Fig. 4.6).

In our experiments, to switch from one level to the other, we use a simple termination criterion involving the variation of the error in Eq. (4.3) over the iterations. The number of iterations for each level differs from the quality of the initial guess. In our experiments, we tend to achieve success after 10 iterations on coarser levels, 5 iterations on middle levels, and only a few iterations on the finest level.

In Sec. 4.4.3, we present a demonstration illustrating the impact of employing RGB-D and LiDAR simultaneously on the final estimate. Our findings clearly indicate that the use of these two sensors in conjunction, as part of our unified approach, consistently outperforms the single sensor modality.

In Fig. 4.5, we show the contribution of different cues (depth, grayscale, and surface normals) on the estimation results for both RGB-D and LiDAR. Our findings reveal that leveraging all available information enhances the performance of both sensors, with a more pronounced effect observed for LiDAR. The inclusion of surface normals plays a vital role in accurately determining the orientation of the estimate, thereby influencing the ATE. Additionally, we observe that the improvement in LiDAR performance diminishes when intensity values are combined with depth values. This can be attributed to the active nature of the LiDAR sensor, which yields contrasting results compared to RGB-D, where relying solely on noisy depth data is insufficient for achieving significant improvements.

4.4.1 RGB-D

As a public benchmark for RGB-D we used several sequences of the ETH3D dataset [109]. This dataset is acquired with global shutter cameras and accurate active stereo depth. Modeling rolling shutter effects and light changes cannot be encapsulated in the projection function $\pi(\cdot)$, our only pipeline component that differs between the RGB-D and LiDAR. For this reason, we restrict our comparison to the setting mentioned above.

The work proposed in this chapter refines the map from a reasonable initial guess. We compute such initial configurations employing an improved online CUDA version of MD-SLAM [28], our previous SLAM system that unifies depth sensors through hierarchical photometric odometry estimation and feature-based loop-closures.

We compare different approaches representative of different classes of SLAM and BA algorithms specific for RGB-D sensors: DVO-SLAM [62], ElasticFusion [140], BundleFusion [22] BAD-SLAM [109], ORB-SLAM2 [84]. DVO SLAM implements a mixed geometry-based and direct registration. Internally the alignment between pairs of keyframes

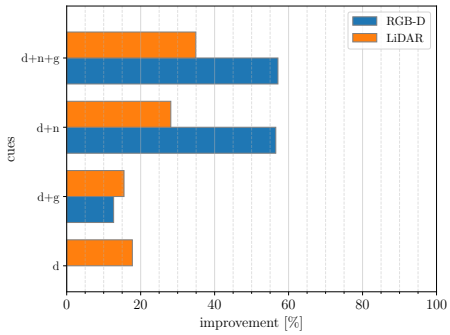


Figure 4.5. The contribution of each cue in our BA strategy. Our strategy uses three types of information (grayscale/intensity g, depth d, and surface normals n), the histogram shows the contribution of pairing each of them for RGB-D and LiDAR with respect to the initial guess in terms of ATE. Overall, using the three information together perform always better compared to coupling only few of them, for both the sensors.

is obtained by jointly minimizing point-to-plane and photometric residuals. This is similar to ElasticFusion, whose estimate consists of a mesh model of the environment and the current sensor location instead of the trajectory. BundleFusion refines the global estimate by interleaving feature-based and photometric BA. Similar to us, their photometric refinement does not consider the structure, but only the sensor poses. This method highly depends on data association, employing correspondences based on sparse features and dense geometric/photometric matching. BAD-SLAM is a surfel-based direct Bundle Adjusted SLAM system that combines photometric and geometric errors alternating optimization of motion and structure. In contrast to these approaches, ORB-SLAM2 implements a traditional visual SLAM pipeline, where a local map of landmarks around the RGB-D sensor is constructed from ORB features [101]. The map is constantly optimized as the camera moves by performing local and global BA.

Most of the compared approaches run global refinement on a separate thread in an anytime fashion. The work presented in this chapter addresses only this global aspect. Reporting the timings of this experiment would be unfair since our method addresses only a part of the problem.

ETH3D provides images of 740×460 pixels. We compute a 3-level pyramid from these images with scales 1/2, 1/4, and 1/8. In Tab. 4.1, we can see that our photometric refinement performs on par (second after BAD-SLAM) with other SOTA ad-hoc RGB-D SLAM and BA systems. Our method reduces the trajectory error to a few millimeters. More importantly, it improves by 60% the accuracy of the initial guess provided by MD-SLAM. Fig. 4.3 and Fig. 4.4 illustrate the effect of the hierarchical optimization on self-recorded data.

Summarizing, using our general pipeline of MD-SLAM and photometric BA described in this chapter provides results comparable with other RGB-D specific approaches, being second only to BAD-SLAM.

4.4.2 3D LiDAR

To validate our approach on LiDAR measurements we used both our data and public benchmarks. We used the Newer College Dataset [156] as a public benchmark. The dataset

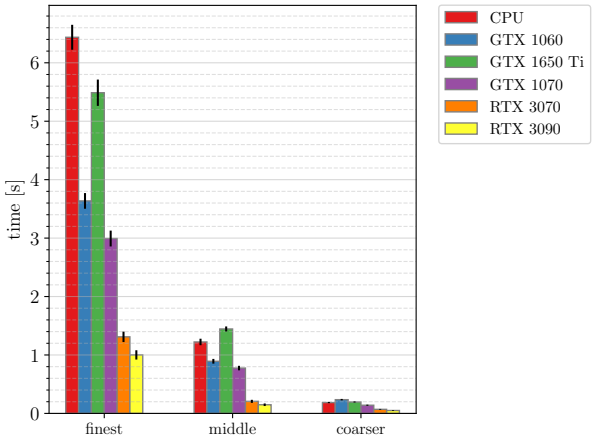


Figure 4.6. Runtimes of our BA strategy evaluated on multiple commercial GPUs and our Intel Core i7-7700K CPU. Cumulative time and standard deviation spent on an iteration of optimization. In this experiment, the finest level includes around 86M pixels, the middle level includes 22M pixels, and the coarser level has around 5M pixels. Time is expressed in seconds.

	ElasticFusion	ORB-SLAM2	DVO-SLAM	BundleFusion	BAD-SLAM	MD-SLAM	MD-SLAM + **Ours**
table3	–	0.007	0.008	0.017	**0.002**	0.016	0.009
table4	0.012	0.008	0.018	–	**0.002**	0.023	0.008
table7	–	0.010	0.007	0.010	**0.003**	0.018	0.009
cables1	0.018	0.007	**0.004**	0.022	0.007	0.021	0.006
plant2	0.017	0.003	0.003	0.004	**0.001**	0.005	**0.001**
planar2	0.011	0.005	**0.002**	0.003	0.003	0.009	0.004
mean	0.014	0.007	0.007	0.011	**0.003**	0.015	0.006
std	0.003	0.002	0.005	0.008	0.002	0.007	0.003

Table 4.1. ATE RMSE [m] on ETH3D benchmark, recorded with global shutter camera and synchronous streams. ElasticFusion fails in *table3* and *table7*, BundleFusion fails in *table4*. Best results are highligthed as **first**, **second**, and **third**.

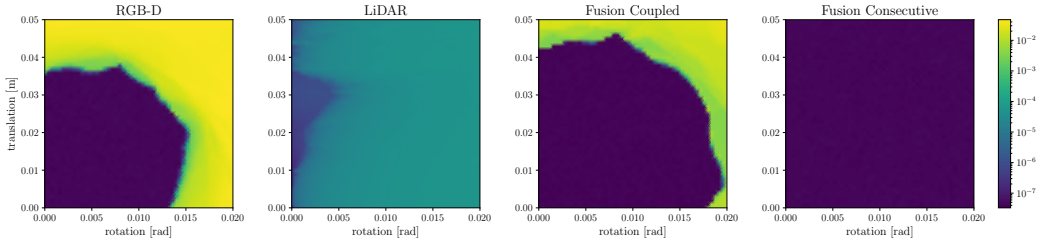


Figure 4.7. Results of fusion experiments. Plots show how the initial perturbation affects the convergence basin and the algorithm's accuracy. On the x-y axis, respectively, translation [m] and rotation [rad] perturbations, the value is denoted by the mean between rotation and translation error in log scale. Fusing the two sensors with our straightforward approach always shows better results compared to single sensors.

is recorded at 10 Hz with Ouster OS0-128. More specifically, we used the *cloister*, *quad* (easy), and *stairs* sequences. The *quad* sequence contains two loops that explore the Oxford campus courtyard, *cloister* mixes outdoor and indoor scenes while *stairs* captures an indoor scenario with multiple floors. Being based on image comparison, our approach operates well on LiDAR data having a good vertical resolution. LiDARs with fewer beams (i.e., 64, 32, 16) would produce an unbalanced horizontal image, reducing the converge basin of the algorithm.

We compare our method with BALM2 [73], which, to the best of our knowledge, is the only publicly available LiDAR global refinement approach. BALM2 is based on the overall consistency of points, lines, and planes. This system requires the same input as our method, namely an initial guess trajectory and the point clouds.

To compute the initial guess, we used several SLAM algorithms specific for LiDAR: LeGO-LOAM [111], SuMA [6] and our unified MD-SLAM. LeGO-LOAM is a pure geometric feature-based frame-to-model LiDAR SLAM system, where the optimization on roll, yaw, and z-axis (pointing up) is decoupled from the planar parameters. SuMa constructs a surfel-based map and estimates the changes in the sensor's pose by exploiting the projective data association in a frame-to-model or frame-to-frame fashion.

Tab. 4.2 reports the accuracy of our method and BALM2, for each dataset, and each initial guess. The ATE of the SLAM solution measures the quality of an initial guess. We observe that LeGO-LOAM provides a good guess on all planar data but fails on the *stairs* dataset resulting in an ATE of more than 3 meters. MD-SLAM performs reasonably well with a maximum ATE of 0.36 meters, while SuMA yields an acceptable initial guess only in the stairs dataset. If the initial guess is good, both BALM2 and our method perform well, improving the initial estimate. However, as the initial guess degrades, our unified global refinement's accuracy remains stable by systematically improving the trajectory estimate. On these data, we observed BALM2 to be particularly sensible to rotations and less dense trajectories (i.e., trajectories sampled with fewer keyframe poses). Quantitative results show that our strategy is successful within LiDAR data, raising the initial estimate close to 60% improvement when a good initial guess is provided (i.e., *stairs* with MD-SLAM). The convergence basin of these global refinement strategies are bounded by inconsistency in the laser measurements (i.e., skewed point clouds); this is why none of the systems can further enhance the trajectory in the LeGO-LOAM *quad* experiment.

Fig. 4.1 shows our large-scale reconstruction of the historical part of Viterbo (Italy) from

	LeGO-LOAM			MD-SLAM			SuMA		
	SLAM	BALM2	**Ours**	SLAM	BALM2	**Ours**	SLAM	BALM2	**Ours**
cloister	0.20	0.25	**0.16**	0.36	0.37	**0.32**	3.34	2.67	**2.53**
quad	**0.09**	**0.09**	**0.09**	0.25	1.98	**0.17**	1.74	1.72	**1.68**
stairs	**3.20**	5.13	3.48	0.23	0.38	**0.10**	0.67	0.86	**0.60**

Table 4.2. ATE RMSE [m] on Newer College Dataset, recorded with OS0-128. We always improve the SLAM baseline, a part for *stairs* starting from LeGO-LOAM estimate. Best results are highligthed as **first**, **second**, and **third**.

self-recorded data using a hand-held Ouster OS0-128 LiDAR. The trajectory is about 2 km long, and the dataset is available from our repository.

4.4.3 RGB-D + 3D LiDAR

Thanks to the unified nature of our pipeline, it allows the straightforward integration of multiple heterogeneous sensors. In Sec. 4.3.3, we formulated an optimization problem that estimates the sensor's trajectory $\mathbf{X}_{1:N}$.

In a multiple sensor configuration, we can express the optimization problem as a function of the trajectory of a multi-device platform, to which all sensors are rigidly attached. The multiple sensors setting slightly modify the cost function presented in Eq. (4.3) to include the constant sensor offsets. The cost function for multi-sensor photometric refinement is just the sum of the cost functions of each device. In this section, we report an analysis of our optimization strategy with both RGB-D and LiDAR. For these experiments, we mounted an Intel D455 on the top of an OS0-128 and accurately calibrated using the methodology described in Chap. 3. We recorded some static indoor sequences, each of them containing synchronized grayscale and depth images from D455 and its corresponding intensity and range images from the OS0-128.

We carried out an experiment to evaluate the accuracy and converge basin of our approach in this configuration. To this extent, we perform our photometric alignment strategy using a single sensor frame consisting of a RGB-D and a LiDAR measurement pair. Since we align a sensor frame with respect to itself, we expect the computed estimate to be as close as possible to the identity. The optimization is carried on starting from increasingly wrong initial guesses.

In Fig. 4.7, we show the result of this experiment, starting from single sensor optimization as a baseline. Due to their different characteristics, the two sensors behave very differently during photometric alignment. The 3D LiDAR has an inferior resolution but a very large $360\,\mathrm{deg}$ horizontal FoV. This results in a relatively large convergence basin, but the limited resolution affects the minimum. Conversely, RGB-D, provides a high-resolution image with a much smaller FoV of around $70\,\mathrm{deg}$. This results in better minima at the expense of a smaller converge basin.

We can refine the data in two ways that we name *coupled* and *consecutive*. The coupled approach aims to simultaneously finding the minimum of both photometric errors of RGB-D $F^{\mathrm{r}}(\mathbf{X}_{1:N})$ and LiDAR $F^{\mathrm{l}}(\mathbf{X}_{1:N})$, as follows

$$\mathbf{X}_{1:N}^{\text{coupled}} = \underset{\mathbf{X}_{1:N}}{\operatorname{argmin}} F^{\text{r}}(\mathbf{X}_{1:N}) + F^{\text{l}}(\mathbf{X}_{1:N}) \tag{4.5}$$

The *consecutive* optimization combines the strengths of the two sensors, and operates by first carrying on an optimization where only the LiDAR is used. Subsequently, it uses the solution of this first run to find a minimum for the RGB-D problem.

The results suggest that fusing the two sensors always performs better than single sensors (Fig. 4.7). Specifically, *coupled* is generally more accurate compared to RGB-D only, having the converge basin increased by the LiDAR. Similarly, *consecutive* seems to be the best since it takes full advantage of the large convergence basin of the LiDAR and benefits from the resolution of RGB-D. We believe that this proof of concept opens ways for 3D reconstruction algorithms that symmetrically fuse the two sensors, taking benefits from both.

4.5 Conclusions

This chapter presented a photometric BA methodology that operates both with RGB-D and LiDAR in a unified manner. To the best of our knowledge, our approach is the only open-source BA system that works independently for depth sensors and specifically exploits the photometric capabilities of the LiDAR image. Comparative experiments show that our simple schema performs on par or better compared to existing sensor-specific SOTA approaches without making any assumption about the environment and free from data association. Thanks to our photometric registration methodology's inherent data separation, we develop our software in CUDA and release an open-source implementation. Considering the larger convergence basin and accuracy obtained fusing both RGB-D and LiDAR within our registration strategy, we envision a SLAM/BA pipeline that uses both depth sensors jointly. Furthermore, to complete our universal 3D reconstruction schema, future research would involve finding generic structure representation to optimize both RGB-D and LiDAR measurements.

Chapter 5

Enhancing LiDAR performance: Robust De-skewing Exclusively Relying on Range Measurements

5.1 Introduction

This chapter introduces the rolling-shutter measurement distortion that characterize all spinning LiDARs. Countrary to the rest of the thesis, this chapter will focus on 2D LiDARs mounted on robotic platforms constrained to planar motion. This simplified scenario provides a clearer setting in which the impact of motion distortion can be isolated and studied in detail.

This effect arises from the non-instantaneous acquisition time and it is roughly proportional to the translational and rotational velocities of the sensor during acquisition. Atlough for industrial-grade LiDARs and for typical robot velocities (i.e., a linear velocity of $\approx$ 1m/s and an angular velocity $<$ 1rad/s), this effect may appear neglegible, we will show that it does impact registration accuracy, and consequently both mapping and tracking performance of a Simultaneous Localization and Mapping (SLAM) system[1].

In this work we focus on low grade hobby LiDARs such as the InnoMaker-LD-06 or the RPI-Lidar A1 that can be bought for less than 100 € at the time of writing. The major shortcoming of these devices is their relatively slow rotational speed that results in a full sweep of measurements becoming available between 5 to 13 Hz. In contrast to more expensive models these devices are not equipped with an inertial unit or other means to estimate the proprioceptive motion of the sensor while it moves.

In spinning LiDARs, the scan acquisition is not instantaneous, hence the origin of the beam in the world frame changes for each sensed range. Assuming that all measurements gathered during one rotation are sampled at the same time results in a distorted or skewed scan. Since most perception blocks in a navigation system, such as SLAM or localization treat the scans as rigid bodies, the effect of skewing leads to undesirable decay in accuracy that can be so severe to result in failures as analyzed by Al-Nuaimi *et al.* [1]. A common way to compensate for the motion of the LiDAR while it rotates is to integrate external proprioceptive measurements (dead reackoning and/or Inertial Measurement Unit (IMU)), to

[1]The severity of the distortion is intrinsically linked to the LiDAR acquisition time. In the mentioned application, higher acquisition rates (i.e., 75-100 Hz) mitigates the effect.

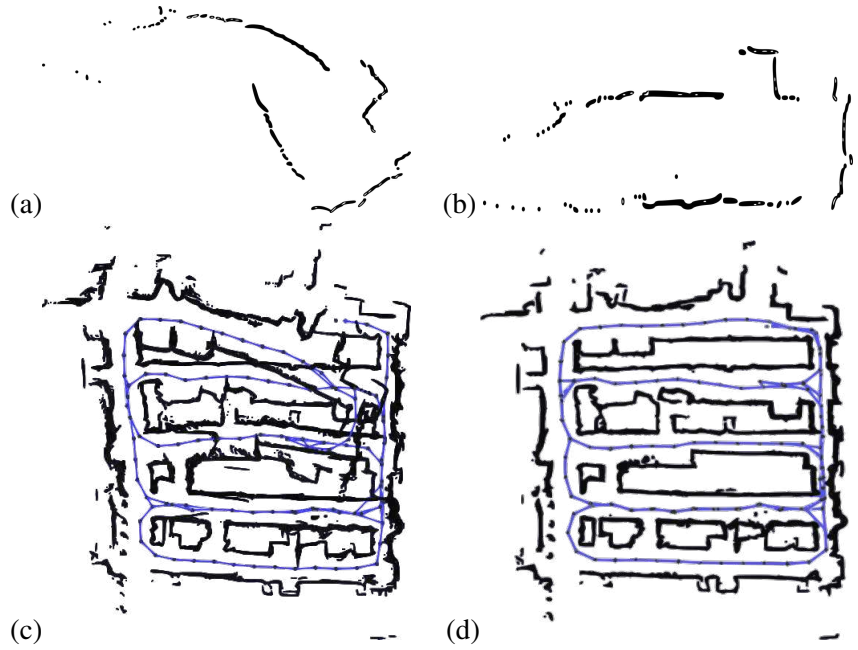


Figure 5.1. Effects of a the de-skewing. Top: (a) unprocessed scan acquired by a moving robot, (b) same scan de-skewed by using our approach. Bottom: (c) map build by a SLAM algorithm on skewed scans, (d) map obtained by the same SLAM algorithm using scans de-skewed with our method.

estimate the origin and orientation of the laser beam each time a range is acquired [48, 154, 111]. High-end 3D LiDARs already contain a synchronized IMU, which is not available on the inexpensive models previously mentioned.

This chapter describes an approach to estimate the velocities at which the robot where the LiDAR is mounted moves, based *solely* on the range measurements, that are processed as a stream. Our method is based on a non-rigid plane-to-plane registration algorithm, where the velocity of the platform carrying the sensor is estimated by maximizing the geometric consistency of the stream of ranges. We conducted several statistical experiments on real and synthetic data. Results show that our de-skewing method is effective estimating the motion of the robot, enhancing the quality of SLAM and localization estimates, at negligible computation. We release an open source C++ implementation of our method as a plug-and-play component that can be added to existing 2D LiDAR pipelines. In Fig. 5.1 we illustrate the effects of our approach both on individual measurements, and when used within a more complex SLAM system. This chapter is divided as follows. Sec. 5.2 introduces a review of the State Of The Art (SOTA), subsequently we describe in detail our de-skewing method for 2D LiDAR data (Sec. 5.3). The chapter concludes with a broad set of experiments aimed at measuring the performances of our method under different operating conditions (Sec. 5.4), and finally, we draw some conclusions on the benefits and the limitations of our approach.

5.2 Related Works

In this section, we review the most recent approaches to de-skew LiDAR data. Existing approaches for de-skewing mostly use either wheeled odometry or IMU to estimate the

relative motion of the sensor/robot while a scan is being acquired. This process involves motion integration relying on estimators that process the raw proprioceptive data such as a filter or an integrator that directly provides an estimate of the sensor position each time a single range is measured.

These methods fall in the class of *loosely-coupled*, since they do not use the LiDAR information to refine the proprioceptive estimate. Among this class, we find the work of Tang *et al.* [122] that proposes to match subsequent scans to compute the relative motion, where the initial relative orientations are provided by an IMU. Subsequently, He *et al.* [48] proposed a method to estimate relative motion between IMU poses and de-skew subsequent ranges by using these smoothed poses.

In contrast to loosely-coupled approaches, *tightly-coupled* methods jointly process LiDAR and IMU. These methods are typically based on either smoothing or filtering. Wei *et al.* [145] uses a pre-integration scheme to estimate the LiDAR's ego-motion based on the inertial measurements, and updates the IMU biases in the correct step of an iterative EKF whenever a scan is completed. Shan *et al.* [112] includes the states of IMU in the factor-graph, updating IMU biases through the optimization process, adapting a well-known computer vision work [36] to the LiDAR case. These ideas have been further extended in [137, 147] where the authors resort to full factor-graph optimization to obtain a state configuration that is maximally consistent with all the IMU measurements and LiDAR ranges.

For coupled approaches, accurate time synchronization between the sensors is essential. Unfortunately, this is not straightforward to obtain on inexpensive small devices due to the unpredicted communication latencies that affects the USB channels. These issues, however, can be completely avoided when using only LiDAR data to perform de-skewing. At their core LiDAR only methods have a registration algorithm, which aims to compute the velocities of the sensor while the robot moves. The basic intuition is that if the correct velocities were found, the sequence of range measurements could be assembled in a maximally consistent scan. In this context, Moosmann *et al.* [83] propose to linearly interpolate scans between the last known pose, and the currently estimated pose, de-skewing the scan at both poses. However, this double approximation might hinder the accuracy when the initial registration fails, resulting in even worse estimates. Al-Nuaimi *et al.* [1] propose a weighting schema based on their adapted Geometric Algebra LMS solver, where they assume that consecutive relative translation and angular motion are equal, which might be inaccurate due to the existence of drift and friction.

In this chapter, we propose a planar LiDAR-only approach that addresses the de-skewing problem by continuously registering the the sequence of range measurements. The registration is carried on by estimating the LiDAR's planar velocity, which iteratively minimizes a plane-to-plane metric between the de-skewed endpoints. Our results demonstrate that our methodology is accurate in estimating the velocity of the robot, only based on the laser measurements.

5.3 Method

Our method leverages some mild assumptions about the motion of the sensor mounted on the robot and the structure of the environment to operate on a continuous stream of range measurements. Specifically we assume to have a 2D LiDAR sensor with slow acquisition

rates (< 13 Hz), for which we define each measured point as triplet $\mathbf{z}_i = (r_i, \alpha_i, t_i)$. Here, r_i is the range, α_i is the angle of the laser beam with respect to the origin of the sensor and t_i is the timestamp. We assume the environment consists of a locally smooth surface and that the robot acceleration is close to 0 during a LiDAR beam revolution.

Our method estimates these velocities by registering the stream of measurements $\{\mathbf{z}_i\}_{i=1}^{M}$ onto itself. The most likely velocities are the ones that if applied for de-skewing renders the measurement maximally consistent. Consistency is measured by a plane-to-plane metric applied between the corresponding LiDAR endpoints. Formally, let $\mathbf{x} = (v \ \ \omega)^T$ be the translational and angular velocity of the robot; We want to estimate $\mathbf{x}^*$, such that

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{\langle i,j \rangle \in \mathcal{C}} \rho \left(\|\mathbf{e}(\mathbf{x}, t_i, t_j)\|^2 \right), \tag{5.1}$$

where ρ denotes the Huber robust estimator, and $\mathbf{e}(\mathbf{x}, t_i, t_j)$ denotes an error vector between two planar scan patches computed around two corresponding range measurements at time t_i and t_j. The optimization step estimates new velocities $\mathbf{x} = (v \ \omega)^T$ under the current set of correspondences $\mathcal{C}$.

Our algorithm is an instance of Iterative Closest Point (ICP) since it proceeds by alternating between the data association and optimization steps. In the data association step, the corresponding endpoints are found using nearest neighbour strategy based on current velocity estimates. In the optimization step, the velocities are refined from the new correspondences found by the data association.

5.3.1 Velocity Based De-skewing

Whereas our approach can be applied to more complex kinematics, for the sake of simplicity we detail the common case of a unicycle mobile base. Our goal is to estimate the location $\mathbf{T}(\mathbf{x}, t_i) = \left[\mathbf{R}(t_i)|\mathbf{t}_i\right] \in \mathbb{SE}(2)$ of the mobile base at time t_i where $\mathbf{R}(t_i) \in \mathbb{SO}(2)$, and $\mathbf{t}_i = (x_i, y_i)^T$, assuming it starts from the origin and progresses with constant translation and angular velocities over sampling time $\mathbf{x} = (v \ \ \omega)^T$. After a time t_i, the robot would have traveled for a distance $l_i = v \cdot t_i$ and its angle is $\theta_i = \omega \cdot t_i$. The base will move along an arc of radius R, such that:

$$R \ = \ \frac{v_i}{\omega_i} = \frac{l_i}{\theta_i}. \tag{5.2}$$

From this consideration, we can easily compute $\mathbf{T}$ as

$$\mathbf{T}(\mathbf{x}, t_i) = \begin{pmatrix} R \sin\theta_i \\ R(1 - \cos\theta_i) \end{pmatrix} = \underbrace{v \cdot t_i}_{l_i} \begin{pmatrix} \frac{\sin\theta_i}{\theta_i} \\ \frac{1-\cos\theta_i}{\theta_i} \end{pmatrix} \tag{5.3}$$

Assuming the sensor (LiDAR) is located at the center of the mobile base, if at time t_i the beam has a relative angle α_i and reports a range measurement r_i, we can straightforwardly find the 2D laser endpoint $\mathbf{p}_i$ as follows:

$$\mathbf{p}_i(\mathbf{x}, t_i) \ = \ \mathbf{R}(\theta_i + \alpha_i) \begin{pmatrix} r_i \\ 0 \end{pmatrix} + \begin{pmatrix} x_i \\ y_i \end{pmatrix}. \tag{5.4}$$

Here $\mathbf{R}(\theta_i + \alpha_i) \in SO(2)$ is 2D rotation matrix of $\theta_i + \alpha_i$. Applying this process to all the measurements and mapping all reconstructed points back to the pose where the acquisition of the current scan was started, results in the desired de-skew operation.

5.3.2 Error Metric

To evaluate a plane-to-plane distance the algorithm needs to compute the normal vectors, which are based on the endpoints. Since the position of the endpoints is a function of the estimated velocities $\mathbf{x}$, also the normal vectors are. Hence the algorithm has to recompute both endpoints and normals at each iteration. Furthermore, when subsequent endpoints are too close, the noise affecting the range might result in an unstable normal vector, which hinders the error metric. To lessen this effect, before each iteration, we regularize the scan to retain only temporally subsequent measurements that are sufficiently far from each other to ensure a stable normal. In our experiments, we set this threshold to 0.15 m. Similarly, if there is a large distance gap between two subsequent endpoints (>0.4 m), it is likely that the surface is not continuous at that point, hence we drop those measurements too.

At the end of this regularization step, we end up with a set of reasonably stable measurements we can use for the remainder of the computation. For each temporally subsequent pair of endpoints, we compute a planar patch $\mathbf{m}_i = (\mathbf{c}_i, \mathbf{n}_i, t_i)$, characterized by a center $\mathbf{c}_i$, a normal vector $\mathbf{n}_i$, and a timestamp t_i, such that:

$$\mathbf{c}_i = \frac{1}{2}\left(\mathbf{p}_{i+k} + \mathbf{p}_i\right), \tag{5.5}$$

$$\mathbf{n}_i = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \frac{\mathbf{p}_{i+k} - \mathbf{p}_i}{\|\mathbf{p}_{i+k} - \mathbf{p}_i\|}, \tag{5.6}$$

$$t_i = \frac{1}{2}\left(t_{i+k} + t_i\right). \tag{5.7}$$

Here, the index $k > 0$ accounts for endpoints suppressed during regularization. If two planar patches $\mathbf{m}_i$ and $\mathbf{m}_j$ corresponds to the same portion of the environment, we can calculate an error vector $\mathbf{e}(\mathbf{x}, t_i, t_j)$ accounting for both their differences in position and orientation. The error vector is a function of the velocities, since the patches $\mathbf{m}_i$ and $\mathbf{m}_j$ are computed based on the endpoints. The latter is related to the velocities by Eq. (5.4).

Let $\mathbf{e}(\mathbf{x}, t_i, t_j) \in \mathbb{R}^3$ be the error vector, whose components are defined as follows:

$$\mathbf{e}(\mathbf{x}, t_i, t_j) = \begin{pmatrix} \frac{1}{2}(\mathbf{c}_i - \mathbf{c}_j)^T(\mathbf{n}_i + \mathbf{n}_j) \\ \mathbf{n}_j - \mathbf{n}_i \end{pmatrix}. \tag{5.8}$$

Here the first dimension accounts for the distance between two corresponding planar patches $\mathbf{m}_i$ and $\mathbf{m}_j$ projected along the average of their normals. The other two account for the difference between the normal vectors. Eq. (5.8) is differentiable in the velocities $\mathbf{x}$, hence we can minimize Eq. (5.1) by Iterative Reweighed Least-Squares (IRLS), as described in Sec. 2.4.2 .

5.3.3 Data Association

To determine the correspondence, we proceed at each iteration by de-skewing the sequence of measurements, to get a set of updated endpoints $\mathbf{p}_i$. Subsequently, we apply the regularization to discard those measurements whose endpoints fall either too close or too far from their temporal neighbors. This gives us a set of sequential stable measurements we can use to extract the planar patches.

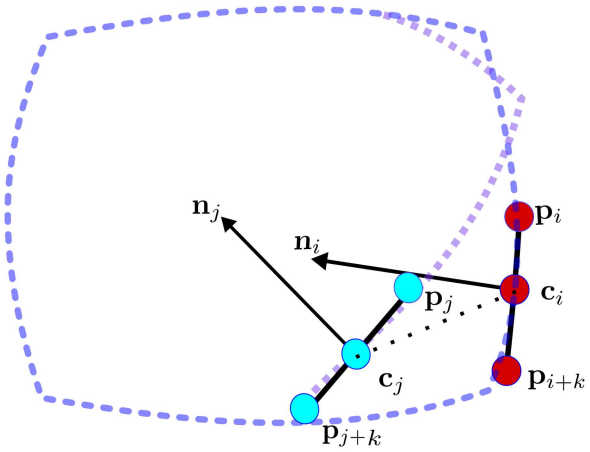


Figure 5.2. Illustration of the planar patches calculation and of the correspondence search between two matching patches $\mathbf{m}_i$ and $\mathbf{m}_j$.

Once this is done, for each $\mathbf{m}_i$ we seek for those other patches $\mathbf{m}_j$ that fulfill all the following criteria:

- their centers are close enough $|\mathbf{c}_i - \mathbf{c}_j| < \tau_{\mathbf{c}}$,
- their normals are sufficiently parallel $\mathbf{n}_i \cdot \mathbf{n}_j > \tau_{\mathbf{n}}$,
- their timestamp are sufficiently distant $|t_i - t_j| > \tau_t$.

Within this set we select as correspondence for $\mathbf{m}_i$, which is the $\mathbf{m}_j$ having the smallest projective distance $(\mathbf{c}_i - \mathbf{c}_j)^T(\mathbf{n}_i + \mathbf{n}_j)$. Fig. 5.3 shows subsequent iterations of our proposed approach.

5.4 Experiments

In this section, we present quantitative evaluations to demonstrate the accuracy of the velocities estimated by our approach, and it's effect on the processed data, using only LiDAR measurements. To this extent, we carried out statistical experiments under changing velocities.

The goal of the first experiment is to compare the velocities of the robot calculated by our approach with the ground truth applied.

Virtual synthetic scans were created with a skewing effect based on the addressed velocities. We proceed by de-skewing twice, the first time we compute the ground truth measurement by setting the applied velocities, and the second time we compute the de-skewed scan by using our velocity estimate. The consistency is measured as the distance between corresponding endpoints in the two scans.

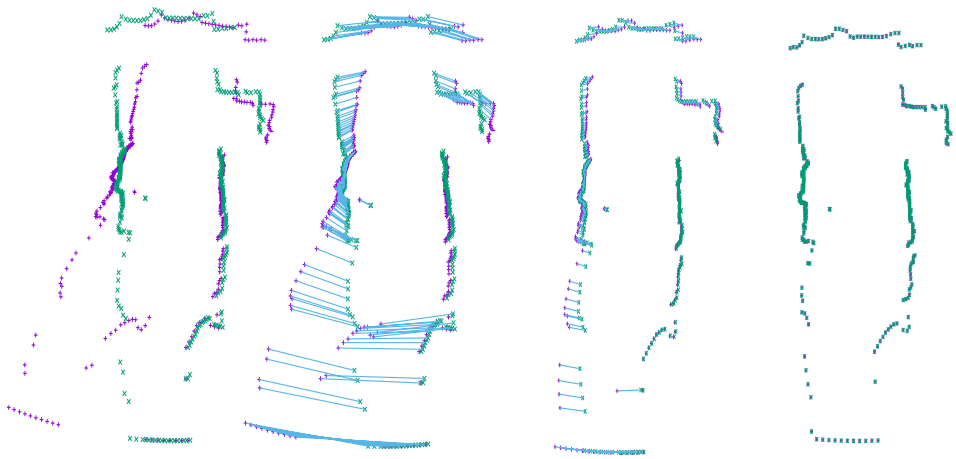


Figure 5.3. Iterative evolution of the de-skewing process. Purple points represent skewed points, green points are the ground truth points, and the blue lines represents the data association. (a): initial configuration, instance of full scan acquisition, with RMSE = 0.4 m. (b): after 4 iterations of processing (de-skewing) the raw scan with the proposed approach, decreading error to RMSE = 0.2 m. (c): after 20 iterations, ending up with a de-skewed scan closer to the ground truth , with RMSE = 0.074 m.

We repeated this experiment on the real robot and the velocity plots in Fig. 5.4 confirm that our system can effectively recover the robot's velocities, correctly de-skewing the scan, in real-world indoor scenarios.

5.4.1 SLAM

In our final experiment, we used our de-skewing mechanism to pre-process the input of our 2D LiDAR SLAM system [18]. We run SLAM on three different inputs: the raw (skewed) scans, the ones de-skewed by using the proposed approach, and finally on the data de-skewed by using the ground truth. We simulated different exploration runs of the same environment while changing the velocity bounds of the robot. Fig. 5.5 illustrates the three different maps obtained.

Furthermore, we measure the Absolute Trajectory Error (ATE) between the ground truth trajectory computed with by the simulator and the ones estimated by SLAM using the two types of scans. ATE measures the distance between corresponding points of the trajectories, after computing a transformation that makes them as close as possible. For trajectory registration we used Horn method [50], while the corresponding poses are associated based on the timestamps. The results are summarized in Tab. 5.2. Consistently, the trajectories obtained by using de-skewed data are substantially close to the ground truth than their skewed counterpart as shown in Fig. 5.5d. This is confirmed by the maps generated based on the trajectories and illustrated in Fig. 5.5. These results are consistent with the intuition that using skewed scans when doing SLAM is not recommended, since these measurements might induce systematic unrecoverable errors in the registration process.

$\omega(rad/s)$ \ $v(m/s)$	-2.000	-1.000	-0.500	0.500	1.000	2.000
-2.000	-1.936±0.090 -1.952±0.081 **0.090**/0.404	-0.950±0.068 -1.933±0.115 **0.083**/0.399	-0.471±0.050 -1.958±0.066 **0.059**/0.351	0.470±0.047 -1.962±0.052 **0.061**/0.414	0.962±0.052 -1.952±0.070 **0.055**/0.460	1.910±0.092 -1.932±0.103 **0.081**/0.579
-1.000	-1.890±0.131 -0.949±0.069 **0.067**/0.297	-0.979±0.031 -0.986±0.023 **0.058**/0.308	-0.477±0.035 -0.973±0.037 **0.055**/0.296	0.482±0.034 -0.979±0.026 **0.049**/0.354	0.946±0.069 -0.953±0.059 **0.054**/0.336	1.897±0.073 -0.950±0.056 **0.062**/0.399
-0.500	-1.929±0.085 -0.487±0.020 **0.040**/0.308	-0.978±0.063 -0.493±0.022 **0.035**/0.345	−0.479±0.054 -0.477±0.035 **0.041**/0.188	0.492±0.033 -0.495±0.009 **0.043**/0.200	0.935±0.064 -0.484±0.026 **0.060**/0.218	1.906±0.062 -0.482±0.022 **0.084**/0.338
0.500	-1.955±0.024 0.483±0.012 **0.119**/0.158	-0.954±0.044 0.495±0.018 **0.029**/0.140	-0.474±0.071 0.476±0.045 **0.044**/0.112	0.481±0.048 0.485±0.026 **0.052**/0.132	0.969±0.042 0.488±0.020 **0.059**/0.161	1,967±0.036 0.496±0.012 **0.159**/0.271
1.000	-1.844±0.130 0.933±0.077 **0.063**/0.261	-0.976±0.064 0.976±0.053 **0.063**/0.225	-0.472±0.057 0.940±0.053 **0.024**/0.231	0.495±0.028 0.992±0.015 **0.055**/0.302	0.969±0.059 0.970±0.048 **0.058**/0.303	1.98±0.038 0.992±0.017 **0.039**/0.335
2.000	-1.906±0.116 1.904±0.142 **0.074**/0.416	-0.941±0.076 1.919±0.120 **0.071**/0.368	-0.465±0.081 1.912±0.104 **0.081**/0.358	0.480±0.065 1.905±0.137 **0.075**/0.435	0.940±0.083 1.922±0.116 **0.076**/0.494	1.947±0.069 1.963±0.069 **0.091**/0.424

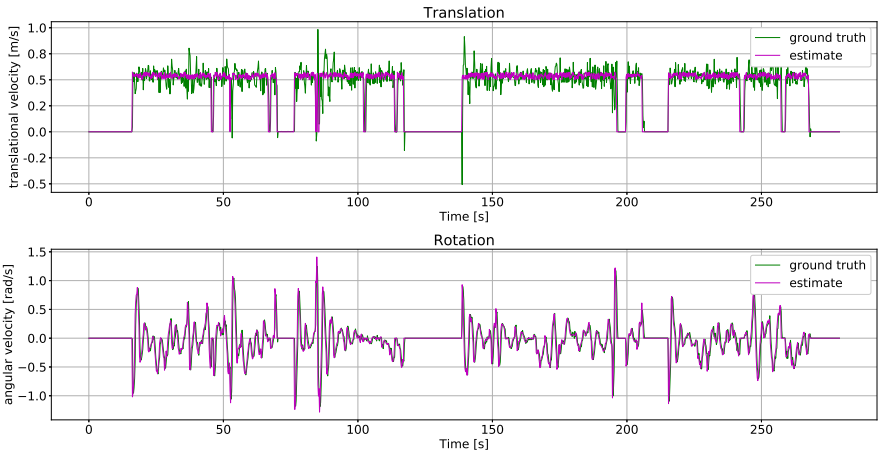
Table 5.1. Evaluation of the proposed approach using different combinations of translation and angular velocities. In each cell, the first row is the estimated translational velocity $\bar{v}$, while the second represents the angular velocity $\bar{\omega}$. The third row is the point-to-point Root Mean Square Error (RMSE) for the de-skewed/skewed clouds respectively.

Velocities		ATE(m) ↓	
$v_{\max}$ $[m/s]$	$\omega_{\max}$ $[rad/s]$	w/o deskewing	deskewed
0.5	0.5	0.329	**0.113**
0.5	1.0	0.246	**0.232**
1.0	1.0	1.018	**0.977**
1.0	1.5	1.586	**1.520**
2.0	2.0	11.18	**1.805**

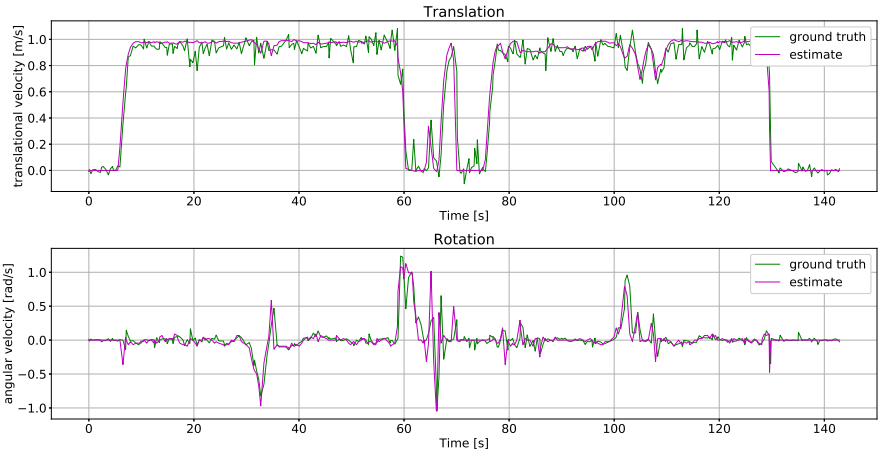
Table 5.2. SLAM comparison. The table reports the absolute trajectory error for varying translational and rotational velocities. Results are highligthed as **first** , and **second** .

5.5 Conclusions

In this chapter, we presented a simple and effective planar LiDAR de-skewing mechanism based on plane-to-plane registration criteria. To the best of our knowledge, this is the only de-skewing pipeline that does not rely on additional sensors (i.e. IMU, wheel encoders), targeting specifically low grade LiDARs with slow acquisition rates (< 15 Hz), and enhancing the quality of the scan. For the future work we are planning to integrate the proposed approach in more complex systems, while generalizing the approach for 3D data.



(a) Velocity estimated using synthetic data



(b) Velocity estimated using real data

Figure 5.4. Evolution of estimated and real velocities.

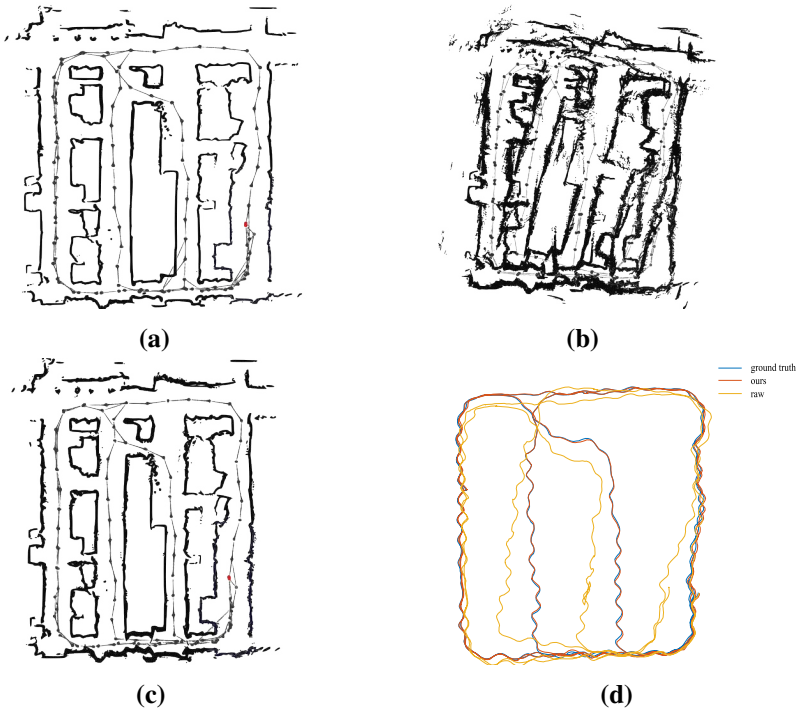


Figure 5.5. SLAM Results. (a) Ground truth map (reference). (b) Map constructed using raw scans acquired directly from the LiDAR. (c) Map constructed using processed LiDAR measurements de-skewed by our proposed approach. (d) Estimated trajectories from SLAM, produced by scan matching between cosecutive measurements.

Chapter 6

VBR: A Vision Benchmark in Rome

6.1 Introduction

The Computer Vision and Robotics communities relies on standard datasets to benchmark and improve new research works since the early days. When ground truth data is accessible for a specific task, the researchers may devise appropriate metrics to evaluate the accuracy of their algorithm's results. With the rapid advancement of machine learning, datasets equipped with ground truth have become essential inputs for algorithms designed to learn intricate, non-parametric models. After KITTI, several other multi-sensor datasets [77, 109, 64, 157] have been presented, but no one seemed to have the same impact on the robotics and computer vision community as the original work [40].

Whereas the merits of KITTI are undisputed, and the core ideas are still valid, the dataset shows its years. The available sensors in the last decade improved significantly, and the same holds for computing devices and ground truth systems. Perhaps the main shortcoming of many datasets [40, 64, 146, 13] is the limited positional ground truth that is purely based on RTK-GPS and IMU and suffers from synchronization issues. In addition to that, the work is targeted at autonomous driving, hence, the data cover only road-like scenarios.

Other works aimed at addressing other aspects, such as seasonal changes [77], offering hand-held motion with a more accurate ground truth [99]. Still, to our knowledge, none of the recent datasets seem to address multiple issues. In this work, we present a contribution that aims to approach all these aspects simultaneously. We propose 6 sequences acquired with a hardware-synchronized sensor setting consisting of a 3D LiDAR, a stereo camera with a large baseline, an RTK-GPS, and an inertial sensor. Our data covers some of the most characteristic areas of Rome, spanning over 40 km of trajectory in almost 4 hours of recording. The raw data has a footprint of about 2TB. The sequences have been recorded in different environments, covering urban, forest, and indoor scenarios, using the same kind of sensors but at different frequencies and modalities. Heterogeneous sequences have been intentionally recorded to create a more challenging dataset, preventing domain overfitting. Moreover, we illustrate a procedure for obtaining highly accurate ground truth in large environments combining an RTK-GPS with a Bundle Adjustment schema on the LiDAR data to obtain precise trajectories. The accuracy of our ground truth, validated with a Total Station is ± 3 cm along an indoor/outdoor trajectory of 1.5 km. Each sequence is split into a training subsequence with available ground truth and, a test subsequence for benchmarking. Furthermore, the benchmark leaderboard is available at our website:

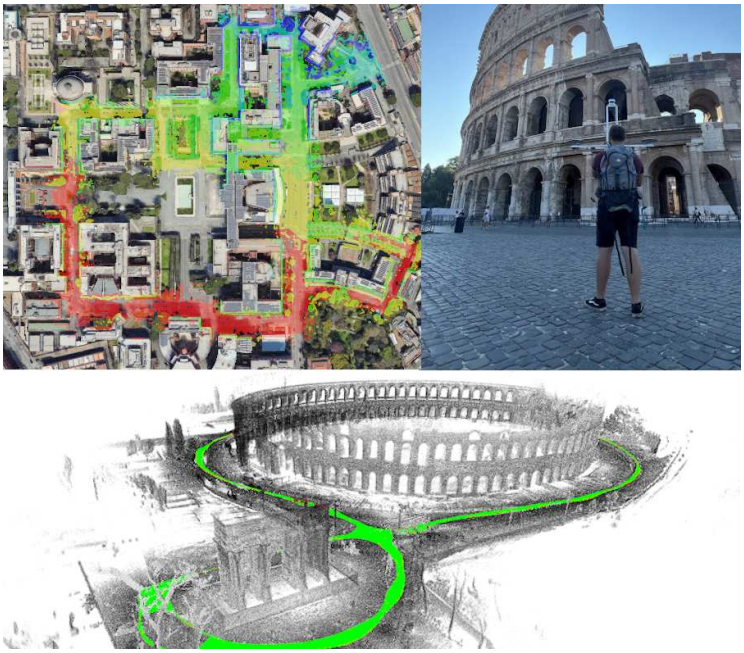


Figure 6.1. A summary of our dataset. Data illustrating some of the sequences recorded (top). 3D mapping done with of our ground truth (bottom).

https://rvp-group.net/slam-benchmark.html

6.2 Related Works

Within the domain of SLAM and 3D reconstruction, several datasets have played pivotal roles in benchmarking and advancing autonomous robotics systems and algorithms. These datasets vary in terms of their operational contexts, sensor configurations, and the accuracy of their associated ground truth data. One of the seminal car datasets in this field is KITTI. Its strength is providing different benchmarks (i.e. visual odometry, optical flow, stereo matching, and object detection). The KITTI datasets are publicly available and divided into training and evaluation sequences, fostering fair comparisons among the approaches. Despite being made available for more than a decade nowadays, KITTI is the most used benchmark in robotics perception and computer vision. However, KITTI presents synchronization issues between IMU readings and images, the ground truth data for visual odometry is produced only by fusing RTK GPS receiver and IMU, and the hardware used for recording data is nowadays outdated (Fig. 6.2, Fig. 6.3). Still, KITTI was pivotal in the development of many popular SLAM methods. Oxford RobotCar [77] is another noteworthy car dataset. In contrast to KITTI, it distinguishes itself by featuring the longest sequences among the datasets. Yet, the ground truth in the Oxford RobotCar dataset relies only on partial GPS and INS data, which makes the baselines unreliable for benchmarking the accuracy of SLAM and localization methods.

In contrast to datasets collected from ground-based vehicles, certain research efforts have focused on data acquisition from micro aerial vehicles. For instance, the EuRoC dataset [14] stands out for its use of synchronized hardware and a laser tracking system to attain

accurate ground truth data. However, the dataset does not provide LiDAR aquisition but only a stereo-camera and an IMU. In addition, data is recorded only in industrial environments.

Recent advancements in handheld datasets, exemplified by Newer College [99] and Hilti [157], have achieved high levels of accuracy in ground truth generation through the use of terrestrial industrial-grade LiDARs. This technique involves acquiring a prior map and registering LiDAR point clouds using a localization approach. Nevertheless, a notable limitation in this case is the impracticality of applying this method to large-scale scenarios, which constrains its broader utility.

This chapter describes a diverse and heterogeneous dataset encompassing a wide range of environments. Our design accommodates various robotic platforms, including quadrupeds, quadrotors, and autonomous vehicles, making it a versatile resource for the robotics community.

We maintain hardware synchronization to ensure data accuracy and reliability and employ stereo cameras with a wide baseline to capture robust visual information. Furthermore, we provide an accurate 6-Degrees Of Freedom (DOF) ground truth even in large scale scenarios.

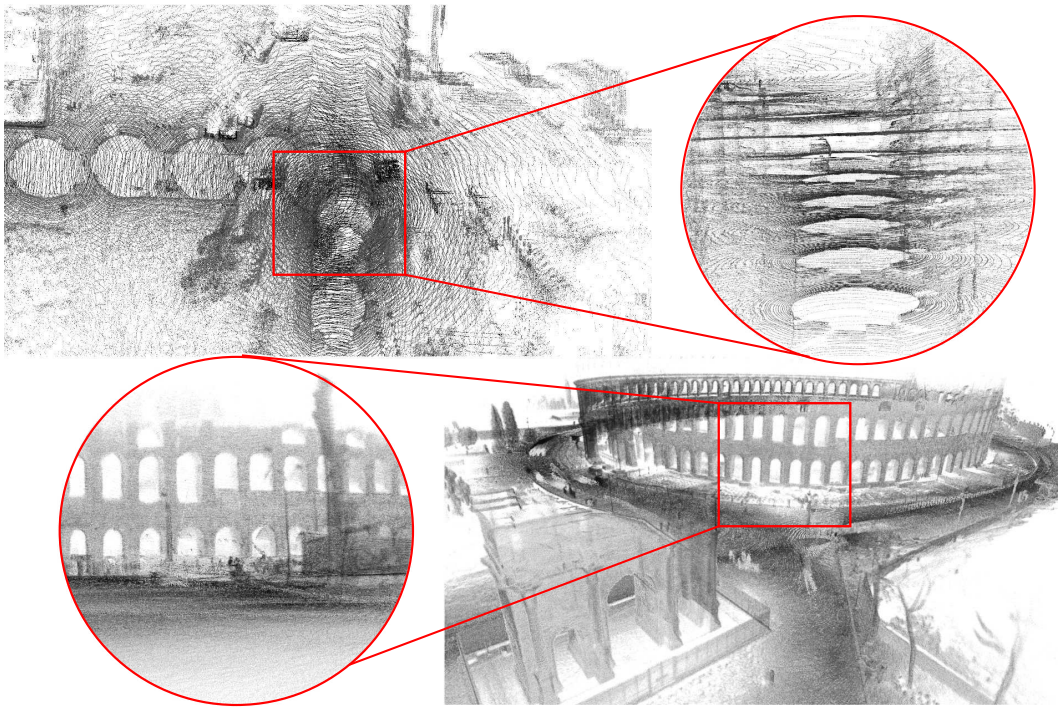


Figure 6.2. Comparison between LiDAR clouds. First, attached to ground truth trajectories of KITTI (up) and ours (down). The zoom shows the elevation view.

6.3 Method

Creating comprehensive and authentic benchmarks for the tasks mentioned earlier is challenging. These challenges encompass collecting vast data in real-time, calibrating different sensors operating at various speeds, producing accurate ground truths with minimal oversight, and choosing the right sequences and frames for every benchmark. The following

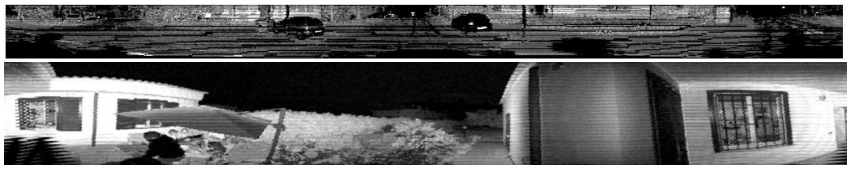


Figure 6.3. Spherical projection (described in Sec. 2.3.2) of the KITTI LiDAR points into an image plane (up) and of our LiDAR. Altough deskewed, KITTI LiDAR sensor exhibit lower resolution compared to modern industrial-grade LiDARs.

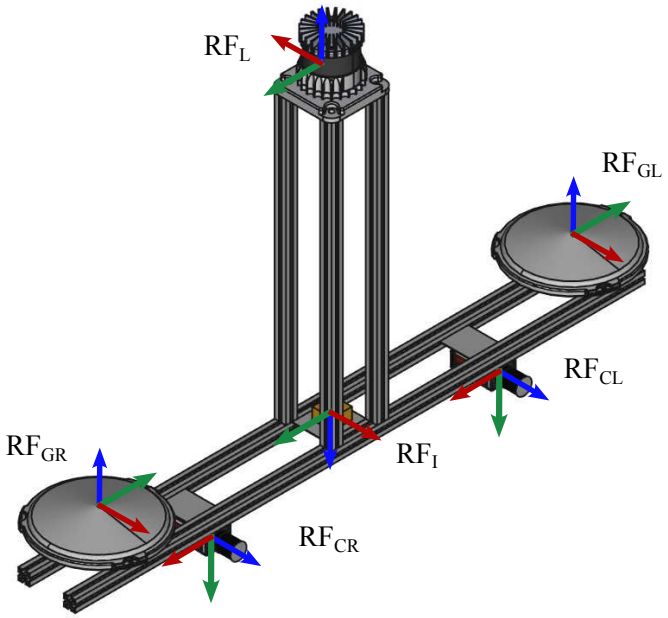


Figure 6.4. Sensor setup and reference frames. Our ground truth is expressed in the LiDAR reference frame $\mathrm{RF_L}$.

section delves into our approaches to address these issues.

6.3.1 Sensor Setup

Our sensor system is illustrated in Fig. 6.4 and consists of two RGB cameras, a 3D LiDAR, an RTK-GPS, and an IMU. The cameras are two global shutter Manta G-145 capturing in RGB and arranged in a wide stereo fashion, with a baseline of approximately 50 cm. The cameras have a horizontal FoV of $45°$ and a vertical one of $40°$. During the acquisition, we enable auto white-balance and auto-exposure, while maintaining a fixed focus. The maximum exposure is fixed at 20 ms. Each image is 1388×700 pixels and stored in Bayer pattern to reduce the memory footprint without losing information.

Two LiDARs were employed, tailored to the specific motion characteristics of the captured sequences. For hand-held sequences, an Ouster OS0-128 was used. This sensor offers a maximum range of 55 meters and a vertical FoV of $90°$ spanned by 128 beams. For car sequences, we used an Ouster OS1-64. This sensor provides a longer maximum range of 120 meters, a narrower vertical FoV of $45°$ spanned by 64 lasers.

Dataset	Accurate GT	Indoor	Outdoor	Various Motion	Large Scale	Benchmark
KITTI [40]			X		X	X
EuRoC [14]	X	X				
Oxford RobotCar [77]			X			
ETH3D [109]	X	X				X
MulRan [64]	X		X		X	
Newer College [99]	X		X			
Hilti [157]	X	X	X	X		X
Kitti-360 [71]	X		X		X	X
Ours	X	X	X	X	X	X

Table 6.1. Comparison of popular datasets in robotics perception and computer vision related to odometry estimation and SLAM. For "Accurate GT" we mean any ground truth recorded with motion capture, Laser Total Station, or globally refined.

Sensor	Type	Details	Rate
LiDARs	Ouster, OS0-128	Vertical FOV: 90° Horizontal Res: 2048	10 Hz
	Ouster, OS1-64	Vertical FOV: 45° Horizontal Res: 1024	20 Hz
Cameras	Manta G-125B/C	Global Shutter Stereo configuration Wide baseline	20 Hz 30 Hz
IMU	SBG Ellipse-E	GNSS synchronization 0.05° precision	100 Hz

Table 6.2. Summary of the devices in our sensor setup.

The Inertial Measurement Unit (IMU) is an SBG Ellipse-E, with 0.05 ° of roll/pitch accuracy, whose firmware supports GNSS integration. The RTK-GPS antennas are two Septentrio PolaNt-x MF GNSS antennas mounted in differential configuration. The GPS receiver supports multi-frequency GPS, GLONASS, Galileo, BeiDou, QZSS, NavIC, Compass and L-band signal reception with an accuracy open-sky condition of 0.6 cm horizontally and 1 cm vertically. All sensors are rigidly attached to an aluminum frame. The relative position of the sensor is the same for both the hand-held datasets and for the driving datasets. Fig. 6.1 shows the sensor placement on the car. Tab. 6.2 summarizes the devices used in our system.

6.3.2 Calibration

The accuracy of intrinsic and extrinsic sensor calibration is fundamental in achieving dependable ground truth data. Our calibration process is outlined below.

Initially, we calibrate the stereo camera intrinsically and extrinsically. Subsequently, we determine the $\mathbb{SE}(3)$ parameters that connect the coordinate systems of the laser scanner within the right camera. Finally, we align the LiDAR /camera system with GPS/IMU reference frame. To calibrate the camera's intrinsic and extrinsic parameters, we use an A3 checkerboard. Keeping the camera steady, we move the checkerboard and detect its corners in the calibration images. Minimizing the average reprojection error allows us to find optimal parameters for our setup [11]. Using the same measurements, we employ the

LiDAR-RGB camera calibration methodology described in Chap. 3 to find the extrinsics between the LiDAR and the right camera ($\mathrm{RF}_{\mathrm{CR}}$). For each recorded sequence we acquire the calibration data which are public available.

Determining the relative pose between the GPS/IMU and the LiDAR relies on the sensor systems' motion, since the two devices cannot observe a common target. To this extent, we recover a trajectory from the LiDAR/camera system using a LiDAR odometry based on point-to-plane ICP. During the process, we ensure a wide range of orientations and translations essential for addressing the minimization issue. This technique is known as *hand-eye* calibration [30] and aims at computing the sensor offset that results in the maximum overlap between two LiDAR trajectories: the one computed by the odometry, and the one obtained by computing the LiDAR motion from the GPS measurements (after applying the estimated offset).

6.3.3 Synchronization

A key challenge during acquisition involves synchronizing sensors to establish a shared temporal reference. Within our platform, two separate subsystems are at play: one comprises the LiDAR and RGB stereo pair, while the other includes the GNSS receiver and IMU. In the first system, the LiDAR takes on the role of the master, generating synchronization pulses during its acquisition phase based on angle data from its encoder. For hand-held use, the LiDAR records at 10 Hz, and the synchronization pulse activates every 120 degrees, resulting in a 30 Hz signal. Moreover, in the automotive setup, the LiDAR operates at 20 Hz, with the synchronization pulse set to trigger once per revolution. The signal triggers the frame acquisition for both cameras, leading to sub-millisecond synchronization between the frames. Once the frames are received, their timestamp is overwritten with one of the LiDAR data packets received when the encoder was at the trigger angle. This ensures an accurate hardware synchronization between the two sensors[1].

In contrast, the GNSS-IMU system relies on Pulse Per Second (PPS) protocol for synchronization, which is directly addressed by the IMU firmware. The streams from the two subsystems are merged together by performing an offline time synchronization to determine the difference between the internal clocks of GPS and LiDAR. We exploit the internal LiDAR IMU measurements to compute the temporal shift respect to the IMU, using cross-correlation. This technique allows us to obtain a maximum of 5 ms error considering possible un-observable phase shift between the IMUs signals that come every 10 ms. A temporal drift also affects the internal clocks of the LiDAR and GPS. In the longest sequences, we observed a maximum shift between the two clocks of about 10 ms, which is neglectable compared to the scan/image frequency and the velocity of the sensor.

6.3.4 Ground truth generation

Generating accurate trajectories is the main objective of this work; hence, major attention was dedicated to this task. Some work produces very accurate ground truth using ICP localization within 3D Total Station reconstruction ([99], [157]). This is not always possible when moving in a very large environment. In such cases, a GNSS RTK system is usually

[1]Altough the LiDAR we employed supported mutually exclusive IEEE PTP synchronization or encoder trigger, we opted for the latter to ensure matching LiDAR-RGB Field Of View (FOV) captures.

used. These systems can reach an accuracy of a few centimeters, but the signal quality is not always optimal. In addition, these systems provide good global estimation, but poor locality.

We combine the GPS priors $\mathbf{Z}_t^{\mathrm{g}} \in \mathbb{SE}(3)$ with a variation of Bundle Adjustment (BA) formulation proposed in Chap. 4. This allows us to combine the global accuracy of the GPS with the local precision of registration approaches. Our procedure works first by computing a LiDAR odometry that expresses the relative transform $\mathbf{Z}_{t,t+1}^{\mathrm{sm}} \in \mathbb{SE}(3)$ between subsequent frames. To this extent, we use the same ICP algorithm used for temporal synchronization mentioned in Sec. 6.3.3. This odometry is accurate, reliable in the short term, and can cope with GPS outages. Subsequently, we determine a global alignment of all the poses using the RTK-GPS readings and considering the incremental measurements of the LiDAR odometry. In short, we solve the following optimization problem:

$$\begin{aligned}\mathbf{X}_{1:T}^{*} = \underset{\mathbf{X}_{1:T}}{\operatorname{argmin}} \sum \|\log(\mathbf{Z}_{t,t+1}^{\mathrm{sm}-1}\mathbf{X}_t^{-1}\mathbf{X}_{t+1})\|^2_{\mathbf{\Omega}_{t,t+1}^{\mathrm{sm}}} \\ + \sum \|\log(\mathbf{Z}_t^{\mathrm{g}-1}\mathbf{X}_t)\|^2_{\mathbf{\Omega}_t^{\mathrm{g}}}\end{aligned} \tag{6.1}$$

Here $\log(\mathbf{K})$ is the operator that maps an element $\mathbf{K}$ of the Lie group to its corresponding element of the Lie algebra, as described in Eq. (2.59). Accordingly, in Eq. (6.1) $\mathbf{\Omega}_t^{\mathrm{sm}}$ is the information matrix resulting from scan matching, while $\mathbf{\Omega}_t^{\mathrm{g}}$ encodes the GPS accuracy.

Once this process is completed and we have a reasonable initial guess, we look for the set of poses that is maximally consistent with all LiDAR scans. We solve the following problem using geometric and photometric error terms. The geometric part based on point-to-plane results in a configuration of the rigid motion which is close to the optimum. As already shown in Chap. 4, the photometric step increases the accuracy of the estimates by ensuring subpixel consistency. Let $\langle i, j \rangle$ be the set of poses, $\langle k, l \rangle$ geometric associations; The total residual can be expressed as

$$E^{\mathrm{ba}} = \sum_{i,j,k,l} \rho_{\mathrm{geo}} \|e_{k,l}^{\mathrm{geo}}\|^2_{\mathbf{\Omega}_{\mathrm{geo}}} + E^{\mathrm{photo}}, \tag{6.2}$$

where E^{photo} is computed from Eq. (4.4).

We employ a point-to-plane metric for the geometric error term, explicitly relying on efficient KD-tree data association based on PCA splitting criteria. The geometric term can be compactly written as:

$$e_{k,l}^{\mathrm{geo}}(\mathbf{X}_i, \mathbf{X}_j) = (\mathbf{X}_i \mathbf{p}_{i,j} - \mathbf{X}_j \mathbf{p}_{k,l}) \cdot (\mathbf{R}_i \mathbf{n}_{i,k}) \tag{6.3}$$

with $\mathbf{p}_{i,k}$ and $\mathbf{p}_{j,l}$ denoting corresponding points between the poses $\mathbf{X}_i$ and $\mathbf{X}_j$, and $\mathbf{n}_{i,k}$ be the corresponding normal. The photometric term, instead, follows the formulation illustrated in [29], but relies only on range and intensity images. The overall error function to minimize, taking into account GPS information, will be therefore

$$\mathbf{X}_{1:T}^{gt} = E^{\mathrm{ba}} + \sum \|\log(\mathbf{Z}_t^{\mathrm{g}-1}\mathbf{X}_t)\|^2_{\mathbf{\Omega}_t^{\mathrm{g}}} \tag{6.4}$$

We measured the accuracy of our ground truth using a Total Station and 6 highly reflective markers disposed as a hexahedron in the scene to lock all the 6 DOF. Since ranges are invariant of reference frame, we measure the differences between the distances measured from the points acquired from Total Station and the one detected in our estimated map. Our 6-dof ground truth results in ± 3 cm accuracy on a trajectory of length of approximately 1.5

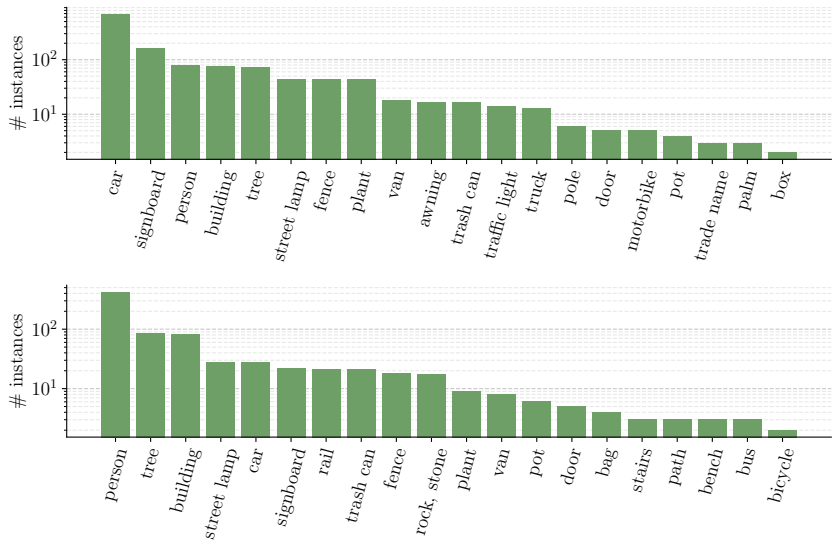


Figure 6.5. Number of top 20 most frequent semantic instance for Ciampino (above) and Colosseum (below) sequences. The instances were counted using OneFormer [55] over a subset of images for each sequence and excluding the most predominant classes: sky, wall, road, grass, sidewalk, ground.

km (indoor/outdoor). Our ground truth generation process has been shown to scale well to large environments while relying only on the onboard sensors. The final estimated global clouds are usually in the order of billions of points.

We release the ground truth for each training sequence, always expressed in the LiDAR reference frame $\mathrm{RF_L}$.

6.3.5 Data Selection

In the context of this research study, the OS0-128 LiDAR system offers extensive capabilities for collecting spatial data. Specifically, it delivers precise range measurements across the entire horizontal plane, covering large distances. Furthermore, it employs an dense array of vertical beams distributed over a $90°$, enabling comprehensive scans of a spherical area surrounding the sensor.

Given the nature of the chosen environments and to ensure a rich and diverse dataset, we used various configurations provided by the LiDARs with varying resolution and frequency. We used the OS1-64 for car sequences at 20 Hz to maximize the observation in wide scenarios and to reduce the skewing effect at higher speeds. Moreover, we employed the OS0-128 for hand-held sequences at 10 Hz to maximize the observation in narrow scenarios.

We provide 6 datasets split into different sequences. Among the datasets, 4 were acquired by walking using the hand-held device, while the other 2 collected by car. Each sequence was collected in a different environment, with different challenging scenarios such as dynamics, traffic, long sequences, and wide areas (Fig. 6.5). Tab. 6.3 summarizes some parameters for the sequences and provides some illustration.

In the remainder, we shortly review each sequence, describing the scenario:

Spagna: this sequence has been acquired in *Piazza di Spagna* and in the nearby streets. It

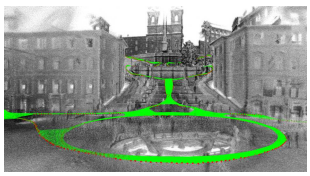
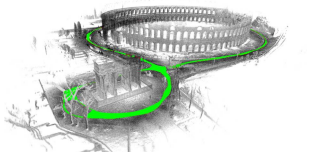
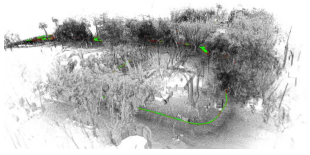
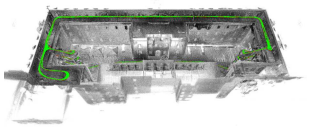
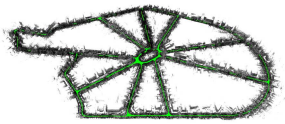
Dataset	Detail
Name: *Spagna* Motion: *hand-held* Type: *urban/vertical* Length: *2.045 km* Duration: *1827 s*	
Name: *Colosseum* Motion: *hand-held* Type: *urban/dynamics* Length: *2.159 km* Duration: *1383 s*	
Name: *Pincio* Motion: *hand-held* Type: *park/trees* Length: *2.541 km* Duration: *2064 s*	
Name: *DIAG* Motion: *hand-held* Type: *outdoor/indoor* Length: *1.480 km* Duration: *1458 s*	
Name: *Campus* Motion: *car* Type: *urban, underpasses* Length: *11.455 km* Duration: *2290 s*	
Name: *Ciampino* Motion: *car* Type: *urban/traffic* Length: *21.064 km* Duration: *3688 s*	

Table 6.3. Summary of our sequences.

Figure 6.6. Overlay of the 3D model obtained from our ground truth system and a view from Google Maps.

features several large loops going up and down the stairs hence, the trajectory is non-planar. The narrow streets limit the FoV of the LiDAR, but the building facades are a rich source of structure.

Colosseum: consists of two rounds around the *Colosseum* and the *Arco di Costantino*. The range of the LiDAR is not always sufficient to capture vertical structure. In some cases, the maximum range of the sensor is not sufficient to measure the entire surroundings, and the environment is repetitive, making some chunks difficult for state estimation.

Pincio: several loops were collected in *Villa Borghese*. This dataset is characterized by rich vegetation and a repetitive environment.

DIAG: this sequence is a mixed indoor/outdoor dataset. We traveled inside and outside our building, walking inside the corridors, through the courtyard, up to the stairs, and on the roof, which is the only part where the reception of RTK-GPS was available.

Campus: in contrast to all other sequences, this has been acquired at the main Campus of Sapienza University using the equipped car, and features several loops, spanning approximately all the streets that can be traversed by car. There are several narrow passage and some tunnels that pass under buildings. The dynamic is composed mainly of people walking composing a low percentage of the recorded data.

Ciampino: these sequences have been recorded in the city of Ciampino (Fig. 6.6). It is the longest sequence so far: the length of the total trajectory is about 21 km, while being subject to moderate dynamics.

6.4 Benchmark

For a detailed assessment of SLAM and odometry estimation, we concentrate on the Absolute Trajectory Error (ATE) and Relative Pose Error (RPE). As in [40], we assess rotation and translation errors independently rather than merging them into one metric.

ATE emphasizes SLAM performance over odometry. To compute its RMSE, we first align the estimated trajectory with the ground truth using a $\mathbb{SE}(3)$ transformation, matching poses with synchronized timestamps and employing the Horn method [50]. Subsequently, we calculate the RMSE of the ATE [m] among all the matched poses.

On the other hand, RPE emphasizes the odometry comparing local motion estimate chunks within the ground truth. It involves computing the RPE [%] (measured in percentage),

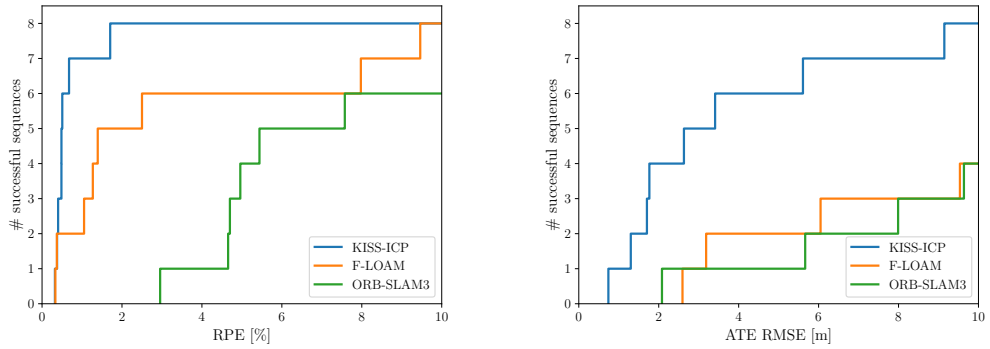


Figure 6.7. Our benchmark. Cumulative RPE [%] and ATE RMSE [m] across all training sequences in our dataset for KISS-ICP, F-LOAM and ORB-SLAM3.

over a set of subsequences of different lengths, as proposed by [40]. Afterwards, the translational RPE [%] and the rotational RPE [deg/m] of a sequence are computed as the average of all chunks RPE. Differently than any other benchmark, we have chosen to make chunk lengths adaptive to the total sequence length. In fact, local and global accuracy of our ground truth trajectories allows us to choose subsequences of arbitrary lengths, without biasing evaluation results.

Given a trajectory estimate for each sequence, a cumulative error curve is computed, like those in Fig. 6.7. For a given error value on the x-axis, the y-axis shows in how many sequences a method achieves a lower error. Therefore, the method ranking is determined as the area under curve, up to the selected maximum error and, for this metric, the larger the better. This metric rewards the robustness of evaluated methods, since a successful result on a sequence usually adds much more area under the curve than slightly improving the accuracy on many sequences.

6.5 Evaluation

To assess our recorded data's integrity, we evaluated different LiDAR odometry and visual SLAM systems on all our training sequences, specifically focusing on three notable solutions: KISS-ICP [133], F-LOAM [135] and ORB-SLAM3 [15]. The evaluation outcomes are reported in Fig. 6.7, showing cumulative RPE [%] and ATE RMSE [m], with threshold limits set to 10 % and 10 m respectively. As expected, LiDAR odometry methods are more robust and accurate than visual SLAM, in fact both KISS-ICP and F-LOAM perform successfully across all our 8 training sequences. The outcomes slightly change in the ATE RMSE [m] graph, where the area under the curve of every method is reduced, emphasizing the challenges in estimating the egomotion with global accuracy.

6.6 Conclusions

This chapter presented a vision and perception dataset, specifically targeted at SLAM and odometry estimation methods. Our sequences cover different environments and are acquired in a hand-held fashion and by using a car. Our design is to accommodate various types of

robotic platforms, including quadrupeds, quadrotors, and autonomous vehicles, making it a versatile resource for the robotics and vision community. Compared to existing datasets, we offer a variety of environments within our sequences. Moreover, this work presents a novel ground truth estimation, fusing an RTK-GPS with a LiDAR Bundle Adjustment schema. All the sequences are split into training and test sets. In addition we provide a public benchmark evaluation system, accessible from our website, that produces a leaderboard from the results submitted by the community.

As a further service to the community, we plan to extend our benchmark with other sequences, annotations and challenges in the area of computer vision and robotic perception (i.e. semantics, monocular and stereo dense depth estimation, object tracking, etc.).

Chapter 7

Splat-LOAM: Gaussian Splatting LiDAR Odometry and Mapping

7.1 Introduction

As described in Sec. 1.1, finding a common representation on which to integrate both LiDAR and RGB camera measurements is critical for achieving a good reconstruction. To this extent, Gaussian-based representations (described in Sec. 2.5.2) seem to be quite effective. This work aims at closing the gap between tightly-coupled LiDAR-camera unified scene optimization. To achieve this, we need a way to interact with the Gaussian Splatting framework directly from LiDAR measurements. Moreover, the chapter will focus solely on LiDAR.

Since the measurements already capture the 3d structure, many LiDAR Simultaneous Localization and Mapping (SLAM) pipelines do not explicitly refine the underlying point representation [8, 110, 10, 34, 25, 28]. Moreover, a global map can be obtained by directly stacking the measurements together. However, this typically leads to extremely large point clouds that cannot be explicitly used for online applications. Several approaches attempted to use surfels [6, 98] and meshes [100]. Although these approaches manage to simultaneously estimate the sensor's ego-motion while optimizing the map representation, they result in a trade-off between accuracy, memory usage, and runtime.

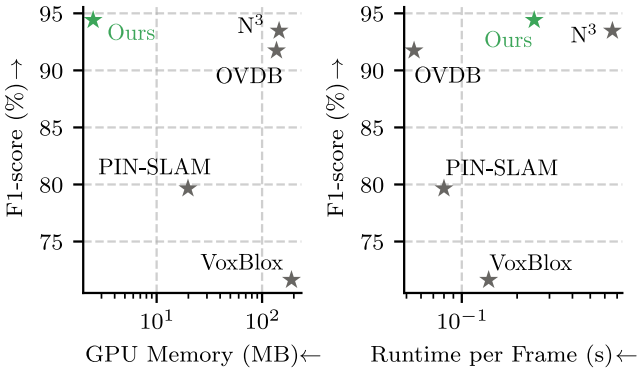


Figure 7.1. Performance overview of Splat-LOAM. F1 score to Active Memory [Mb] and Runtime [s]. The plots provide a quantitative comparison between state-of-the-art mapping pipelines, while PIN-SLAM and ours also perform online odometry.

As described in Sec. 2.5, the recent advent of Neural Radiance Field (NeRF) [80] sparked fresh interest in Novel View Synthesis (NVS) tasks. Specifically, given a set of input views and triangulated points (i.e. obtainable via COLMAP [106, 107]), NeRF learns a continuous volumetric scene function. Color and density information is propagated throughout the radiance field by ray-casting pixels from the view cameras and, through a Multi-Layer Perceptron (MLP), the method learns a representation that ensures multi-view consistency. Concurrently, NeRFs inspired the computer vision community to tackle the problem of dense visual SLAM through *implicit* methods. The pioneering work in this context was iMAP [120], which stores the global appearance and geometry of the scene through a single MLP. Despite its success in driving a new research direction, the method suffered from issues related to the limited capacity of the model, leading to low reconstruction quality and catastrophic forgetting during the exploration of larger areas. These issues were later handled by shifting the paradigm and by moving some of the appearance and geometric information over a hierarchical feature grid [164] or neural point clouds [103, 72]. Similarly, these techniques were employed on LiDAR measurements to provide more accurate and lighter explicit dense representations [160, 117, 27, 54, 93]. However, these methods still require tailored sampling techniques to estimate the underlying Signed Distance Function (SDF) accurately. This bottleneck still poses problems for online execution.

To this end, as described in Sec. 2.5.2, 3D Gausian Splatting (3DGS) [61] leverages 3D Gaussian-shaped primitives and a differentiable, tile-based rasterizer to generate an appearance-accurate representation. Furthermore, having no need to model empty areas and no neural components, 3DGS earned a remarkable result in accuracy and training speeds. Additionally, several approaches further enhanced the reconstruction capabilities of this representation [46, 23, 51]. Being fast and accurate, this representation is now sparking interest in dense visual SLAM. Recently, 3D Gaussians were employed in several works, yielding superior results over implicit solution [78, 151, 162].

One issue with Gaussian Splatting relates to the primitive initialization. In areas where few or no points are provided by SFM, adaptive densification tends to fail, often yielding under-reconstructed regions. LiDAR sensor is quite handy at solving this problem as it provides explicit spatial measurements that can be used to initialize the local representation [49, 141].

However, to our knowledge, no attempt has been made to evaluate the performance of this representation for pure LiDAR data. This technique could be interesting for visual NVS initialization and LiDAR mapping as produce a lightweight, continuous, and view-consistent representation. These insights led us to the development of Splat-LOAM, the first LiDAR Odometry and Mapping pipeline that only leverages Gaussian primitives as its surface representation. Our system demonstrates results on par with other State Of The Art (SOTA) pipelines at a fraction of the computational demands, proving as an additional research direction for real-time perception in autonomous systems.

7.2 Related Works

Classic LiDAR Odometry and Mapping: *Feature*-based methods that leverage specific points or groups of points to perform incremental registration. For instance, [154, 111] track feature points on sharp edges and planar surface patches, enabling high-frequency odometry estimation. On the other hand, *direct* methods leverage the whole cloud to perform

registration. Specifically, these methods can be categorized based on the subjects of the alignment. *Scan-to-Scan* methods matches subsequent clouds, either explicitly [8, 110, 10] or through neural methods [68, 134], while *Scan-to-Model* methods match clouds with either a local or a global map. Typically, the map is represented using points [34, 25], surfels [6, 98], meshes [100]. Another explored solution project the measurements onto a spherical projection plane to leverage visual techniques for ego-motion estimation [88, 159, 28, 29].

Implicit Methods: Concerning mapping only methods, Zhong *et al.* [160] proposed the first method for LiDAR implicit mapping that, given a set of point clouds and the corresponding sensor poses, used hierarchical feature grids to estimate the SDF of the scene. Their results demonstrated once again the advantages of such representation for surface reconstruction in terms of accuracy and memory footprint. A similar approach from Song *et al.* [117] improve the mapping accuracy by introducing SDF normal guided sampling and a hierarchical, voxel-guided sampling strategy for local optimization. Building on these advancements, several LiDAR odometry and mapping techniques were proposed. Deng *et al.* [27] presented the first implicit LiDAR LOAM system using an octree-based feature representation to encode the scene's SDF, used both for tracking and mapping. Similarly, Isaacson *et al.* [54] propose a hierarchical feature grid to store the SDF information while using point-to-plane ICP to register new clouds. Pan *et al.* [93] leverage a neural point cloud representation to ensure a globally consistent estimate. The new clouds are registered using a correspondence-free point-to-implicit model approach. These methods prove that implicit representation can offer SOTA results in accuracy at the cost of potentially high execution times or memory requirements. Although targeting a different problem setting, related are also a series of visual neural SLAM methods with RGB or RGB-D input [164, 103, 72, 152, 60, 78, 151, 104], see [125] for a survey.

Gaussian Splatting. Few works tackle the use of LiDAR within the context of Gaussian Splatting. Wu *et al.* [141] propose a multi-modal fusion system for SLAM. Specifically, by knowing the LiDAR to camera rigid pose, the initial pose estimate is obtained through point cloud registration and further refined via photometric error minimization. In this framework, the LiDAR points are leveraged to initialize the new 3D Gaussians. In a related approach, Hong *et al.* [49] propose a LiDAR-Inertial-Visual SLAM system. The initial estimate is computed through a LiDAR-Inertial odometry. Points are partitioned using size-adaptive voxels to initialize 3D Gaussians using per-voxel covariances. Both the primitives and the trajectory are further refined via photometric error minimization. While these methods introduce LiDAR measurement, 3D Gaussians are still inherently processed by cameras. Recently, Chen *et al.* [16] apply 3D Gaussians for the task of LiDAR NVS for re-simulation. The authors propose the use of Periodic Vibrating 3D Gaussian primitives to account for dynamic objects present in the scene. The primitives are initialized using a lightweight MLP, and the rasterization is carried out in a spherical frame by computing a per-primitive plane orthogonal to the ray that connects the primitive's mean to the LiDAR frame, thus removing any distortion in the projection process. Focusing on a different formulation, Jiang *et al.* [56] propose a method of LiDAR NVS that leverages Periodic Vibrating 2D Gaussian primitives. The primitives are initialized by randomly sampling points and, further optimized using the losses described in [51], along with a Chamfer loss is introduced to constrain the 3D structures of the synthesized point clouds, and an additional ray-drop term to account for phenomena like non-returning laser pulses. This term is further refined through a U-Net that considers other factors, such as the distance of the surface from the sensor. Compared

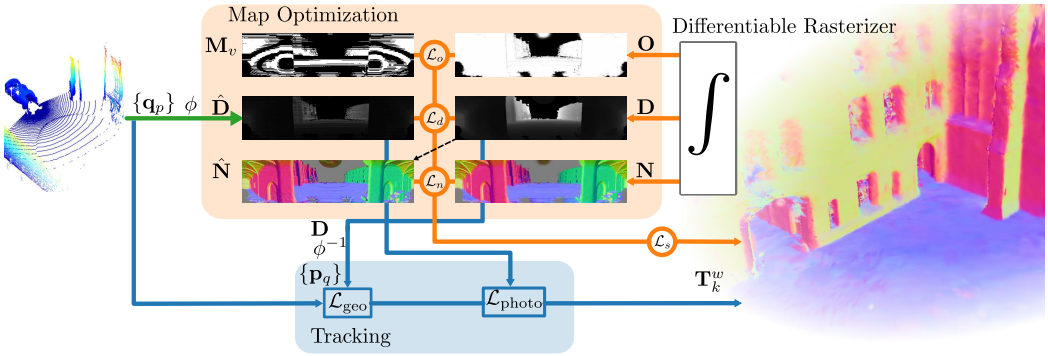


Figure 7.2. Splat-LOAM Overview. Given a LiDAR point cloud, we leverage the spherical projection to generate an image-like representation. Moreover, using an ad-hoc differentiable rasterizer, we guide the optimization for structural parameters of 2D Gaussians. The underlying representation is concurrently used to incrementally register new measurements.

to this work, we provide a thorough methodology to render 2D Gaussians on spherical frames while accounting for coordinates singularity and a cloud registration technique to allow for simultaneous odometry and mapping. To our knowledge, Splat-LOAM is the first pure LiDAR Odometry and Mapping pipeline that leverages Gaussian primitives both for mapping and tracking. In sum, our contributions are:

- A differentiable, tile-based rasterizer for 2D Gaussians for spherical frames.
- A mapping pipeline that allows the merging of sequential LiDAR measurements into a 2D Gaussian representation.
- A tracking schema that leverages both 3D and 2D representations to register new measurements and estimate the sensor ego-motion.

7.3 Method

This section introduces our novel LiDAR odometry and mapping method based on 2D Gaussian primitives. We detail a mapping strategy for initializing, refining, and integrating these primitives alongside a registration method that leverages geometric and photometric cues from the continuous local model for ego-motion estimation.

Ray-splat Intersection

We render the 2D Gaussians via α-blending as in [61] and as described in Sec. 2.5.2. Let $\mathbf{v} = \boldsymbol{\pi}_l^{-1}(\mathbf{u})$ be the normalized direction of pixel $\mathbf{u}$. We parametrize $\mathbf{v}$ as the intersection of two orthogonal planes:

$$\mathbf{h}_x = \frac{\mathbf{v} \times \mathbf{u}_z}{\|\mathbf{v} \times \mathbf{u}_z\|} \ , \qquad \mathbf{h}_y = \mathbf{h}_x \times \mathbf{v} \ , \tag{7.1}$$

where $\mathbf{u}_z$ is the unit z-direction vector. Then, we represent the planes in the splat's space as:

$$\mathbf{h}_\alpha = ({}^c\mathbf{T}_w\mathbf{H})^T \, \mathbf{h}_x \ , \qquad \mathbf{h}_\beta = ({}^c\mathbf{T}_w\mathbf{H})^T \, \mathbf{h}_y \ , \tag{7.2}$$

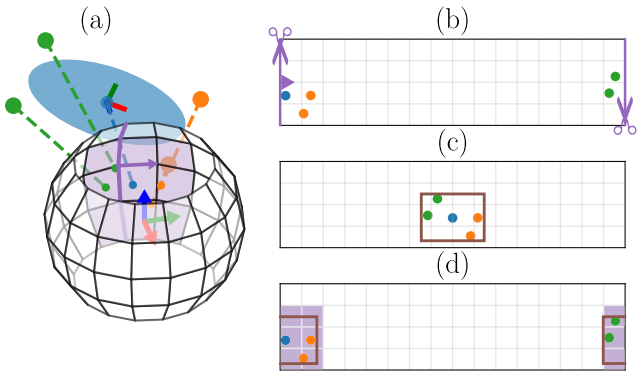


Figure 7.3. Bounding box computation for near-singularity splats. **(a)** shows the 3D configuration of a splat that approximately lies behind the camera. **(b)** shows the corresponding spherical image with the projected bounding box vertices. The distortion is removed by shifting the vertices along the horizontal direction to align the projected center to the image center. **(c)** being far from the coordinate singularity, we compute the maximum extent of the splat. **(d)** we revert the shift and propagate to the corresponding tiles via an offset from the central vertex, matching with the tiles highlighted on (a).

where ${}^{k}\mathbf{T}_{w} \in \mathbb{SE}(3)$ describes the *world* in the k-th sensor reference frame. We express the intersection of the two planes bundle as:

$$\mathbf{h}_{\alpha} \left(\alpha, \beta, 1, 1\right)^{T} = \mathbf{h}_{\beta} \left(\alpha, \beta, 1, 1\right)^{T} = 0 \ . \tag{7.3}$$

The solution α and β is then computed by solving the homogeneous linear system:

$$\alpha(\mathbf{u}_s) = \frac{\mathbf{h}_{\alpha}^{2}\mathbf{h}_{\beta}^{4} - \mathbf{h}_{\alpha}^{4}\mathbf{h}_{\beta}^{2}}{\mathbf{h}_{\alpha}^{1}\mathbf{h}_{\beta}^{2} - \mathbf{h}_{\alpha}^{2}\mathbf{h}_{\beta}^{1}} \ , \qquad \beta(\mathbf{u}_s) = \frac{\mathbf{h}_{\alpha}^{4}\mathbf{h}_{\beta}^{1} - \mathbf{h}_{\alpha}^{1}\mathbf{h}_{\beta}^{4}}{\mathbf{h}_{\alpha}^{1}\mathbf{h}_{\beta}^{2} - \mathbf{h}_{\alpha}^{2}\mathbf{h}_{\beta}^{1}} \ . \tag{7.4}$$

Further details on rasterization and gradients computation are provided in Sec. A.2

Bounding Box Computation

To leverage the efficient tile-based rasterizer, we need to know the tiles that should rasterize each primitive. Typically, this is achieved by estimating a bounding box around the projected central point of the splat. On spherical images, however, we need to account for the coordinate singularity at the horizontal boundaries $\{0, W\}$. Figure 7.3 describes the approach that we designed to compute the splat image-space bounding box for spherical projection models. First, we compute the 3σ splat-space bounding box and use Eq. (2.34) to obtain its image-space vertices. Moreover, we shift the vertices to match the central vertex to the image center to ensure that no vertex is projected near the coordinate singularity at the horizontal image boundary. Eventually, we compute the horizontal extent of the splat, revert the previous rotation, and propagate the splat's ID to the nearby tiles while accounting for the coordinate singularity. This allows us to obtain consistent bounding boxes even when the splat lies approximately behind the sensor's frame.

7.3.1 Odometry And Mapping

Following the modern literature of RGB-D SLAM [162, 104, 72, 151], we rely on keyframing to optimize local maps. We choose this approach for the following advantages: first,

continuous integration over the same model can have adverse effects on artifacts and, more importantly, on runtime. Instead of decreasing the local density, we reset to a new local model if certain conditions are met, while also restricting the number of frames joining the optimization stage to allow effective online processing. Based on this, we define each local map as an individual Gaussian model $\mathbf{P}^s$:

$$\mathbf{P}^s = \{\mathcal{G}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, o) | i = 1, \dots, N\}\,. \tag{7.5}$$

Local Model Initialization

We initialize a new local model using the input LiDAR point cloud whenever necessary, such as at system startup or when visibility conditions require it. As a first step, let $\hat{\mathbf{D}}$ be the depth image obtained by projecting the LiDAR point cloud (Eq. (2.34)), then we generate a valid pixel mask via the indicator function $\mathbb{1}[\cdot]$ as

$$\mathbf{M}_v = \mathbb{1}[\hat{\mathbf{D}} > 0], \tag{7.6}$$

and compute the range gradients $\boldsymbol{\nabla}_{\mathbf{D}}$ to construct a weight map over the range image. We use a weighted sampling of n_d points to prioritize complex regions. Splat positions are computed by back-projecting the range image, while their orientations are initialized by directing surface normals toward the sensor center to enhance initial visibility.

Local Model Refinement

We perform a limited number of refinement iterations n_o on the most recent keyframes. Unlike [162, 151], we employ a geometric distribution-based sampling scheme that guarantees at least 40% probability of selecting the most recent keyframe and progressively decreases the likelihood of choosing the older ones. To filter out artifacts caused by ray-drop phenomena in the LiDAR measurements, we only consider valid pixels to refine the local model parameters $\mathbf{x}$. We start by applying a densification strategy similar to the one described in Sec. 7.3.1, with two extra terms. The first one, $\mathbf{M}_n = \mathbb{1}[\mathbf{D}_i \leq \lambda_{d,o}]$, target newly discovered areas while the second, $\mathbf{M}_e = \mathbb{1}[|\mathbf{D}_i - \hat{\mathbf{D}}_i| \geq \lambda_{d,e}]$, target under-reconstructed regions.

To optimize the geometric consistency of the local model, we employ a loss term that minimizes the L_1 error:

$$\mathcal{L}_d = \sum_{\mathbf{u} \in \mathbf{M}_v} \rho_d \|\mathbf{D}(\mathbf{u}, \mathbf{x}) - \hat{\mathbf{D}}(\mathbf{u})\|, \tag{7.7}$$

where ρ_d is a weight function dependent on the measurement's range. In addition, we employ a self-regularization term to align the splat's normals to the surface normals $\mathbf{N}$ estimated by the gradients of the range map $\mathbf{D}$ [51]:

$$\mathcal{L}_n = \sum_{\mathbf{u} \in \mathbf{M}_v} 1 - \mathbf{n}^T \mathbf{N}(\mathbf{D}(\mathbf{u}, \mathbf{x})) \tag{7.8}$$

Furthermore, to promote the expansion of splats over uniform surfaces, we introduce an additional term that operates on the opacity channel of the rasterized images. Specifically, we drive the splats to cover the areas of the image containing valid measurements by correlating the opacity image $\mathbf{D}$ with the valid mask $\mathbf{M}_v$.

$$\mathcal{L}_o = \sum_{\mathbf{u} \in \mathbf{M}_v} -\log\left(\mathbf{D}(\mathbf{u}, \mathbf{x})\right). \tag{7.9}$$

While $\mathcal{L}_o$ allows the splats to grow, it can also cause some splats to become extremely large in unobserved areas.

We employ an additional regularization term $\mathcal{L}_s$ on the scaling parameter of all primitives N to compensate for this effect. We propose a novel regularization that allows the splat to extend up to a certain value τ_s before penalizing its growth:

$$\mathcal{L}_s = \max(0, \tau_s - \max(s_\alpha, s_\beta)) \tag{7.10}$$

We found this term to be more effective rather than directly minimizing the deviation from the average [162, 151, 78] as it allows for anisotropic splats that are particularly useful for mapping details and edges, and provides more control on the density and structure of the model.

The final mapping objective function $\mathcal{L}_{\text{map}}$ is defined as:

$$\mathcal{L}_{\text{map}} = \mathcal{L}_d + \lambda_o \mathcal{L}_o + \lambda_n \mathcal{L}_n + \lambda_s \sum_{i=1}^{N} \mathcal{L}_{s_i} \ , \tag{7.11}$$

with $\lambda_o, \lambda_n, \lambda_s \in \mathbb{R}$ being loss weights. We do not perform an opacity reset step, which could lead to catastrophic forgetting.

Frame-To-Model Registration

Each time a new keyframe is selected, we sample the local model by back-projecting the rasterized range image $\mathbf{D}$ onto a point cloud $\{\mathbf{p}_q\}_{q=1}^{W \times H}$ at its estimated pose. We design an ad-hoc tracking schema to benefit from both geometric and photometric consistencies provided by the LiDAR and the rendered local model. Hence, the total odometry loss is composed of the sum of both residuals:

$$\mathcal{L}_{\text{odom}} = \mathcal{L}_{\text{geo}} + \mathcal{L}_{\text{photo}}. \tag{7.12}$$

Geometric Registration. To associate geometric entities, we employ a PCA-based kd-tree [34]. The kd-tree is built on the back-projected rendered range map $\{\mathbf{p}_q\}_{q=1}^{W \times H}$ and segmented into tree leaves, corresponding to planar patches. Each leaf corresponds to $l = \langle \mathbf{p}_l, \mathbf{n}_l \rangle$, that is, the mean point $\mathbf{p}_l \in \mathbb{R}^3$ and the surface normal $\mathbf{n}_l \in \mathbb{R}^3$. The geometric loss $\mathcal{L}_{\text{geo}}$ represents the sum of the point-to-plane distance between the mean leaf $\mathbf{p}_l$ and the point of the current measurement point cloud k with $\{\mathbf{q}_p\}_{p=1}^{Q}$, along the normal $\mathbf{n}_l$ expressed in the local reference frame ${}^k\mathbf{T}_w$. Specifically, we have:

$$\mathcal{L}_{\text{geo}} = \sum_{p,q \in \{a\}} \rho \left(({}^k\mathbf{T}_w \mathbf{n}_{l_q})^T ({}^k\mathbf{T}_w \mathbf{q}_p - \mathbf{p}_{q_i}) \right), \tag{7.13}$$

where w is the global frame, k is the local frame, $\{a\}$ is the set of leaf associations with the point from the measurement point cloud, and ρ is the Huber robust M-estimator.

Photometric Registration: Leveraging the rendered range map $\mathbf{D}$, we employ photometric registration for subpixel refinement, minimizing the photometric distance between the rendered and the spherical projected query point cloud $\hat{\mathbf{D}}$. The photometric loss is formulated as:

$$\mathcal{L}_{\text{photo}} = \sum_{\mathbf{u}} \left\| \rho \Big(\mathbf{D}(\mathbf{u}) - \hat{\mathbf{D}} \underbrace{\left(\phi({}^k\mathbf{T}_w \phi^{-1}(\mathbf{u}, \hat{d})) \right)}_{\mathbf{u}'} \Big) \right\|^2. \tag{7.14}$$

Dataset	Newer College [156]				Oxford Spires [123]						Mai-City [131]			
	quad-easy		math-easy		keble-02		bodleian-02		observatory-01		mai-01		mai-02	
Approach	C-l1↓	F-score↑	C-l1↓	F-score↑	C-l1↓	F-score↑	C-l1↓	F-score↑	C-l1↓	F-score↑	C-l1↓	F-score↑	C-l1↓	F-score↑
OpenVDB [85]	7.92	88.85	11.69	84.51	7.19	91.74	**7.51**	89.68	9.59	86.16	3.33	97.29	3.26	97.37
VoxBlox [90]	16.5	64.63	11.93	80.51	15.03	71.63	15.24	58.77	15.12	70.45	7.04	93.81	5.80	95.51
N^3-Mapping [117]	8.04	94.54	fail	fail	**7.01**	93.47	7.89	**90.36**	**9.35**	**87.94**	**2.64**	**99.06**	**2.76**	**99.04**
PIN-SLAM [93]	12.89	88.05	14.69	73.82	11.83	79.65	10.74	82.71	14.49	72.31	4.91	93.84	4.82	94.11
Ours	**5.37**	**96.74**	**9.15**	**90.02**	7.43	**94.41**	7.60	90.09	10.56	83.04	3.64	97.27	5.05	95.31

Table 7.1. Reconstruction quality evaluation. The pipelines were run with ground-truth poses. Voxel size is set to 20 cm and F-score is computed with a 20 cm error threshold. Splat-LOAM yields competitive mapping performance on both the Newer College [156] and Oxford Spires [123] datasets and outperforms most competitive approaches. Best results are highligthed as **first**, **second**, and **third**.

The evaluation point $\mathbf{u}'$ in the query image $\hat{\mathbf{D}}$ is computed by first back-projecting the pixel $\mathbf{u}$, applying the transform ${}^{k}\mathbf{T}_{w}$, and then projecting it back. As described in Sec. 2.4.3, since ${}^{k}\mathbf{T}_{w}$ is an element of the manifold $\mathbb{SE}(3)$, local updates are parametrized in the Lie algebra $\mathfrak{se}(3)$. These updates are carried out using Gauss-Newton (GN) (Sec. 2.4.2).

7.4 Experiments

In this section, we report the results of our method on different publicly available datasets. We evaluate our pipeline on both tracking and mapping. We recall that, to our knowledge, this is the first Gaussian Splatting LiDAR Odometry and Mapping pipeline, and no direct competitors are available. We compared our approach with other well-known geometric and neural implicit methods. The experiments were executed on a PC with an Intel Core i9-14900K @ 3.20Ghz, 64GB of RAM, and an NVIDIA RTX 4090 GPU with 24 GB of VRAM.

For the odometry experiments, we evaluate over several baselines. The first one is a basic point-to-plane ICP odometry, as a minimal example. Then, we include two geometric pipelines: SLAMesh simultaneously estimates a mesh representation of the scene via Gaussian Processes and perform registration onto it [100]. Moreover, MAD-ICP leverages a forest of PCA-based KD-Trees to perform accurate registration [34]. Furthermore, we include PIN-SLAM [93] as SOTA baseline for implicit LiDAR SLAM. It leverages neural points as primitive and interleaves an incremental learning of the model's SDF and a correspondence-free, point-to-implicit registration schema. We also highlight that we could not run the official implementation of NeRF-LOAM [27], and thus, we do not include it in the evaluation.

We evaluate the mapping capabilities of our method against popular mapping SOTA pipelines. We include a custom Truncated Signed Distance Function (TSDF) integration pipeline based on OpenVDB [85][1] and VoxBlox [90] as "classic" baselines. OpenVDB provides a robust volumetric data structure to handle 3D points, while VoxBlox combines adaptive weights and grouped ray-casting for an efficient and accurate TSDF integration. Additionally, we include two neural-implicit mapping pipelines. N^3-Mapping is a neural-

[1]This is a custom reimplementation of TSDF integration based on the OpenVDB data structure, inspired by VDBFusion [132].

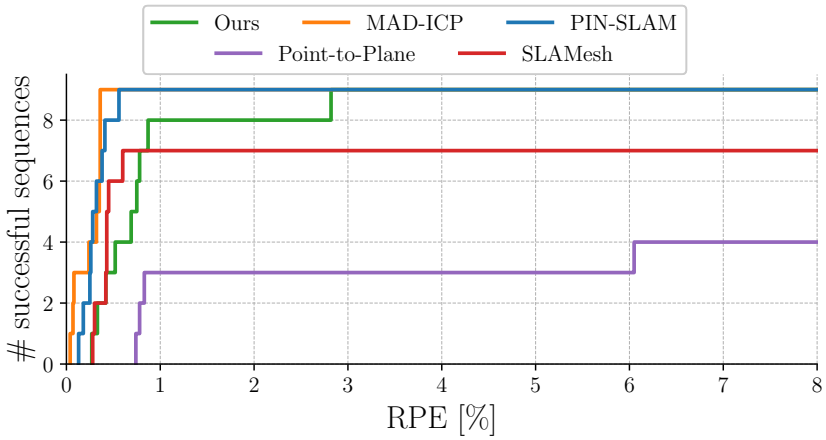


Figure 7.4. RPE evaluation. Number of successful sequences across RPE thresholds. It includes the sequences of Newer College [156], VBR [12], Oxford Spires [123] and Mai City [131].

Method	quad	math	keble	bodl	observ	pincio	spagna	campus	mai-1	mai-2	avg [%]
point2plane	0.0605	0.0103	0.1715	0.2110	0.1747	0.0074	0.0078	0.0083	fail	fail	8.14
SLAMesh	0.0043	0.0039	0.0060	0.0045	0.0043	fail	fail	fail	0.003	0.0028	0.41
MAD-ICP	**0.0036**	0.0037	**0.0036**	0.0036	0.0035	0.0024	**0.0008**	**0.0007**	0.0032	0.004	**0.29**
PIN-SLAM	0.0041	**0.0019**	0.0038	**0.0025**	**0.0028**	**0.0018**	0.0013	0.0056	**0.0026**	**0.0032**	0.30
Ours	0.0057	0.0038	0.0078	0.0087	0.0052	0.0027	0.0033	0.0075	0.0069	0.0071	0.59

Table 7.2. Tracking RPE evaluation. Averages in red are computed without failed sequences. Best results are highligthed as **first**, **second**, and **third**.

implicit non-projective SDF mapping pipeline [117]. It leverages normal guidance to produce more accurate SDFs, leading to SOTA results for offline LiDAR mapping. PIN-SLAM [93] is included also for the mapping experiments. In fact, using marching cubes, it can produce a mesh from the underlying implicit SDF. Below, we report the datasets and the evaluation metrics employed. We highlight that we could not run the official implementation of SLAMesh [100] with ground-truth poses. Hence, we do not include the pipeline for quantitative comparisons.

7.4.1 Datasets

We used the following four publicly available datasets:

- **Newer College Dataset (NCD) [156]:** Collected with a handheld Ouster OS0-128 LiDAR in structured and vegetated areas. Ground truth was generated using the Leica BLK360 scanner, achieving centimeter-level accuracy over poses and points in the map.
- **A Vision Benchmark in Rome (VBR) [12]:** Described in Chap. 6.
- **Oxford Spires [123]:** Recorded with a hand-held *Hesai QT64* LiDAR featuring a 360° horizontal FoV, 104° vertical FoV, 64 vertical channels, and 60 meters of range detection. Similar to [156], each sequence includes a prior map obtained via a

survey-grade 3D imaging laser scanner, used for ground-truth trajectory estimation and mapping evaluation. Specifically, we choose the *Keble College*, *Bodleian Library*, and *Radcliffe Observatory* sequences to include both indoor and outdoor scenarios with different levels of detail.

- **Mai City [131]:** A synthetic dataset captured using a car-like simulated LiDAR with 120 meters of range detection. The measurements are generated via ray-casting on an underlying mesh, providing error-free, motion-undistorted data. We select the *01* and *02* sequences, which capture similar scenarios with different vertical resolutions.

7.4.2 Evaluation

We use Relative Pose Error (RPE) computed with progressively increasing delta steps to evaluate the odometry accuracy. Specifically, we adapt the deltas to the trajectory length to provide a more meaningful result (see Chap. 6). Differently, to evaluate mapping quality, we use several metrics [79]: Accuracy (Acc) measures the mean distance of points on the estimated mesh with their nearest neighbors on the reference cloud. Completeness (Comp) measures the opposite distance, and Chamfer-l1 (C-l1) describes the mean of the two. Additionally, we use the F-score computed with 20 cm error threshold. Table 7.1 shows the mapping results in which, each pipeline is evaluated with known sensor poses. This allows us to evaluate the capacity to integrate new measurements incrementally and produce a geometric consistent surface reconstruction. Splat-LOAM achieves SOTA results in this scenario, highlighting how 2D Gaussians show promising results for LiDAR only surface reconstruction tasks. Additionally, using our local map strategy, we reported very low peak GPU memory usages (average of 2 MB per local map). Additional qualitative comparisons are shown in Fig. 7.6. Furthermore, Fig. 7.4 and Tab. 7.2 shows the tracking results over the evaluated datasets. Compared with other SOTA pipelines, Splat-LOAM performs slightly worse (approximately 0.3% worse on average than the best performing). We notice that the gap in RPE typically stems from failures at tracking incoming frames. This clearly indicates some weakness in the tracking procedure for our methodology. In the future, it would be interesting to study more sophisticated methodologies for registering incoming LiDAR measurements within this representation.

7.4.3 Ablation Study

Our approach employs Gaussian primitives for LiDAR odometry estimation and mapping, yielding results comparable to state-of-the-art methods while significantly enhancing computational efficiency. In this section, we evaluate some key aspects of our pipeline and evaluate their contributions.

Memory and Runtime Analysis: In Fig. 7.5, we report how the increment of active primitives affects the active GPU memory requirements and the per-iteration mapping frequency. It shows an experiment ran over the large-scale *campus* sequence [12] where we set a maximum of 100 keyframes per local map and sample, at most, 50% of points for the incoming point cloud. It's possible to notice that the mapping FPS remains stable between 200k and 300k primitives. This is most likely related to the saturation of the rasterizer.

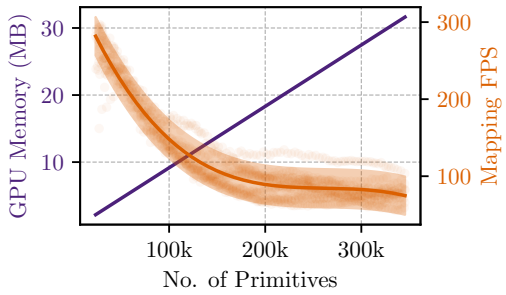


Figure 7.5. Memory and Runtime Analysis. The plot relates the used GPU Memory and the mapping iteration frequency with the number of active primitives. The measurements were obtained over the *campus* (Chap. 6), the longest sequence we evaluated.

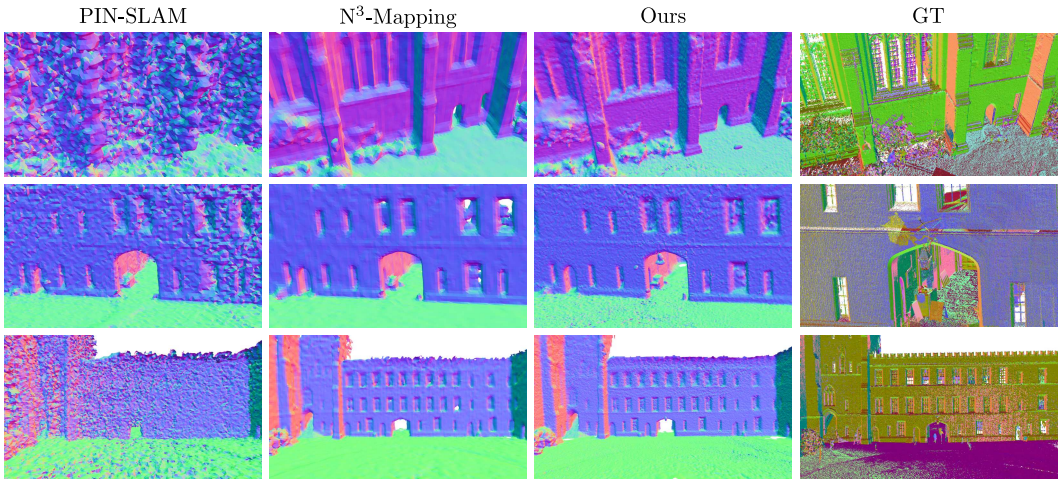


Figure 7.6. Comparison of Mesh Reconstruction. The figure shows reconstruction results for *quad-easy* sequence from the newer college dataset. Our method recovers a geometry with much higher data fidelity. PIN-SLAM lacks many details and exhibits a large level of noise. N^3-Mapping performs more similar to ours, but oversmoothes fine geometric details.

Odometry: Fig. 7.7 shows the results for different tracking methods over our scene representation. Using both geometric and photometric components, we achieve a better result than using point-to-point or point-to-plane. The last three bars show an ablation of geometric and photometric loss components. The results are best for the joint use of both terms which support our design choice.

Mesh generation: To generate a mesh from the underlying representation, we sample n_m points from each keyframe rendered range and normal maps. Moreover, similar to [46], we accumulate the points into a single point cloud and generate the mesh using Poisson Reconstruction [59]. Tab. 7.3 shows the results of two additional methods for meshing that we tested: the Poisson Reconstruction of the Gaussian primitives' centers and normals and the TSDF integration using the rendered depth and normal maps from the keyframes.

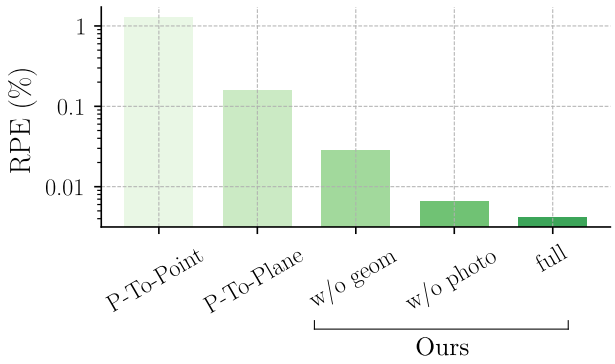


Figure 7.7. Ablation on registration methods. The plot reports the RPE (%) of several tested registration methods on the quad-easy sequence. Enabling both geometric and photometric factors in sequence, provides a more robust estimate. the quad-easy sequence [156]

Extraction Method	Acc↓	Com↓	C-l1↓	F-score↑
Marching Cubes[85]	16.76	5.53	11.14	76.76
Poisson (centers)	10.15	6.70	8.43	92.33
Ours	**6.64**	**4.09**	**5.37**	**96.74**

Table 7.3. Ablation on Meshing Methods. We report mapping results with varying meshing methods on the quad-easy sequence [156]. Our method yields consistently better results.

7.5 Conclusions

We present the first LiDAR Odometry and Mapping pipeline that leverages 2D Gaussian primitives as the sole scene representation. Through an ad-hoc tile-based Gaussian rasterizer for spherical images, we leverage LiDAR measurements to optimize the local model. Furthermore, we demonstrate the effectiveness of combining a geometric and photometric tracker to register new LiDAR point clouds over the Gaussian local model. The experiments show that our pipeline obtains tracking and mapping scores that are on par with the current SOTA at a fraction of the computational demands.

Future Work: We plan on improving Splat-LOAM by simultaneously estimating the sensor pose and velocity to compensate for motion skewing of the LiDAR measurements. Moreover, we plan on including Loop Closure (LC) to improve the pose estimates, along with the mapping accuracy.

Chapter 8

Resolution Where It Counts: Hash-based GPU-Accelerated 3D Reconstruction via Variance-Adaptive Voxel Grids

8.1 Introduction

One aspect that is missing in Chap. 7 is the dedicated optimization of surfaces. Several recent works tackle the problem of Gaussian Splatting surface reconstruction and, in accordance with Neural Radiance Field (NeRF)-like methods, the most promising path relies on first estimating the implicit Signed Distance Function (SDF) and then using it as a guidance to seed new Gaussians [76, 149, 92]. This is a reasonable workflow as SDF models surfaces by construction, while the occupancy estimated by radiance fields methods, only describes the density of a given point in space. The latter produces worse surfaces because, by default, occupancy is not constrained to live on a thin surface (as opposed to the 0-level SDF set).

To this end, this chapter presents a different implementation for modeling TSDF that can maintain low memory requirements, provides adaptive resolution and is fully integrated in CUDA[1].

Introduced by Curless and Levoy [20], TSDF-based fusion integrates depth over time into a voxel grid, enabling robust surface reconstruction. However, uniform grids scale poorly due to high memory and compute demands. To improve efficiency, adaptive structures like octrees and OpenVDB have been proposed. Octrees hierarchically allocate memory where needed but suffer from recursive traversal and poor GPU performance. OpenVDB [85] offers sparse multi-resolution storage via a tree structure, and variants like VDBFusion [132] have extended it to TSDF fusion. Despite their versatility, these structures introduce non-trivial access overhead and are not optimized for real-time, GPU-based reconstruction. Furthermore, many existing multi-resolution approaches depend on input-specific cues (e.g., semantics or depth confidence), limiting their generality and adaptability across sensor types. In this work, we propose a novel variance-adaptive, multi-resolution voxel grid implemented

[1] Allows to share results with other GPU radiance field methods without having to transfer them between RAM-GPU.

Figure 8.1. Overview. Our pipeline for real-time 3D reconstruction and rendering. Our variance-adaptive voxel grid dynamically adjusts resolution based on Truncated Signed Distance Function (TSDF) variance, enabling compact and accurate reconstructions from both RGB-D and unstructured 3D point cloud. The mesh output supports high-quality rendering via Gaussian Splatting, with adaptive splat control handled fully on the GPU.

entirely on the GPU using a flat spatial hash table. Our system adjusts resolution based on local TSDF variance—independent of sensor modality or semantic priors—enabling fine detail where needed and coarse representation elsewhere. Unlike octrees or VDBs, our approach avoids hierarchical traversal, achieving constant-time access and real-time performance. Our main contributions are:

- A variance-adaptive voxel grid that adjusts resolution based on local TSDF variance, preserving fine detail and reducing memory in uniform areas. We extend Marching Cubes to support seamless meshing across resolution boundaries.
- A flat hash table structure that manages mixed-resolution voxel blocks without hierarchical overhead, enabling efficient GPU integration and access.
- A fully parallel GPU-based quad-tree for controlling Gaussian Splatting, allowing adaptive splat density without CPU-GPU streaming.
- An open-source CUDA/C++ implementation supporting real-time reconstruction and rendering, designed for extensibility and community use.

8.2 Related Works

Volumetric 3D reconstruction using the TSDF has been a foundational approach for over three decades. The seminal work by Curless and Levoy [20] introduced volumetric fusion of depth maps, enabling robust surface reconstruction through SDF accumulation. Although initially restricted to offline processing, this approach laid the groundwork for real-time systems.

KinectFusion [87] demonstrated real-time TSDF integration on a dense, uniform voxel grid, enabling accurate reconstructions in small, bounded scenes. However, the fixed resolution and exhaustive memory allocation made the approach unsuitable for large-scale environments. Voxel Hashing [89] addressed this limitation by dynamically allocating voxel blocks using a spatial hash table, reducing memory usage and enabling real-time mapping for mobile platforms.

Building on this foundation, works like VoxBlox [90] and Supereight [129] further improved scalability. VoxBlox uses block-wise hashing and ray bundling to support onboard updates on resource-constrained UAVs, while Supereight introduces an adaptive allocation strategy to improve runtime and memory efficiency. These systems prioritize real-time performance, often at the cost of geometric precision.

OpenVDB [85] offers a more general-purpose sparse volumetric data structure widely used in graphics and VFX. It supports multi-resolution storage through a hierarchical tree structure and has been adapted to 3D reconstruction in works like VDBFusion [132], which applies it to unstructured point cloud fusion. However, OpenVDB's multi-level hierarchy introduces non-trivial lookup costs and is not well-suited for real-time integration. Additionally, each leaf node operates at a fixed resolution, limiting local adaptivity within a region.

In parallel, learning-based representations have emerged that model 3D geometry via continuous neural fields. Methods such as NICE-SLAM [163] and PIN-SLAM [93] leverage hierarchical feature grids and MLP decoding to reconstruct high-quality geometry while incorporating SLAM constraints. Although these approaches achieve accurate results, they are typically limited to small-scale scenes and require substantial compute and memory. Fully neural pipelines like NKSR [52] and N^3-Mapping [117] achieve high fidelity in sparse data regimes but rely on offline training and incur significant computational overhead.

Recent works have explored adaptive resolution guided by semantics or image gradients [158], enabling selective detail refinement. However, such approaches depend on structured RGB-D input and do not generalize to raw point clouds. Similarly, Funk et al. [38] use octree structures for adaptive occupancy mapping, but rely on log-odds surface modeling and suffer from hierarchical access overhead.

Unlike prior work, we introduce a variance-driven, multi-resolution voxel grid that is sensor-agnostic and GPU-native. Instead of relying on semantic cues or hierarchical traversal, our system adaptively allocates voxel resolution based on local TSDF variance using a flat hash table structure. This enables constant-time access, seamless resolution mixing, and real-time performance across both RGB-D and LiDAR inputs, without sacrificing geometric fidelity or memory efficiency.

8.3 Method

The TSDF map representation used consists of a collection of spatially hashed voxels V_i with a variable voxel size $\nu \in \mathbb{R}^+$, which is determined based on the density of neighboring voxels. Each voxel V_i is grouped into B_i blocks/cuboids, each of which contains a number of voxels that vary with the voxel resolution. Each block B_i is identified by a global position $\mathbf{b}_i = (x, y, z)^T \in \mathbb{Z}^3$. A voxel mainly stores a truncated signed distance D_i, a weight W_i that reflects the confidence in D_i, the color, and the variance of the mean SDF σ_i^2.

At each frame k, the system integrates the current sensor's measurement (either a raw point cloud or a dense image).

8.3.1 Integration

We integrate 3D data from the sensor into a TSDF volume, following a standard volumetric fusion strategy. The TSDF provides an implicit surface representation, where each voxel

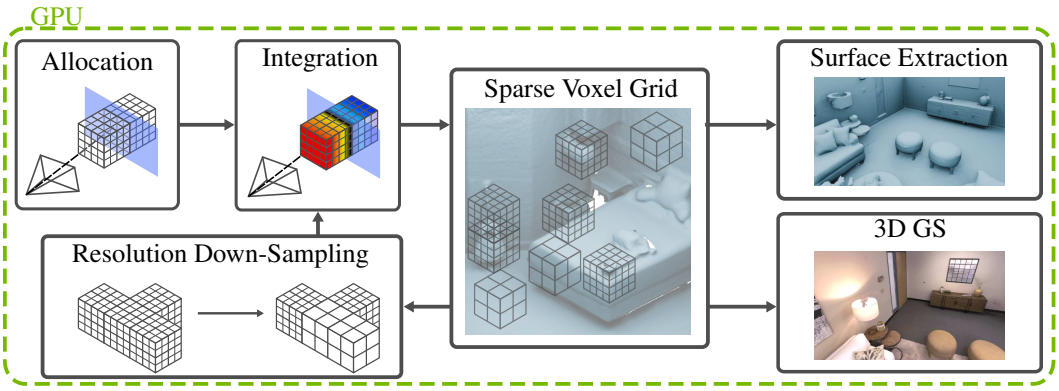


Figure 8.2. System Overview: Our system takes the input data and, if necessary, allocates the voxel blocks associated with such 3D points and integrates them into a multi-resolution sparse voxel grid. The blocks whose variance is smaller than a predefined threshold are downsampled and re-integrated at a coarser resolution into the voxel grid. Moreover, our pipeline can either extract the isosurface from the volumetric representation or render the surrounding environment.

V_i encodes a scalar value D_i indicating the signed distance from the voxel center to the closest observed surface along the sensor's line of sight. Positive values represent free space (outside the surface), while negative values correspond to space behind the surface (inside objects). To restrict updates to relevant regions, the distance is truncated beyond a fixed threshold $\tau > 0$. We differentiate integration logic based on measurement kind and whether the dense color data is available or not.

3D Point Cloud: For 3D point cloud measurements (i.e. LiDAR sensors), we perform ray-based geometric integration using a DDA traversal [3] from the sensor origin to each point $\mathbf{p}$. Voxels intersected by the ray are updated based on their signed distance to $\mathbf{p}$. For a voxel center at position $\mathbf{x}$ and a surface point $\mathbf{p}$ observed by the sensor, the truncated signed distance is defined as:

$$d_k(\mathbf{x}) = \text{clip}\left(\frac{(\mathbf{p}-\mathbf{x})^T\,\mathbf{n}}{\|\mathbf{n}\|}, -\tau, \tau\right), \tag{8.1}$$

where $\mathbf{n}$ is the direction from sensor origin to the point $\mathbf{p}$ and $\text{clip}(a, m, M)$ clips a between the lower bound m and the upper bound M.

Cameras: Differently, for cameras with depth information (i.e. RGB-D), we use a projective mapping approach, where the depth map is back-projected into 3D using camera intrinsics, and voxel updates are computed per pixel. The signed distance is computed along the viewing direction, which points from the camera center to the voxel center. Let $\pi^{-1}(u, v, d)$ be the back-projected 3D point corresponding to a depth value d at pixel coordinates (u, v). The TSDF value is computed as:

$$d_k(\mathbf{x}) = \text{clip}\left(\pi^{-1}(u, v, d) - \|\mathbf{x}\|, -\tau, \tau\right). \tag{8.2}$$

Here, $\|\mathbf{x}\|$ is the depth of the voxel center along the viewing ray, and $\pi^{-1}(u, v, d)$ gives the actual surface depth observed at that pixel.

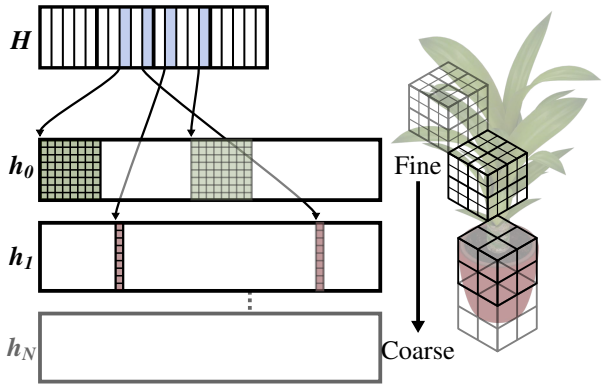


Figure 8.3. Illustration of grid representation to hash table mapping. The hash function maps the world points from integer world coordinates to the respective buckets of the hash-table H. Every entry of the hash-table contains a pointer to a block in the specific heap of that resolution

We note that the above integration strategy follows standard volumetric fusion techniques [20, 89], and is not a core contribution of our work. However, we include it here for completeness and clarity of exposition.

Independent of the integration type, each voxel maintains a weighted average of these observations:

$$D_i \leftarrow \frac{W_i D_i + w_k d_k(\mathbf{x})}{W_i + w_k}, \quad W_i \leftarrow W_i + w_k. \tag{8.3}$$

Here, w_k is a confidence weight. While it is often set based on sensor noise or distance [89, 20], we fix $w_k = 1$ to simplify the computation of voxel-wise variance. Different from other works, with the running average D_i, we maintain the sample variance σ_i^2 of the TSDF values using an online update. This variance reflects the local geometric complexity: low in flat regions, high near surface boundaries, or noise. We use σ_i^2 to guide voxel resolution—preserving fine detail where needed and merging voxels in smoother areas, enabling adaptive resolution and improved memory efficiency.

8.3.2 Grid Representation

Given the inherent sparsity of most 3D environments, a dense volumetric grid would be highly inefficient. Instead, we adopt a sparse volumetric representation based on a hash table, as introduced in [89], which enables memory-efficient storage and access to active voxel blocks. While this approach introduces additional challenges related to data locality and memory coherence, it allows for efficient large-scale scene reconstruction without requiring predefined spatial bounds. One of our main contributions is a multi-resolution grid tailored for a single hash table. Different from fixed-voxel-size methods [90, 89], we allocate voxel blocks of a constant metric size, while varying their internal resolution by adjusting the number of contained voxels per block. Each block B_i is identified by a global position $\mathbf{b}_i = (x, y, z)^T \in \mathbb{Z}^3$, which is mapped to the hash table using the following commonly used spatial hashing function:

$$H(x, y, z) = (x \cdot p_1 \oplus y \cdot p_2 \oplus z \cdot p_3) \bmod n_{hash} \tag{8.4}$$

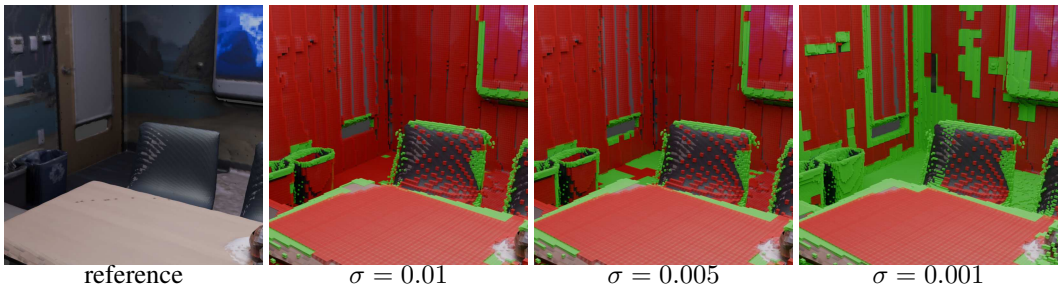


Figure 8.4. Comparison of different variance thresholds. In red, the coarser voxels, in green, the finer ones. From left to right: the reference ground truth mesh and the underlying grid at the respective thresholds. For simplicity, we show just two levels at a time. The sequence is part of Replica dataset [119].

where $p_1 = 73856093, p_2 = 19349669, p_3 = 83492791$ are large prime numbers [124] and n_{hash} is the number of entry of the hash table. As in [89], to handle collisions, each entry in the hash table maintains a fixed-size bucket for up to N voxel blocks. In the rare event of overflow, additional blocks are appended to a per-entry linked list. In addition, as is commonly done [89, 90] to efficiently allocate all intersected voxel blocks, we leverage DDA [3]. One of the major differences of our method is having a unified hash table supporting a multi-resolution voxel representation within a single address space. The hash table addresses N memory heaps, each dedicated to storing voxel blocks at a particular resolution. Regardless of their resolution, all blocks are allocated, indexed, and accessed via this shared structure. While managing multiple resolutions within a flat hash table increases implementation complexity (e.g., in hashing, memory layout, and consistency), it enables a more flexible and efficient system compared to traditional hierarchical representations [130]. Most multi-resolution volumetric approaches use octree hierarchies [38], which naturally support dyadic spatial subdivision and adaptive refinement near surfaces. However, octrees incur higher computational cost—typically $O(N\log N)$—for insertion, deletion, and neighborhood queries due to recursive traversal and pointer-based memory layouts. Even methods leveraging hybrid tree-based data structures, such as those proposed in [85, 86], provide highly efficient sparse volumetric representations, however, their hierarchical organization introduces considerable overhead for frequent updates and random access. In contrast, our method achieves constant-time access $O(1)$ for these operations by encoding multi-resolution voxel blocks directly into a flat hash table. Resolution is handled explicitly through key encoding, enabling the simultaneous representation of voxels at different scales within a unified, non-hierarchical structure. This avoids the overhead of recursive descent and facilitates more efficient parallel execution on modern hardware, particularly in GPU-based implementations where memory indirection and dynamic branching are costly. We show the benefits of our implementation in Sec. 8.4.

Voxel Merging Criterion

We initialize the voxel blocks to be of the finest resolution, since "ideally" this accounts for the best geometrical accuracy. Then, to preserve surface complexity and deal with high-frequency signals, we keep the voxel at a higher resolution in areas with high TSDF

variance. Instead, for regions exhibiting low variance, we merge voxels of the same block, as in Fig. 8.5. Our approach is fully sensor-agnostic, relying exclusively on TSDF metrics without presupposing any specific noise model. We compute variance using a numerically stable, single-pass method based on Welford's algorithm [139]. This variance calculation is performed at every integration step across the entire volumetric grid, a computationally intensive task. To address this challenge, we implemented it in a parallelized manner on the GPU, leveraging intra-block reductions to efficiently compute the sum of squared deviations.

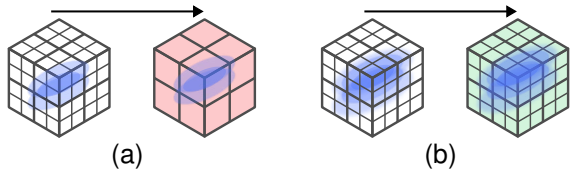


Figure 8.5. Adopted merging strategy. a) Both voxel blocks have the same initial resolution, but the right block exhibits lower TSDF variance than the right one. (b) After variance evaluation, the right block (in red) is re-allocated at a coarser resolution due to its lower variance, while the left block remains unchanged.

8.3.3 Streaming

GPU memory capacity and bandwidth are fundamental bottlenecks in large-scale 3D reconstruction, where the volumetric data grows with the scale of the environment. Retaining all voxel data in GPU memory becomes infeasible for expansive scenes, and frequent CPU-GPU transfers introduce significant overhead due to limited bandwidth and synchronization costs. To address this, it is essential to adopt a streaming strategy that manages data movement efficiently. Unlike previous systems such as [89], which stream voxel blocks based on geometric heuristics (e.g., frustum culling or proximity thresholds), our approach defers streaming until memory capacity is nearly saturated (or up to user threshold). Streaming decisions are therefore governed by memory constraints rather than spatial criteria.

8.3.4 Multi-resolution Marching Cubes

To extract the isosurface from the TSDF, we implement the Marching Cubes algorithm [74] directly on the multi-resolution voxel grid, leveraging GPU parallelism. Each voxel is processed independently using local signed distance values tri-linearly interpolated at its corners. While voxel size does not affect the core algorithm, resolution boundaries pose a challenge: interpolating corner values involves querying neighboring voxels, whose TSDF value is defined relative to different centers (see Fig. 8.7). Zheng et al. [158] address this by enforcing uniform refinement of neighboring voxels, increasing memory usage in otherwise coarse regions. Instead, we assume limited variation at the scales considered and interpolate between coarse and fine values using a weighted scheme that favors the finer resolution.

Unlike voxel-based terrain rendering in computer graphics like Transvoxel [67], a volumetric integration system like ours cannot freely allocate voxel blocks, as their creation is driven by incoming measurements and does not account for neighboring voxel resolutions. Hence, outer coarse voxels may spatially overlap with adjacent finer-resolution voxels. To resolve this, inspired by [67], we truncate coarse voxels along the dimensions corresponding to faces shared with finer neighbors, ensuring a clean partitioning of space and preventing multiple faces from being generated within the same spatial region (see Fig. 8.6). To mitigate

artifacts introduced by these approximations, we apply a vertex collapsing step (based on voxel size) during mesh generation, simplifying redundant geometry and improving consistency across resolution transitions.

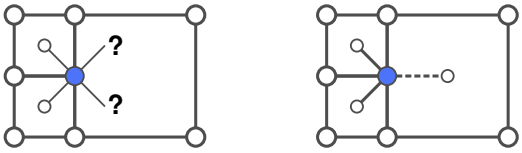


Figure 8.6. Ambiguity of SDF interpolation. The blue vertex attempts to interpolate the SDF value from its four neighboring voxels during sampling but encounters undefined entries due to missing neighbors. To address this, we apply a weighted interpolation scheme that prioritizes finer-resolution values where available. For clarity, the visualization is shown in 2D assuming bilinear interpolation; in practice, our method performs full trilinear interpolation in 3D.

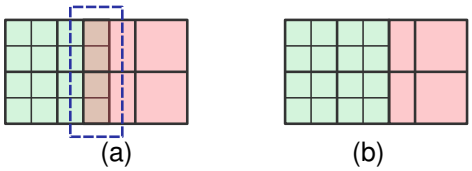


Figure 8.7. Transitional voxels. (a) Due to the measurement-driven allocation, overlapping regions (dotted area) can occur between voxel blocks of different resolutions, which poses challenges during the vertex evaluation step in Marching Cubes. (b) To address this, we reduce the size of coarser voxels and adjust the placement of Marching Cubes vertices accordingly to maintain consistency across resolutions.

8.3.5 Rendering

Our underlying voxel grid enables a more structured and efficient approach to controlling Gaussian in 3D Gaussian Splatting [61], addressing the oversaturation issues. We compute an image-space quadtree to regulate the spawning of Gaussians, using voxel-derived visibility and density cues to inform subdivision. Instead of performing recursive subdivision on the CPU [138], we propose a parallel breadth-first quadtree construction on the GPU. The quadtree construction avoids recursion by employing a parallel, stack-free implementation optimized for GPU execution. To maintain high throughput, both error estimation and quadtree construction are executed using a parallel breadth-first strategy, avoiding recursion and leveraging shared memory for reductions. Notably, all supporting data already resides in GPU memory, eliminating costly transfers and accelerating the auxiliary computations that guide adaptive splat generation.

8.4 Experiments

In this section, we report the results of our method on several publicly available datasets. We evaluate our pipeline in terms of both mapping accuracy and rendering performance. To the best of our knowledge, ours is the first approach to combine a variance-driven, multi-resolution voxel grid with a single flat hash table structure that enables fully GPU-based

TSDF integration and rendering. We compared our method against popular mapping SOTA pipelines. We include VDBFusion [132] and VoxBlox [90] as purely geometric baselines. VDBFusion employs OpenVDB [85] as volumetric data structure to handle 3D points, while VoxBlox combines adaptive weights and grouped ray-casting for an efficient TSDF integration. Additionally, we include two neural-implicit mapping pipelines. N^3-Mapping is a neural-implicit non-projective SDF mapping pipeline [117]. It leverages normal guidance to produce more accurate SDFs, leading to SOTA results for offline LiDAR mapping. PIN-SLAM [93], leverages neural points as primitives and interleaves an incremental learning of the model's SDF and, using marching cubes, it can produce a mesh from the underlying implicit SDF. To evaluate the performance of our method, we used several publicly available benchmarks that feature heterogeneous sequences and varying resolutions across different sensor types RGB-D([119],[21]) and LiDAR ([156],[123]). The experiments were executed on a PC with an Intel Core i9-14900K @ 3.20Ghz, 128GB of RAM, and an NVIDIA RTX 4090 GPU with 24 GB of VRAM. In addition, we compare our method with other approaches that incorporated Gaussian Splatting into their map representation, like [138]. Differently from us, they employed an octree-based map representation, which aids the splats initialization. Below, we report the datasets and the evaluation metrics employed. Our results, show that our method (supporting both RGB-D and point cloud inputs) offers improved performance and efficiency while maintaining a SOTA accuracy in 3D reconstruction.

Method	**ScanNet**		**Replica**		**Oxford-Spires**		**NC**	
	Time [s] ↓	FPS ↑	Time [s] ↓	FPS ↑	Time [s] ↓	FPS ↑	Time [s] ↓	FPS ↑
VDBFusion [132]	312	5.71	1038	1.93	113	24.65	206	9.61
PIN-SLAM [93]	120	14.80	136	14.7	213	13.15	226	8.79
N^3-Mapping [117]	330	5.40	–	–	716	3.91	–	–
NKSR [52]	–	–	–	–	45	62.28	–	–
VoxBlox [90]	236	7.57	746	2.68	173	16.17	323	6.14
Ours (single)	**27**	**64.30**	**40**	**48.9**	**40**	**68.86**	60	21.6
Ours (multi)	30	59.34	75	26	45	61	**56**	**35.25**

Table 8.1. Runtime comparison. Time [s] refers to the total computation time of the sequence, while FPS refers to processing speed. Our runtimes show an improvement from 2-3x to 13x compared to existing SOTA. Best results are highligthed as **first**, and **second**. Dash "-" indicates missing values due to failure.

8.4.1 3D Reconstruction

To evaluate mapping quality, we adopt standard surface reconstruction metrics as defined in [79]. Accuracy (Acc) measures the average distance from points on the reconstructed mesh to their nearest neighbors in the reference point cloud, while Completeness (Comp) captures the inverse—how well the reference surface is covered by the reconstruction. The Chamfer-L_1 distance (C-L_1) is the mean of these two and provides a balanced measure of geometric fidelity. Additionally, we compute the F-score with a 20 cm threshold, which evaluates geometric consistency by combining precision and recall into a single value. Our

method achieves reconstruction quality comparable to or better than prior approaches across multiple datasets. The results demonstrate that our variance-adaptive multi-resolution grid does not compromise accuracy, even while significantly reducing memory usage Tab. 8.3 and improving runtime Tab. 8.1. Notably, N^3-Mapping failed to complete any sequence from the Newer College Dataset (NCD) dataset, likely due to its high computational and memory requirements, which limit its scalability. For VoxBlox, we used the "FastTSDF" mode for RGB-D input, as the standard configuration was unstable or slow in our tests. For LiDAR data, we switched to the "MergedTSDF" configuration to ensure compatibility with sparser point clouds. These settings were chosen to reflect best-case performance for each method in its intended domain. To ensure a fair comparison, voxel sizes were adapted to sensor resolution: 0.01 m for dense RGB-D input and 0.20 m for sparse LiDAR input. This allows each method to operate under conditions that match the characteristics of its input, and helps isolate the effect of voxel structure and memory management. Tab. 8.2 presents the quantitative results, while Fig. 8.8 and Fig. 8.9 show qualitative examples. These visuals illustrate how our multi-resolution approach maintains surface detail in geometrically complex regions and avoids over-allocation in homogeneous areas, supporting both high accuracy and efficiency across sensor types.

8.4.2 Runtimes

Table 8.1 reports both the total processing time per sequence (in seconds) and the corresponding frame processing rate (FPS). Our approach outperforms all baselines in nearly every scenario. The single-resolution variant already yields significant speedups—up to 13× faster than prior state-of-the-art methods such as VDBFusion and VoxBlox. The multi-resolution configuration achieves additional performance gains, especially in large-scale or sparse scenes, while preserving reconstruction quality. Notably, our method maintains real-time or near-real-time performance across all datasets, including challenging LiDAR-based sequences from Newer College (NC). Competing methods such as N^3-Mapping failed to complete these sequences, highlighting the scalability and robustness of our approach. These improvements stem from three key design choices: (1) a flat hash table that enables constant-time access without hierarchical traversal overhead, (2) a fully GPU-based pipeline that avoids costly CPU-GPU data transfers, and (3) an adaptive voxel resolution strategy that reduces unnecessary computation and memory usage in low-detail regions. Together, these contribute to the significant runtime advantage of our system.

8.4.3 Memory

We report memory consumption indirectly by measuring the number of mesh vertices and faces produced during reconstruction, which closely correlates with storage requirements and runtime memory usage. Table 8.3 summarizes average values across both RGB-D and LiDAR datasets. Our method, particularly in its multi-resolution configuration, produces significantly more compact reconstructions compared to existing approaches. In the RGB-D setup, we observe memory savings ranging from 2.0× to 4.5×, while for LiDAR data, reductions range from 1.03× to 4.3×, depending on the baseline. This demonstrates the effectiveness of our variance-driven voxel allocation, which allows the system to retain high resolution in geometrically complex areas while using coarser voxels in flat or redundant regions. These gains are made possible by our design, which avoids over-allocating memory in

		0000	0010	0059	0106	0109	0181	0207	quad	math	cloi	Avg RGB-D	Avg LiDAR	Avg
		RGB-D							***LiDAR***					
PIN-SLAM	Acc.[cm]↓	2.680	4.069	5.789	6.540	2.802	4.161	4.004	9.942	10.271	9.082	4.292	9.765	5.934
	Comp.[cm]↓	1.078	1.060	1.464	1.732	0.862	1.648	1.085	13.764	13.960	14.584	1.276	14.103	5.124
	C-L1↓	1.879	2.565	3.626	4.136	1.832	2.905	2.544	11.853	12.116	11.833	2.784	11.934	5.529
	F-score[%]↑	97.966	94.486	87.775	83.363	96.209	93.494	93.496	83.724	83.596	82.431	92.398	83.250	89.654
VDBFusion	Acc.[cm]↓	2.217	2.473	4.615	5.472	1.603	4.090	2.874	**6.511**	8.669	**6.511**	3.336	**7.230**	4.504
	Comp.[cm]↓	**0.925**	**0.707**	**1.122**	**0.885**	**0.516**	**0.928**	**0.779**	**11.645**	**12.684**	**12.657**	**0.837**	**12.329**	**4.285**
	C-L1↓	1.571	1.590	2.869	3.179	**1.059**	2.509	1.826	**9.078**	**10.677**	**9.584**	2.086	**9.780**	4.394
	F-score[%]↑	97.052	95.687	88.896	85.161	97.659	89.723	94.692	**89.532**	**87.157**	**87.963**	92.696	**88.217**	91.352
VoxBlox	Acc.[cm]↓	2.120	2.718	4.156	4.137	2.583	4.246	2.678	9.824	**8.619**	9.284	3.234	9.242	5.037
	Comp.[cm]↓	1.759	6.168	2.145	2.304	9.045	2.638	3.817	17.083	14.756	26.207	3.982	19.349	8.592
	C-L1↓	1.939	4.443	3.150	3.220	5.814	3.442	3.247	13.453	11.688	17.746	3.608	14.296	6.814
	F-score[%]↑	96.076	89.194	89.313	88.593	84.653	91.204	91.329	79.400	83.699	63.518	90.052	75.539	85.698
N^3-Mapping	Acc.[cm]↓	fail	1.705	**2.338**	3.374	1.706	**2.134**	1.786	fail	fail	fail	2.174	-	-
	Comp.[cm]↓	fail	1.335	1.664	1.390	1.133	2.616	1.222	fail	fail	fail	1.560	-	-
	C-L1↓	fail	1.520	**2.001**	2.382	1.420	**2.375**	1.504	fail	fail	fail	1.867	-	-
	F-score[%]↑	fail	**98.811**	**96.398**	93.671	98.076	**97.034**	97.293	fail	fail	fail	96.880	-	-
Ours	Acc.[cm]↓	**0.963**	**1.587**	2.981	**2.822**	**1.364**	2.789	**1.701**	7.134	9.061	7.015	**2.030**	7.737	**3.742**
	Comp.[cm]↓	1.053	1.079	1.414	1.202	0.961	1.968	1.234	11.898	13.116	14.208	1.273	13.074	4.813
	C-L1↓	**1.008**	**1.333**	2.198	**2.012**	1.162	2.379	**1.366**	9.516	11.088	10.612	**1.637**	10.405	**4.267**
	F-score[%]↑	**99.674**	98.408	94.124	**94.200**	**98.079**	96.183	**97.246**	89.503	85.811	84.815	**96.845**	86.710	**93.804**

Table 8.2. 3D Reconstruction Results. The RGB-D data are sequences of ScanNet [21] and LiDAR data are sequences of NCD [156]. All the pipelines are run with ground truth fixed poses. Voxel size is set to 1 cm for RGB-D data and 20 cm for LiDAR data and the F-score is computed with a 10 cm error threshold for the RGB-D and 20 cm for the LiDAR. Best results are highligthed as **first**, and **second**. N^3-Mapping fails on one RGB-D sequence and all the LiDAR sequences; for completeness, we report the average for RGB-D despite is calculated only on its working sequences.

uniform areas through adaptive resolution, and by our use of a flat hash table that eliminates the overhead associated with hierarchical structures like octrees or VDBs. As a result, our approach achieves better memory scalability while maintaining high reconstruction quality.

8.4.4 Rendering

To evaluate the rendering performance of our pipeline, we integrate our voxel grid with a Gaussian Splatting (GS) renderer and compare against GS-Fusion [138], a recent method designed for efficient point-based rendering. Tab. 8.4 reports standard perceptual metrics—PSNR, SSIM, and LPIPS—as well as rendering speed in frames per second (FPS). Our multi-resolution configuration achieves consistently better results across all quality metrics. This improvement is attributed to our adaptive voxel grid, which maintains high-resolution detail in complex regions while avoiding redundant splat generation in homogeneous areas. This leads to more accurate reconstructions and better perceptual fidelity in rendered images.

Method	RGB-D		LiDAR	
	# Vertices ↓	# Faces ↓	# Vertices ↓	# Faces ↓
VDBFusion	2738750	5166635	632804	1144472
PIN-SLAM	5137959	9508670	2676274	4812832
VoxBlox	6112865	7076666	1776286	1791420
Ours (single)	4718172	2949443	753320	1246638
Ours (multi)	**1345825**	**2224068**	**615204**	**1035968**

Table 8.3. Average memory usage. We report the number of vertices and faces as a proxy for memory consumption, as they correlate with mesh complexity and storage requirements. Our multi-resolution setup improves from 2.03x to 4.5x in the RGB-D setup and from 1.03x to 4.3x in the LiDAR setup. Best results are highligthed as **first**, and **second**.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	FPS ↑
GS-Fusion	34.65	0.949	0.056	14.99
Ours	33.90	0.949	0.057	**28.0475**
Ours (+iteration)	34.27	0.951	**0.052**	14.218
Ours (multi)	**35.73**	**0.960**	**0.044**	23.703

Table 8.4. Rendering quality and performance. Comparison of perceptual metrics (PSNR, SSIM, LPIPS) and rendering speed (FPS). Best results are highligthed as **first**, **second**, and **third**. The multi-resolution setup improves overall rendering quality in the GS pipeline, with a small computational overhead. Nevertheless, our GPU-based quad-tree implementation achieves better overall performance compared to other GS-Fusion.

In terms of performance, our base version achieves the highest FPS (28.05), demonstrating the strength of our fully GPU-based quad-tree rendering mechanism. Unlike prior methods that rely on fixed splat densities or CPU-side pre-processing, our implementation leverages a parallel quad-tree structure to dynamically manage splat density at runtime. This enables both high-quality and real-time rendering, trading off efficiency and visual quality.

8.5 Conclusions

This chapter describes our novel multi-resolution voxel grid framework for real-time 3D reconstruction that adapts resolution based on the local variance of TSDF observations. Unlike traditional octree or VDB-based methods, our approach leverages a flat hash table to manage voxel blocks of varying resolution, enabling constant-time access and full GPU parallelization. This design supports efficient integration and rendering while remaining sensor-agnostic, handling both RGB-D and LiDAR inputs. Our experiments demonstrate that the proposed method achieves competitive reconstruction quality, with up to 4× memory savings and 13× speedup compared to fixed-resolution baselines. The integration with a GPU-accelerated quad-tree further enables high-performance rendering through Gaussian Splatting, balancing visual fidelity and efficiency. By combining adaptive resolution, scalability, and real-time performance, our method offers a practical and extensible so-

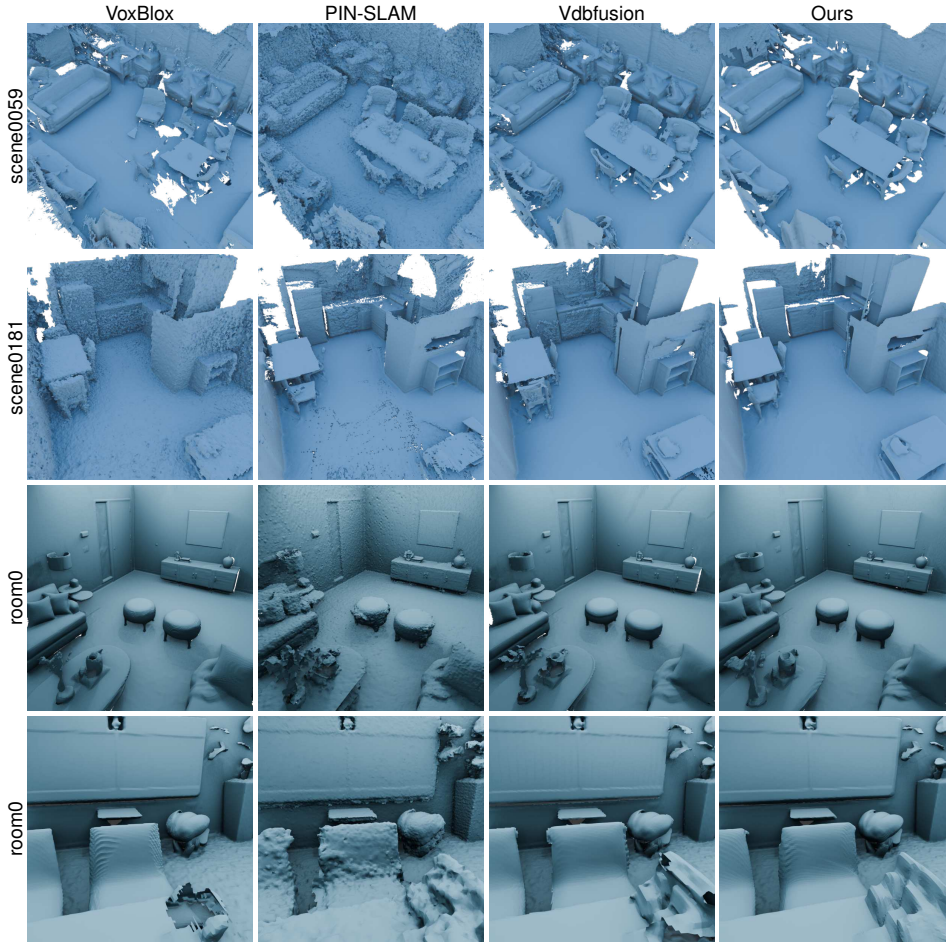


Figure 8.8. Qualitative results on RGBD Datasets: The images show the mapping results on scene0059 and scene0181 of ScanNet dataset [21] and sequence room0 of Replica dataset [119]. We compare our multi-resolution method against other SOTA.

lution for large-scale 3D mapping in graphics and robotics applications. We release our implementation as open-source to encourage further research and adoption.

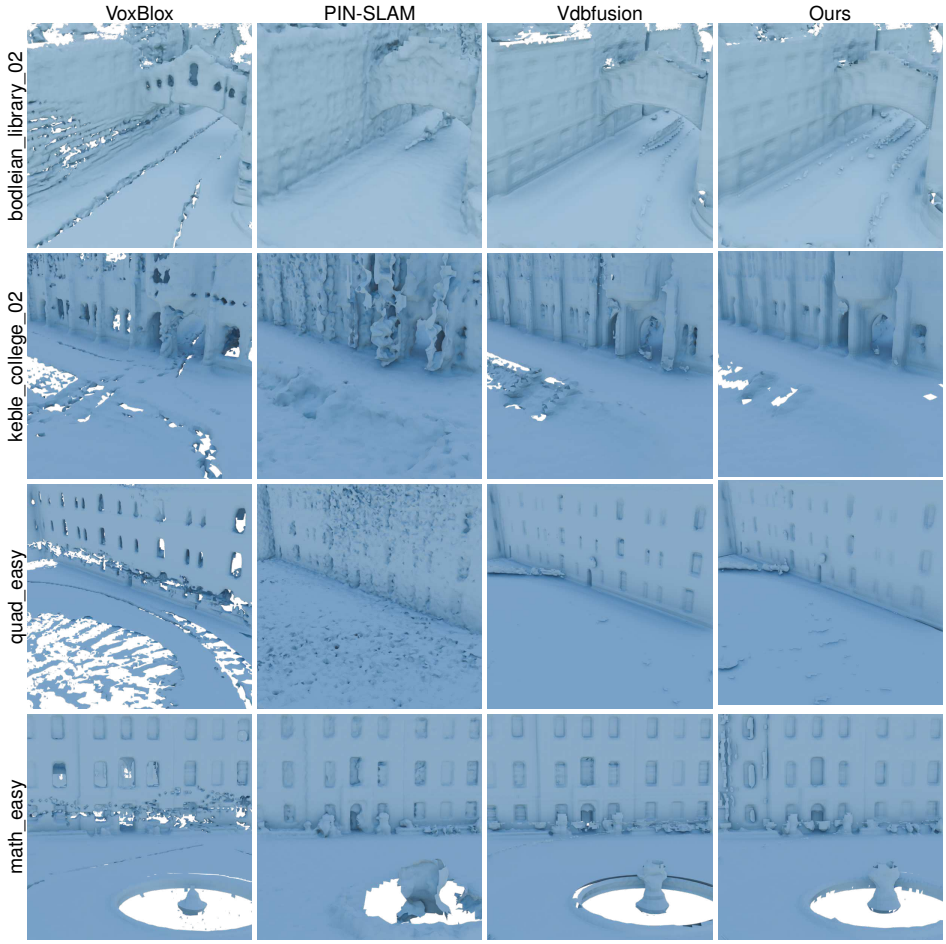


Figure 8.9. Qualitative results on LiDAR Datasets: The images show the mapping results on sequences bodleian-library-02 and keble-college-02 Oxford-Spires [123] and sequences quad-easy and math-easy of NCD [156]. We compare our multi-resolution method against other SOTA.

Chapter 9

Conclusion

This thesis presented several steps directed towards a unified approach for 3D reconstruction that leverages both RGB cameras and LiDARs. Specifically, a great effort was put to bring the camera methodologies within the LiDAR framework[1].

Starting from the fundamentals, Chapter 3 introduces an effective calibration toolbox for estimating the LiDAR-camera extrinsics. Compared to other approaches, ours enable accurate estimates while only requiring a planar calibration target (i.e., a chessboard), widely employed for camera calibration.

In Chapter 4 we presented a photometric method for refining SLAM trajectories that handles LiDARs and cameras uniformly.

Chapter 5 introduces the rolling-shutter (or skewing) effect observable on spinning LiDARs during high velocity motions. The problem is resolved using an ICP-like formulation that recovers associations between successive clouds and estimates the intra-scan sensor velocity, providing the means to compensate the rolling-shutter bias and continuous trajectory estimation.

In Chapter 6 we introduced our dataset recorded in Rome that offers synchronized visual, geometric and inertial measurements. Additionally, using the knowledge of Chap. 4, we achieved highly accurate ground truth trajectories via LiDAR-Inertial Measurement Unit (IMU)-GPS fusion. The proposed sequences allow the community to evaluate new SLAM methods on indoor-outdoor, structured-unstructured scenarios. Additionally, we include a sequence ran on multiple floor which can still pose challenges on SLAM systems. The large scale and dynamics provided through our sequences, opens up new challenges within the context of Simultaneous Localization and Mapping (SLAM) and 3D reconstruction.

Chapter 7 closes the gap between unified Gaussian representation between cameras and LiDARs. Specifically, We reformulate the rasterizer for 2D Gaussian Splatting to enable spherical projections. We validated this formulation by developing an odometry and mapping pipeline that offers comparable results in terms of trajectory estimation and SOTA results in mapping.

Finally, Chapter 8 presents a new architecture for integrating RGB-D and LiDAR measurements within a Truncated Signed Distance Function (TSDF) representation. Compared to other approaches, ours leverage a flat hash table to manage multi-resolution voxel blocks, enabling efficient access and full GPU parallelization. This translates in effective memory savings and strong speed-ups compared to other baselines. Additionally, the integration of a

[1] Starting from the spherical projection model which "unifies" the measurement dimensionality.

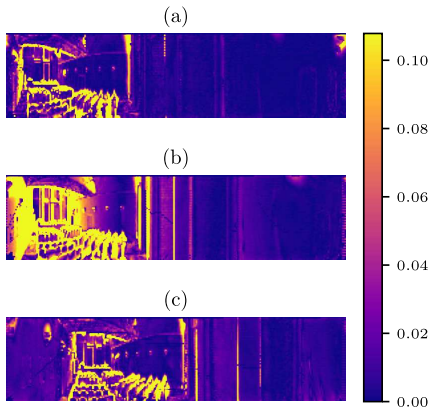


Figure 9.1. Rolling-shutter effect shown on spherical GS rasterizer. The figure depicts the range L-1 error between the prediction and the LiDAR measurement before a pure rotation (a), during (b) and after (c). Qualitatively, the intra-rotation measure exhibits higher error on far regions. The most likely cause for this effect is the rolling-shutter distortion effect.

GPU-accelerated quad-tree enables high performance, incremental appearance reconstruction via 3D Gaussian Splatting.

9.1 Outlook

In this section, We provide some comments on the future research of a unified 3D reconstruction methodology. Each paragraph will cover some specific issues or ideas that I believe should be explored within the near-future to move one step closer to a viable solution.

Improving LiDAR-only Gaussian Splatting: The results presented in Chap. 7 can be greatly improved if we tackle some issues related to Gaussian Splatting. Specifically, we noticed three critical aspects.

First, despite our adaptive densification strategy targets high-error or unobserved areas, we still face some overpopulation issues. To this end, leveraging a proper structure to handle the Gaussian density would greatly benefit the pipeline in terms of accuracy, speed, and memory consumption. Recently, Wu *et al.* [144] presented a pruning strategy based on a *Multi-level Occupancy Hash Voxel* structure that reduces the number of redundant and overlapping Gaussians within the scene.

Another serious issue is related to aliasing of the primitives. For this to happen, the projected Gaussian size should be comparable or smaller than a pixel within the spherical image. While this typically does not happen for room-like scenarios, the large Field Of View (FOV) of LiDARs while exploring large scenarios drastically increases this chance. When initializing a scene, the close, detailed regions are seeded with many small Gaussians. Once the sensor is far enough, the projection of those gaussians becomes so small that the rasterizer may miss them. When this happens, typically details becomes smoothed and are often lost. A straightforward explanation of this effect is that, casting one ray per pixel can under-account the details while optimizing the representation. Within the Neural Radiance Field (NeRF) framework, Mip-NeRF [5] contrasts this effect by rendering anti-aliased conical frustums instead of rays, yielding better scores on multiscale sequences. Similarly, Mip-Splatting [150] presents a similar formulation for Gaussian Splatting methods. A

thorough analysis and implementation of similar solutions within our framework could greatly benefit the mapping accuracy.

Finally, as described in Chap. 5, motion distortion affects in a way or another all spinning LiDARs. Fig. 9.1 shows the effect of skewing across our pipeline. Compared to common approaches that tackle the LiDAR from a 3D perspective, projection based methods are likely more subject to this type of distortion and therefore, it should be accounted accordingly.

Tightly-coupled LiDAR-camera 3D Reconstruction: Gaussian Splatting is extremely sensitive to initial seeding. In the original implementation, this process is guided from initialized Structure from Motion (SfM) triangulated points because these lie on well-posed 3D locations. In the context of incremental large-scale reconstruction, RGB-D cameras typically does not have the range to cover the whole image, causing either partial reconstructions or geometrically inconsistent representation. On the other hand, the results of Chap. 7 could play a beneficial role in seeding RGB reconstruction by providing a dense, detailed initial guess. Furthermore, LiDAR GS could benefit from refinements provided by a dense RGB sensor. These claims hint at a possible reconstruction strategy that tightly couple LiDAR and RGB sensors. Results in Chap. 4 already demonstrated the advantages of merging the two systems.

Handling dynamics: In the field of SLAM and 3D Reconstruction, a common assumption is that the entirety of the observable world to be static. When this assumption is broken, two conditions may arise. If structured dynamics accounts for less than 50% of the measurements, then typically, via robust M-estimators or consensus based techniques, they can be classified as outliers and filtered away. Otherwise, we typically face the *consensus inversion* phenomenon in which our estimate may be tricked to assume that component to be the static one[2].

In this paragraph, we will not cover the latter case which is still an open problem within the community. However, in many situations such as simulation or high-abstraction tasks, being able to track objects is particularly useful. Tracking and refining instead of discarding seems like a strong and interesting field of research. Gaussian Splatting once again seems a reasonable choice for such tasks (i.e., periodic vibrating Gaussians [17] or via deformation fields [142, 33]). Integrating dynamic handling into a LiDAR-camera unified reconstruction system, could bridge the gap for GS-based simulations.

[2]This can happen to us too! (i.e., we are inside a static train, and we see another train outside the window that slowly starts moving. Is that train moving or is it ours ?).

Appendices

Appendix A

Appendix

A.1 Proof of Rotation Matrices

Starting from Eq. (2.8), write the dot between two columns of $\mathbf{R}$. The same rule apply for the other combinations of columns.

$$\begin{aligned}\mathbf{x}'^T\mathbf{y}' &= \begin{pmatrix}\mathbf{x}'^T\mathbf{x} & \mathbf{x}'^T\mathbf{y} & \mathbf{x}'^T\mathbf{z}\end{pmatrix}\begin{pmatrix}\mathbf{y}'^T\mathbf{x} & \mathbf{y}'^T\mathbf{y} & \mathbf{y}'^T\mathbf{z}\end{pmatrix} \\ &= \mathbf{x}'^T\mathbf{x}\mathbf{y}'^T\mathbf{x} + \mathbf{x}'^T\mathbf{y}\mathbf{y}'^T\mathbf{y} + \mathbf{x}'^T\mathbf{z}\mathbf{y}'^T\mathbf{z} \\ &= \mathbf{x}'^T\left(\mathbf{x}\mathbf{x}^T + \mathbf{y}\mathbf{y}^T + \mathbf{z}\mathbf{z}^T\right)\mathbf{y}' \\ &= \mathbf{x}'^T\mathbf{I}\mathbf{y}'^T = \mathbf{x}'^T\mathbf{y}' = \mathbf{0}\end{aligned} . \tag{A.1}$$

A.2 Spherical 2DGS Rasterizer derivation

This section introduces the process for rasterizing a 2D Gaussian on the spherical frame defined in Sec. 2.3.2. This process assumes that tiling is already completed and we know which Gaussian (or set of Gaussians) has to be traversed by the selected pixel $\mathbf{u}$.

Recall that a Gaussian primitive $\mathbf{G} = (\boldsymbol{\mu}, \mathbf{q}, \mathbf{s}, o)$ its defined by an opacity $o \in \mathbb{R}$, and a covariance matrix $\boldsymbol{\Sigma}$ centered at $\boldsymbol{\mu} \in \mathbb{R}^3$ and decomposed into a rotation matrix $\mathbf{R} \in \mathbb{SO}(3)$ represented by the quaternion $\mathbf{q}$, and a scaling matrix $\mathbf{S} = \text{diag}(\mathbf{s})$. Additionally, recall that $\mathbf{H}$ is the mapping between splat-space and world-space defined in Eq. (2.84) as

$$\mathbf{H} = \begin{pmatrix} s_\alpha \mathbf{t}_\alpha & s_\beta \mathbf{t}_\beta & \mathbf{0} & \boldsymbol{\mu} \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{A.2}$$

To ease the process of derivation, let's identify the columns of $\mathbf{R} = (\mathbf{r}_\alpha, \mathbf{r}_\beta, \mathbf{r}_n)$. Let ${}^c\mathbf{T}_w \in \mathbb{SE}(3)$ be the transform from world-space to sensor-space. The mapping from splat-space to sensor-space is defined as

$$\mathbf{T}_{4\times3} = {}^c\mathbf{T}_w\mathbf{H} = \begin{pmatrix} \mathbf{b}_\alpha & \mathbf{b}_\beta & \mathbf{b}_c \\ 0 & 0 & 1 \end{pmatrix}, \tag{A.3}$$

where $\mathbf{b}_c = {}^c\mathbf{R}_w\boldsymbol{\mu} + {}^c\mathbf{t}_w$. In Eq. (A.3) we suppressed the third column of $\mathbf{H}$ which contains only zeros by construction.

Forward: Let $\mathbf{u} = (u, v)$ be a pixel of the spherical image. We compute the intersection point within the splat-space as described in Sec. 2.5.2 by factoring the two orthogonal planes as described in Sec. 7.3. The orthogonal planes are expressed in splat-space with

$$\mathbf{h}_\alpha = \mathbf{T}^T \mathbf{h}_x \qquad\qquad \mathbf{h}_\beta = \mathbf{T}^T \mathbf{h}_y, \tag{A.4}$$

and the intersection point $\hat{\mathbf{s}}$ in splat-space is

$$\hat{\mathbf{p}} = \mathbf{h}_\alpha \times \mathbf{h}_\beta \tag{A.5}$$

$$\hat{\mathbf{s}} = (\hat{s}_\alpha, \hat{s}_\beta)^T = \left(\frac{\hat{\mathbf{p}}_x}{\hat{\mathbf{p}}_z}, \frac{\hat{\mathbf{p}}_y}{\hat{\mathbf{p}}_z} \right). \tag{A.6}$$

We use $\hat{\mathbf{s}}$ to measure the Gaussian kernel at the intersection point $bG(\hat{\mathbf{s}})$ to compute the density

$$\alpha = o\mathbf{G}(\hat{\mathbf{s}}), \tag{A.7}$$

and then we compute the contribution depth [1] d as

$$\begin{aligned} \boldsymbol{\nu} &= \hat{s}_\alpha \mathbf{b}_\alpha + \hat{s}_\beta \mathbf{b}_\beta + \mathbf{b}_c \\ d &= \|\boldsymbol{\nu}\|. \end{aligned} \tag{A.8}$$

These quantities are α-blended as described in Eq. (2.88) and Eq. (2.89) to obtain the pixel depth and normals.

Gradients: Let $\frac{\partial \mathcal{L}}{\partial d}$ be the per-pixel channel derivative for depth and $\frac{\partial \mathcal{L}}{\partial \mathbf{n}}$ for normals. To improve the readability of this section, each partial derivative also includes its dimensions using the $\frac{\partial A}{\partial B}\Big|_{\dim(A)\times\dim(B)}$ notation. The derivation we show is for the k-th Gaussian over the M contributions to the pixel.

First, we derive the gradients with respect to the density α:

$$\left.\frac{\partial d}{\partial d_k}\right|_{1\times 1} = d_k A_k - \frac{B_{d,k}}{1-\alpha_k} \tag{A.9}$$

$$\left.\frac{\partial \mathbf{n}}{\partial \mathbf{n}_k}\right|_{1\times 3} = \mathbf{n}_k A_k - \frac{B_{\mathbf{n},k}}{1-\alpha_k} \tag{A.10}$$

$$\left.\frac{\partial \mathcal{L}}{\partial \alpha_k}\right|_{1\times 1} = \frac{\partial \mathcal{L}_d}{\partial d}\frac{\partial d}{\partial \alpha_k} + \frac{\partial \mathcal{L}_n}{\partial \mathbf{n}}\frac{\partial \mathbf{n}}{\partial \mathbf{n}_k}, \tag{A.11}$$

where $A_k = \prod_{i=1}^{k-1}(1-\alpha_i)$, $B_{d,k} = \sum_{i>k} d_i \alpha_i A_i$, and $B_{\mathbf{n},k} = \sum_{i>k} \mathbf{n}_i \alpha_i A_i$. We leverage the sorting of the primitives to efficiently compute these values during the back-propagation of the gradients. Furthermore, we propagate the gradients to the homogeneous transform

[1] Remember that in the context of LiDAR, depth actually describes the point distance from the origin and not the z-distance as for cameras.

matrix $\mathbf{T}$ from the intersection of planes $\hat{\mathbf{p}}_k$:

$$\begin{aligned}\left.\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{s}}_k}\right|_{1\times 2} &= \frac{\partial \mathcal{L}}{\partial \alpha}\frac{\partial \alpha}{\partial \hat{\mathbf{s}}_k} + \frac{\partial \mathcal{L}}{\partial d_k}\frac{\partial d_k}{\partial \hat{\mathbf{s}}_k} \\ &= -\frac{\partial \mathcal{L}}{\partial \alpha}\alpha_k \hat{\mathbf{s}}_k^T + \frac{\partial \mathcal{L}}{\partial d_k}\frac{\alpha_k A_k}{d_k}\begin{pmatrix}\boldsymbol{\nu}^T\mathbf{b}_\alpha \\ \boldsymbol{\nu}^T\mathbf{b}_\beta\end{pmatrix}^T\end{aligned} \tag{A.12}$$

$$\begin{aligned}\left.\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{p}}_k}\right|_{1\times 3} &= \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{s}}_k}\frac{\partial \hat{\mathbf{s}}_k}{\partial \hat{\mathbf{p}}_k} \\ &= \frac{1}{\hat{\mathbf{p}}_z}\begin{pmatrix}\partial \mathcal{L}/\partial \hat{\mathbf{s}}_\alpha \\ \partial \mathcal{L}/\partial \hat{\mathbf{s}}_\beta \\ -\frac{\hat{\mathbf{p}}_x \partial \mathcal{L}/\partial \hat{\mathbf{s}}_\alpha + \hat{\mathbf{p}}_y \partial \mathcal{L}/\partial \hat{\mathbf{s}}_\beta}{\hat{\mathbf{p}}_z}\end{pmatrix}^T.\end{aligned} \tag{A.13}$$

Thus, we can derive the gradients over the matrix $\mathbf{T}$. We keep the three accumulators $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_\alpha}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_\beta}$, and $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_c}$ decoupled to correctly integrate the contributions from each pixel:

$$\boldsymbol{\rho}_\alpha = \left(\frac{\partial \mathcal{L}}{\partial \mathbf{p}_k} \times \mathbf{h}_\alpha\right) \qquad \boldsymbol{\rho}_\beta = \left(\frac{\partial \mathcal{L}}{\partial \mathbf{p}_k} \times \mathbf{h}_\beta\right) \tag{A.14}$$

$$\begin{aligned}\left.\frac{\partial \mathcal{L}}{\partial \mathbf{b}_\alpha}\right|_{1\times 3} &= \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{p}}_k}\frac{\partial \hat{\mathbf{p}}_k}{\partial \mathbf{b}_\alpha} + \frac{\partial \mathcal{L}}{\partial d_k}\frac{\partial d_k}{\partial \mathbf{b}_\alpha} \\ &= -\boldsymbol{\rho}_{\beta,1}\mathbf{h}_x + \boldsymbol{\rho}_{\alpha,1}\mathbf{h}_y + \frac{\mathcal{L}}{\partial d_k}\frac{\hat{s}_\alpha}{d_k}\boldsymbol{\nu}^T\end{aligned} \tag{A.15}$$

$$\begin{aligned}\left.\frac{\partial \mathcal{L}}{\partial \mathbf{b}_\beta}\right|_{1\times 3} &= \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{p}}_k}\frac{\partial \hat{\mathbf{p}}_k}{\partial \mathbf{b}_\beta} + \frac{\partial \mathcal{L}}{\partial d_k}\frac{\partial d_k}{\partial \mathbf{b}_\beta} \\ &= -\boldsymbol{\rho}_{\beta,2}\mathbf{h}_x + \boldsymbol{\rho}_{\alpha,2}\mathbf{h}_y + \frac{\mathcal{L}}{\partial d_k}\frac{\hat{s}_\beta}{d_k}\boldsymbol{\nu}^T\end{aligned} \tag{A.16}$$

$$\begin{aligned}\left.\frac{\partial \mathcal{L}}{\partial \mathbf{b}_c}\right|_{1\times 3} &= \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{p}}_k}\frac{\partial \hat{\mathbf{p}}_k}{\partial \mathbf{b}_c} + \frac{\partial \mathcal{L}}{\partial d_k}\frac{\partial d_k}{\partial \mathbf{b}_c} \\ &= -\boldsymbol{\rho}_{\beta,3}\mathbf{h}_x + \boldsymbol{\rho}_{\alpha,3}\mathbf{h}_y + \frac{\mathcal{L}}{\partial d_k}\frac{1}{d_k}\boldsymbol{\nu}^T,\end{aligned} \tag{A.17}$$

where $\boldsymbol{\rho}_{k,i}$ is the i-th element of $\boldsymbol{\rho}_k$.

Finally, we compute the gradients w.r.t. the Gaussian parameters.

$$\left.\frac{\partial \mathcal{L}}{\partial \mathbf{R}_k}\right|_{3\times 3} = \left(s_\alpha \frac{\partial \mathcal{L}}{\partial \mathbf{b}_\alpha}^T \quad s_\beta \frac{\partial \mathcal{L}}{\partial \mathbf{b}_\beta}^T \quad \frac{\partial \mathcal{L}}{\partial \mathbf{n}_k}^T\right) \mathbf{R}_w^c \tag{A.18}$$

$$\left.\frac{\partial \mathcal{L}}{\partial \mathbf{s}_k}\right|_{1\times 2} = \left(\frac{\partial \mathcal{L}}{\partial \mathbf{b}_\alpha}\mathbf{R}_w^c\mathbf{R}_{[1]} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}_\alpha}\mathbf{R}_w^c\mathbf{R}_{[1]}\right) \tag{A.19}$$

$$\left.\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k}\right|_{1\times 3} = \frac{\partial \mathcal{L}}{\partial \mathbf{b}_c}\mathbf{R}_w^c, \tag{A.20}$$

$$\left.\frac{\partial \mathcal{L}}{\partial o_k}\right|_{1\times 1} = \frac{\partial \mathcal{L}}{\partial \alpha_k} \exp\left(-\frac{1}{2}\mathbf{s}_k^T\mathbf{s}_k\right) \tag{A.21}$$

where $\mathbf{R}_{[i]}$ is the i-th column of $\mathbf{R}$.

Bibliography

[1] AL-NUAIMI, A., LOPES, W., ZELLER, P., GARCEA, A., LOPES, C., AND STEINBACH, E. Analyzing lidar scan skewing and its impact on scan matching. In *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8. IEEE (2016). 53, 55

[2] ALISMAIL, H., BROWNING, B., AND LUCEY, S. Photometric bundle adjustment for vision-based slam. In *Proc. of the Asian Conf. on Computer Vision (ACCV)*, pp. 324–341. Springer (2017). 41

[3] AMANATIDES, J. AND WOO, A. A Fast Voxel Traversal Algorithm for Ray Tracing. (1987). 90, 92

[4] BAI, Z., JIANG, G., AND XU, A. Lidar-camera calibration using line correspondences. *IEEE Sensors Journal*, **20** (2020). doi:10.3390/s20216319. 28

[5] BARRON, J. T., MILDENHALL, B., VERBIN, D., SRINIVASAN, P. P., AND HEDMAN, P. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 5470–5479 (2022). 25, 102

[6] BEHLEY, J. AND STACHNISS, C. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*. Robotics: Science and Systems Foundation (2018). ISBN 978-0-9923747-4-7. doi:10.15607/RSS.2018.XIV.016. 49, 75, 77

[7] BELTRAN, J., GUINDEL, C., DE LA ESCALERA, A., AND GARCIA, F. Automatic extrinsic calibration method for LiDAR and camera sensor setups. *IEEE Trans. on Intelligent Transportation Systems (ITS)*, **23** (2022), 17677. Available from: https://ieeexplore.ieee.org/document/9733276/, doi:10.1109/TITS.2022.3155228. xvii, 29, 33, 34

[8] BESL, P. AND MCKAY, N. D. A method for registration of 3-D shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, **14** (1992), 239. doi:10.1109/34.121791. 75, 77

[9] BLINN, J. F. A homogeneous formulation for lines in 3 space. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pp. 237–241 (1977). 24

[10] BORRMANN, D., ELSEBERG, J., LINGEMANN, K., NÜCHTER, A., AND HERTZBERG, J. Globally consistent 3D mapping with scan matching. *Journal on Robotics and Autonomous Systems (RAS)*, **56** (2008), 130. doi:10.1016/j.robot.2007.07.002. 75, 77

[11] BRADSKI, G. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, **25** (2000), 120. 27, 32, 67

[12] BRIZI, L., GIACOMINI, E., GIAMMARINO, L. D., FERRARI, S., SALEM, O., REBOTTI, L. D., AND GRISETTI, G. VBR: A Vision Benchmark in Rome. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 15868–15874 (2024). doi:10.1109/ICRA57147.2024.10611395. xiv, 4, 83, 84

[13] BURNETT, K., ET AL. Boreas: A multi-season autonomous driving dataset. *Intl. Journal of Robotics Research (IJRR)*, **42** (2023), 33. arXiv:https://doi.org/10.1177/02783649231160195, doi:10.1177/02783649231160195. 63

[14] BURRI, M., NIKOLIC, J., GOHL, P., SCHNEIDER, T., REHDER, J., OMARI, S., ACHTELIK, M. W., AND SIEGWART, R. The euroc micro aerial vehicle datasets. *Intl. Journal of Robotics Research (IJRR)*, **35** (2016), 1157. 64, 67

[15] CAMPOS, C., ELVIRA, R., RODRÍGUEZ, J. J. G. M., MONTIEL, J. M., AND TARDÓS, J. D. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Trans. on Robotics (TRO)*, **37** (2021), 1874. 73

[16] CHEN, Q., YANG, S., DU, S., TANG, T., CHEN, P., AND HUO, Y. LiDAR-GS:Real-time LiDAR Re-Simulation using Gaussian Splatting (2024). arXiv:2410.05111, doi:10.48550/arXiv.2410.05111. 77

[17] CHEN, Y., GU, C., JIANG, J., ZHU, X., AND ZHANG, L. Periodic vibration gaussian: Dynamic urban scene reconstruction and real-time rendering. *arXiv preprint*, (2023). 103

[18] COLOSI, M., ALOISE, I., GUADAGNINO, T., SCHLEGEL, D., CORTE, B. D., ARRAS, K. O., AND GRISETTI, G. Plug-and-play slam: A unified slam architecture for modularity and ease of use. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 5051–5057 (2020). doi:10.1109/IROS45743.2020.9341611. 59

[19] CORTE, B. D., BOGOSLAVSKYI, I., STACHNISS, C., AND GRISETTI, G. A general framework for flexible multi-cue photometric point cloud registration. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 1–8 (2018). 39, 44

[20] CURLESS, B. AND LEVOY, M. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, p. 303–312. Association for Computing Machinery, New York, NY, USA (1996). ISBN 0897917464. Available from: https://doi.org/10.1145/237170.237269, doi:10.1145/237170.237269. 87, 88, 91

[21] DAI, A., CHANG, A. X., SAVVA, M., HALBER, M., FUNKHOUSER, T., AND NIESSNER, M. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2432–2443 (2017). doi:10.1109/CVPR.2017.261. xv, xviii, 95, 97, 99

[22] DAI, A., NIESSNER, M., ZOLLHÖFER, M., IZADI, S., AND THEOBALT, C. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans. on Graphics*, **36** (2017), 1. 41, 46

[23] DAI, P., XU, J., XIE, W., LIU, X., WANG, H., AND XU, W. High-quality Surface Reconstruction using Gaussian Surfels. In *ACM SIGGRAPH 2024 Conference Papers*, SIGGRAPH '24, pp. 1–11. Association for Computing Machinery, New York, NY, USA (2024). ISBN 979-8-4007-0525-0. doi:10.1145/3641519.3657441. 76

[24] DELAUNOY, A. AND POLLEFEYS, M. Photometric bundle adjustment for dense multi-view 3d modeling. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1486–1493 (2014). 41

[25] DELLENBACH, P., DESCHAUD, J.-E., JACQUET, B., AND GOULETTE, F. CT-ICP: Real-time Elastic LiDAR Odometry with Loop Closure . In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 5580–5586 (2022). doi:10.1109/ICRA46639.2022.9811849. 75, 77

[26] DEMMEL, N., GAO, M., LAUDE, E., WU, T., AND CREMERS, D. Distributed photometric bundle adjustment. In *Proc. of the International Conference on 3D Vision (3DV)*, pp. 140–149. IEEE (2020). 41

[27] DENG, J., WU, Q., CHEN, X., XIA, S., SUN, Z., LIU, G., YU, W., AND PEI, L. NeRF-LOAM: Neural Implicit Representation for Large-Scale Incremental LiDAR Odometry and Mapping. In *iccv*, pp. 8184–8193 (2023). doi:10.1109/ICCV51070.2023.00755. 76, 77, 82

[28] DI GIAMMARINO, L., BRIZI, L., GUADAGNINO, T., STACHNISS, C., AND GRISETTI, G. MD-SLAM: Multi-cue Direct SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 11047–11054. IEEE (2022). doi:10.1109/IROS47612.2022.9981147. xii, 40, 42, 46, 75, 77

[29] DI GIAMMARINO, L., GIACOMINI, E., BRIZI, L., SALEM, O., AND GRISETTI, G. Photometric LiDAR and RGB-D Bundle Adjustment. *ral*, **8** (2023), 4362. doi:10.1109/LRA.2023.3281907. 4, 69, 77

[30] DORNAIKA, F. AND HORAUD, R. Simultaneous robot-world and hand-eye calibration. *tra*, **14** (1998), 617. 68

[31] ERIKSSON, A., BASTIAN, J., CHIN, T., AND ISAKSSON, M. A consensus-based framework for distributed bundle adjustment. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1754–1762 (2016). 41

[32] FAN, S., YU, Y., XU, M., AND ZHAO, L. High-precision external parameter calibration method for camera and lidar based on a calibration device. *IEEE Access*, **11** (2023), 18750. Available from: https://ieeexplore.ieee.org/document/10049434/, doi:10.1109/ACCESS.2023.3247195. 29

[33] FANG, J., YI, T., WANG, X., XIE, L., ZHANG, X., LIU, W., NIESSNER, M., AND TIAN, Q. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers* (2022). 103

[34] FERRARI, S., GIAMMARINO, L. D., BRIZI, L., AND GRISETTI, G. MAD-ICP: It is All About Matching Data – Robust and Informed LiDAR Odometry. *ral*, **9** (2024), 9175. doi:10.1109/LRA.2024.3456509. 75, 77, 81, 82

[35] FISCHLER, M. A. AND BOLLES, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, **24** (1981), 381–395. Available from: https://doi.org/10.1145/358669.358692, doi:10.1145/358669.358692. 16, 29

[36] FORSTER, C., CARLONE, L., DELLAERT, F., AND SCARAMUZZA, D. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, **33** (2016), 1. 55

[37] FORSTER, C., ZHANG, Z., GASSNER, M., WERLBERGER, M., AND SCARAMUZZA, D. Svo: Semidirect visual odometry for monocular and multicamera systems. *tro*, **33** (2016), 249. 41

[38] FUNK, N., TARRIO, J., PAPATHEODOROU, S., POPOVIĆ, M., ALCANTARILLA, P. F., AND LEUTENEGGER, S. Multi-Resolution 3D Mapping With Explicit Free Space Representation for Fast and Accurate Mobile Robot Motion Planning. *IEEE Robotics and Automation Letters*, **6** (2021), 3553. doi:10.1109/LRA.2021.3061989. 89, 92

[39] GARRIDO-JURADO, S., NOZ SALINAS, R. M., MADRID-CUEVAS, F., AND MARÍN-JIMÉNEZ, M. Automatic generation and detection of highly reliable fiducial markers under occlusion. *pr*, **47** (2014), 2280. Available from: https://www.sciencedirect.com/science/article/pii/S0031320314000235, doi:https://doi.org/10.1016/j.patcog.2014.01.005. 27

[40] GEIGER, A., LENZ, P., AND URTASUN, R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361. IEEE (2012). doi:10.1109/CVPR.2012.6248074. 63, 67, 72, 73

[41] GIACOMINI, E., BRIZI, L., DI GIAMMARINO, L., SALEM, O., PERUGINI, P., AND GRISETTI, G. Ca2lib: Simple and accurate lidar-rgb calibration using small common markers. *sensors*, **24** (2024). Available from: https://www.mdpi.com/1424-8220/24/3/956, doi:10.3390/s24030956. 4

[42] GIACOMINI, E., GIAMMARINO, L. D., REBOTTI, L. D., GRISETTI, G., AND OSWALD, M. R. Splat-loam: Gaussian splatting lidar odometry and mapping

(2025). Available from: https://arxiv.org/abs/2503.17491, arXiv:2503.17491. 4

[43] GOLDLÜCKE, B., AUBRY, M., KOLEV, K., AND CREMERS, D. A super-resolution framework for high-accuracy multiview reconstruction. *ijcv*, **106** (2014), 172. 41

[44] GRAMMATIKOPOULOS, L., PAPANAGNOU, A., VENIANAKIS, A., KALISPERAKIS, I., AND STENTOUMIS, C. An effective camera-to-lidar spatiotemporal calibration based on a simple calibration target. *Sensors*, **22** (2022). Available from: https://www.mdpi.com/1424-8220/22/15/5576, doi:10.3390/s22155576. 29

[45] GRISETTI, G., GUADAGNINO, T., ALOISE, I., COLOSI, M., CORTE, B. D., AND SCHLEGEL, D. Least squares optimization: From theory to practice. *Robotics*, **9** (2020), 51. Available from: https://www.mdpi.com/2218-6581/9/3/51, doi:10.3390/robotics9030051. 18, 45

[46] GUÉDON, A. AND LEPETIT, V. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 5354–5363 (2024). doi:10.1109/CVPR52733.2024.00512. 20, 76, 85

[47] HARTLEY, R. AND ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edn. (2004). 11, 12

[48] HE, L., JIN, Z., AND GAO, Z. De-skewing lidar scan for refinement of local mapping. *Sensors*, **20** (2020), 1846. 54, 55

[49] HONG, S., HE, J., ZHENG, X., AND ZHENG, C. LIV-GaussMap: LiDAR-Inertial-Visual Fusion for Real-Time 3D Radiance Field Map Rendering. *ral*, **9** (2024), 9765. doi:10.1109/LRA.2024.3400149. 76, 77

[50] HORN, B., HILDEN, H., AND NEGAHDARIPOUR, S. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, **5** (1988), 1127. 46, 59, 72

[51] HUANG, B., YU, Z., CHEN, A., GEIGER, A., AND GAO, S. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *ACM SIGGRAPH 2024 Conference Papers*, SIGGRAPH '24, pp. 1–11. Association for Computing Machinery, New York, NY, USA (2024). ISBN 979-8-4007-0525-0. doi:10.1145/3641519.3657428. 24, 76, 77, 80

[52] HUANG, J., GOJCIC, Z., ATZMON, M., LITANY, O., FIDLER, S., AND WILLIAMS, F. Neural Kernel Surface Reconstruction. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (cvpr)*, pp. 4369–4379. IEEE, Vancouver, BC, Canada (2023). ISBN 979-8-3503-0129-8. doi:10.1109/CVPR52729.2023.00425. 89, 95

[53] ILLINGWORTH, J. AND KITTLER, J. The adaptive hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-9** (1987), 690. doi:10.1109/TPAMI.1987.4767964. 27

[54] ISAACSON, S., KUNG, P.-C., RAMANAGOPAL, M., VASUDEVAN, R., AND SKINNER, K. A. LONER: LiDAR Only Neural Representations for Real-Time SLAM . *ral*, **8** (2023), 8042. doi:10.1109/LRA.2023.3324521. 76, 77

[55] JAIN, J., LI, J., CHIU, M., HASSANI, A., ORLOV, N., AND SHI, H. Oneformer: One transformer to rule universal image segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2023). xiii, 70

[56] JIANG, J., GU, C., CHEN, Y., AND ZHANG, L. GS-LiDAR: Generating Realistic LiDAR Point Clouds with Panoramic Gaussian Splatting. In *iclr* (2025). Available from: https://openreview.net/forum?id=RMaRBE9s2H. 77

[57] KANNALA, J. AND BRANDT, S. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28** (2006), 1335. doi:10.1109/TPAMI.2006.153. 35

[58] KATZ, S., TAL, A., AND BASRI, R. Direct visibility of point sets. *acmgraphics*, **26** (2007), 24–es. Available from: https://doi.org/10.1145/1276377.1276407, doi:10.1145/1276377.1276407. 35

[59] KAZHDAN, M., BOLITHO, M., AND HOPPE, H. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pp. 61–70. Eurographics Association, Goslar, DEU (2006). ISBN 978-3-905673-36-4. 85

[60] KEETHA, N., KARHADE, J., JATAVALLABHULA, K. M., YANG, G., SCHERER, S., RAMANAN, D., AND LUITEN, J. Splatam: Splat, track and map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024). 77

[61] KERBL, B., KOPANAS, G., LEIMKÜHLER, T., AND DRETTAKIS, G. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, **42** (2023), 139. doi:10.1145/3592433. 20, 22, 76, 78, 94

[62] KERL, C., STURM, J., AND CREMERS, D. Dense visual slam for rgb-d cameras. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 2100–2106 (2013). 39, 46

[63] KIM, E.-S. AND PARK, S.-Y. Extrinsic calibration between camera and lidar sensors by matching multiple 3d planes. *sensors*, **20** (2020). Available from: https://www.mdpi.com/1424-8220/20/1/52, doi:10.3390/s20010052. xvii, 30, 33, 34

[64] KIM, G., PARK, Y. S., CHO, Y., JEONG, J., AND KIM, A. Mulran: Multimodal range dataset for urban place recognition. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 6246–6253 (2020). 63, 67

[65] KLEIN, G. AND MURRAY, D. Parallel tracking and mapping for small ar workspaces. In *6th IEEE and ACM international symposium on mixed and augmented reality*, pp. 225–234. IEEE (2007). 40

[66] KOENIG, N. AND HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, vol. 3, pp. 2149–2154. IEEE (2004). 33

[67] LENGYEL, E. Transition Cells for Dynamic Multiresolution Marching Cubes. *Journal of Graphics, GPU, and Game Tools*, **15** (2010), 99. doi:10.1080/2151237X.2011.563682. 93

[68] LI, Q., CHEN, S., WANG, C., LI, X., WEN, C., CHENG, M., AND LI, J. LO-Net: Deep Real-Time Lidar Odometry. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 8465–8474 (2019). doi:10.1109/CVPR.2019.00867. 77

[69] LI, X., HE, F., LI, S., ZHOU, Y., XIA, C., AND WANG, X. Accurate and automatic extrinsic calibration for a monocular camera and heterogenous 3d LiDARs. *sensors*, **22** (2022), 16472. Available from: https://ieeexplore.ieee.org/document/9829219/, doi:10.1109/JSEN.2022.3189041. 29

[70] LIANG, Z., ZHANG, Q., FENG, Y., SHAN, Y., AND JIA, K. Gs-ir: 3d gaussian splatting for inverse rendering. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 21644–21653 (2024). 20

[71] LIAO, Y., XIE, J., AND GEIGER, A. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *Pattern Analysis and Machine Intelligence (PAMI)*, (2022). 67

[72] LISO, L., SANDSTRÖM, E., YUGAY, V., VAN GOOL, L., AND OSWALD, M. R. Loopy-SLAM: Dense Neural SLAM with Loop Closures. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 20363–20373 (2024). doi:10.1109/CVPR52733.2024.01925. 76, 77, 79

[73] LIU, Z. AND ZHANG, F. Balm: Bundle adjustment for lidar mapping. *ral*, **6** (2021), 3184. 42, 49

[74] LORENSEN, W. E. AND CLINE, H. E. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, p. 163–169. Association for Computing Machinery, New York, NY, USA (1987). ISBN 0897912276. Available from: https://doi.org/10.1145/37401.37422, doi:10.1145/37401.37422. 93

[75] LV, X., WANG, B., DOU, Z., YE, D., AND WANG, S. LCCNet: LiDAR and camera self-calibration using cost volume network. In *cvprw*, pp. 2888–2895. IEEE (2021). ISBN 978-1-66544-899-4. Available from: https://ieeexplore.ieee.org/document/9523093/, doi:10.1109/CVPRW53098.2021.00324. 28

[76] LYU, X., SUN, Y.-T., HUANG, Y.-H., WU, X., YANG, Z., CHEN, Y., PANG, J., AND QI, X. 3dgsr: Implicit surface reconstruction with 3d gaussian splatting. *ACM Transactions on Graphics*, **43** (2024), 1–12. Available from: http://dx.doi.org/10.1145/3687952, doi:10.1145/3687952. 87

[77] MADDERN, W., PASCOE, G., LINEGAR, C., AND NEWMAN, P. 1 year, 1000 km: The oxford robotcar dataset. *ijrr*, **36** (2017), 3. 63, 64, 67

[78] MATSUKI, H., MURAI, R., KELLY, P. H., AND DAVISON, A. J. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18039–18048 (2024). doi:10.1109/CVPR52733.2024.01708. 76, 77, 81

[79] MESCHEDER, L., OECHSLE, M., NIEMEYER, M., NOWOZIN, S., AND GEIGER, A. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (cvpr)*, pp. 4455–4465 (2019). doi:10.1109/CVPR.2019.00459. 84, 95

[80] MILDENHALL, B., SRINIVASAN, P. P., TANCIK, M., BARRON, J. T., RAMAMOORTHI, R., AND NG, R. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *eccv* (edited by A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm), Lecture Notes in Computer Science, pp. 405–421. Springer International Publishing, Cham (2020). ISBN 978-3-030-58452-8. doi:10.1007/978-3-030-58452-8_24. 22, 76

[81] MILDENHALL, B., SRINIVASAN, P. P., TANCIK, M., BARRON, J. T., RAMAMOORTHI, R., AND NG, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, **65** (2021), 99. 20

[82] MIRZAEI, F. M., KOTTAS, D. G., AND ROUMELIOTIS, S. I. 3d LIDAR–camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization. *ijrr*, **31** (2012), 452. Available from: http://journals.sagepub.com/doi/10.1177/0278364911435689, arXiv:https://doi.org/10.1177/0278364911435689, doi:10.1177/0278364911435689. 30

[83] MOOSMANN, F. AND STILLER, C. Velodyne slam. In *2011 ieee intelligent vehicles symposium (iv)*, pp. 393–398. IEEE (2011). 55

[84] MUR-ARTAL, R. AND TARDÓS, J. D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *tro*, **33** (2017), 1255. 40, 46

[85] MUSETH, K. Vdb: High-resolution sparse volumes with dynamic topology. *ACM transactions on graphics (TOG)*, **32** (2013), 1. doi:10.1145/2487228.2487235. 21, 22, 82, 86, 87, 89, 92, 95

[86] MUSETH, K. NanoVDB: A GPU-Friendly and Portable VDB Data Structure For Real-Time Rendering And Simulation. In *ACM SIGGRAPH 2021 Talks*, SIGGRAPH '21, pp. 1–2. Association for Computing Machinery, New York, NY, USA (2021). doi:10.1145/3450623.3464653. 92

[87] NEWCOMBE, R. A., ET AL. KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136 (2011). doi:10.1109/ISMAR.2011.6092378. 88

[88] NICOLAI, A., SKEELE, R., ERIKSEN, C., AND HOLLINGER, G. A. Deep Learning for Laser Based Odometry Estimation. In *RSS workshop Limits and Potentials*

of Deep Learning in Robotics, vol. 184, p. 1 (2016). Available from: https://api.semanticscholar.org/CorpusID:30369588. 77

[89] NIESSNER, M., ZOLLHÖFER, M., IZADI, S., AND STAMMINGER, M. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, **32** (2013), 1. 21, 88, 91, 92, 93

[90] OLEYNIKOVA, H., TAYLOR, Z., FEHR, M., SIEGWART, R., AND NIETO, J. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1366–1373. IEEE (2017). doi:10.1109/IROS.2017.8202315. 82, 89, 91, 92, 95

[91] OLSON, E. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pp. 3400–3407 (2011). doi:10.1109/ICRA.2011.5979561. 29

[92] PAN, Y., ZHONG, X., JIN, L., WIESMANN, L., POPOVIĆ, M., BEHLEY, J., AND STACHNISS, C. PINGS: Gaussian Splatting Meets Distance Fields within a Point-Based Implicit Neural Map. In *Proc. of Robotics: Science and Systems (RSS)* (2025). Available from: https://www.ipb.uni-bonn.de/wp-content/papercite-data/pdf/pan2025rss.pdf. 87

[93] PAN, Y., ZHONG, X., WIESMANN, L., POSEWSKY, T., BEHLEY, J., AND STACHNISS, C. PIN-SLAM: LiDAR SLAM Using a Point-Based Implicit Neural Representation for Achieving Global Map Consistency. *IEEE Transactions on Robotics*, **40** (2024), 4045. doi:10.1109/TRO.2024.3422055. 76, 77, 82, 83, 89, 95

[94] PANDEY, G., MCBRIDE, J. R., SAVARESE, S., AND EUSTICE, R. M. Automatic extrinsic calibration of vision and lidar by maximizing mutual information. *robotics*, **32** (2015), 696. 28

[95] PARK, Y., YUN, S., WON, C., CHO, K., UM, K., AND SIM, S. Calibration between color camera and 3d LIDAR instruments with a polygonal planar board. *sensors*, **14** (2014), 5333. Available from: http://www.mdpi.com/1424-8220/14/3/5333, doi:10.3390/s140305333. 29

[96] PUSZTAI, Z., EICHHARDT, I., AND HAJDER, L. Accurate calibration of multi-LiDAR-multi-camera systems. **18**, 2139. Available from: http://www.mdpi.com/1424-8220/18/7/2139, doi:10.3390/s18072139. 29

[97] PUSZTAI, Z. AND HAJDER, L. Accurate calibration of LiDAR-camera systems using ordinary boxes. In *iccvw*, pp. 394–402. IEEE (2017). ISBN 978-1-5386-1034-3. Available from: http://ieeexplore.ieee.org/document/8265264/, doi:10.1109/ICCVW.2017.53. 29

[98] QUENZEL, J. AND BEHNKE, S. Real-time Multi-Adaptive-Resolution-Surfel 6D LiDAR Odometry using Continuous-time Trajectory Optimization. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 5499–5506 (2021). doi:10.1109/IROS51168.2021.9636763. 75, 77

[99] RAMEZANI, M., WANG, Y., CAMURRI, M., WISTH, D., MATTAMALA, M., AND FALLON, M. The newer college dataset: Handheld lidar, inertial and vision with ground truth. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4353–4360 (2020). doi:10.1109/IROS45743.2020.9340849. 63, 65, 67, 68

[100] RUAN, J., LI, B., WANG, Y., AND SUN, Y. SLAMesh: Real-time LiDAR Simultaneous Localization and Meshing. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 3546–3552 (2023). doi:10.1109/ICRA48891.2023.10161425. 75, 77, 82, 83

[101] RUBLEE, E., RABAUD, V., KONOLIGE, K., AND BRADSKI, G. Orb: An efficient alternative to sift or surf. In *iccv*, pp. 2564–2571 (2011). 47

[102] SALEM, O. A. A. K., GIACOMINI, E., BRIZI, L., DI GIAMMARINO, L., AND GRISETTI, G. Enhancing lidar performance: Robust de-skewing exclusively relying on range measurements. In *AIxIA 2023 – Advances in Artificial Intelligence*, pp. 310–320. Springer Nature Switzerland (2023). ISBN 978-3-031-47546-7. 4

[103] SANDSTRÖM, E., LI, Y., VAN GOOL, L., AND OSWALD, M. R. Point-SLAM: Dense Neural Point Cloud-based SLAM. In *iccv*, pp. 18387–18398 (2023). doi:10.1109/ICCV51070.2023.01690. 76, 77

[104] SANDSTRÖM, E., TATENO, K., OECHSLE, M., NIEMEYER, M., GOOL, L. V., OSWALD, M. R., AND TOMBARI, F. Splat-SLAM: Globally Optimized RGB-only SLAM with 3D Gaussians (2024). arXiv:2405.16544, doi:10.48550/arXiv.2405.16544. 77, 79

[105] SCHÖNBERGER, J. L. AND FRAHM, J.-M. Structure-from-motion revisited. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2016). 19

[106] SCHÖNBERGER, J. L. AND FRAHM, J.-M. Structure-from-Motion Revisited. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 4104–4113 (2016). doi:10.1109/CVPR.2016.445. 76

[107] SCHÖNBERGER, J. L., ZHENG, E., FRAHM, J.-M., AND POLLEFEYS, M. Pixelwise View Selection for Unstructured Multi-View Stereo. In *eccv* (edited by B. Leibe, J. Matas, N. Sebe, and M. Welling), Lecture Notes in Computer Science, pp. 501–518. Springer International Publishing, Cham (2016). ISBN 978-3-319-46487-9. doi:10.1007/978-3-319-46487-9_31. 76

[108] SCHÖNBERGER, J. L., ZHENG, E., POLLEFEYS, M., AND FRAHM, J.-M. Pixelwise view selection for unstructured multi-view stereo. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)* (2016). 19

[109] SCHOPS, T., SATTLER, T., AND POLLEFEYS, M. Bad slam: Bundle adjusted direct rgbd-d slam. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 134–144 (2019). xii, 39, 40, 41, 44, 46, 63, 67

[110] SEGAL, A., HAEHNEL, D., AND THRUN, S. Generalized-ICP. In *rss*. Robotics: Science and Systems Foundation (2009). ISBN 978-0-262-51463-7. doi:10.15607/RSS.2009.V.021. 75, 77

[111] SHAN, T. AND ENGLOT, B. LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4758–4765. IEEE (2018). doi:10.1109/IROS.2018.8594299. 41, 42, 49, 54, 76

[112] SHAN, T., ENGLOT, B., MEYERS, D., WANG, W., RATTI, C., AND RUS, D. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, vol. abs/2007.00258, pp. 5135–5142. IEEE (2020). 55

[113] SIGG, C., WEYRICH, T., BOTSCH, M., AND GROSS, M. H. Gpu-based ray-casting of quadratic surfaces. In *PBG@ SIGGRAPH*, pp. 59–65 (2006). 24

[114] SINGANDHUPE, A., LA, H. M., AND HA, Q. P. Single frame lidar-camera calibration using registration of 3d planes. In *irc*, pp. 395–402. IEEE (2022). ISBN 978-1-66547-260-9. Available from: https://ieeexplore.ieee.org/document/10023616/, doi:10.1109/IRC55401.2022.00076. 29

[115] SLAVCHEVA, M., KEHL, W., NAVAB, N., AND ILIC, S. Sdf-2-sdf: Highly accurate 3d object reconstruction. In *eccv*, pp. 680–696. Springer (2016). 41

[116] SOLA, J., DERAY, J., AND ATCHUTHAN, D. A micro lie theory for state estimation in robotics. *arXiv preprint*, (2018). 18

[117] SONG, S., ZHAO, J., HUANG, K., LIN, J., YE, C., AND FENG, T. N^3-Mapping: Normal Guided Neural Non-Projective Signed Distance Fields for Large-Scale 3D Mapping. *IEEE Robotics and Automation Letters*, **9** (2024), 5935. doi:10.1109/LRA.2024.3396638. 76, 77, 82, 83, 89, 95

[118] STOYANOV, T., MAGNUSSON, M., ANDREASSON, H., AND LILIENTHAL, A. J. Fast and accurate scan registration through minimization of the distance between compact 3d ndt representations. *ijrr*, **31** (2012), 1377. 41

[119] STRAUB, J., ET AL. The Replica Dataset: A Digital Replica of Indoor Spaces (2019). arXiv:1906.05797, doi:10.48550/arXiv.1906.05797. xv, 92, 95, 99

[120] SUCAR, E., LIU, S., ORTIZ, J., AND DAVISON, A. J. iMAP: Implicit Mapping and Positioning in Real-Time. In *iccv*, pp. 6209–6218 (2021). doi:10.1109/ICCV48922.2021.00617. 76

[121] SUN, C., WEI, Z., HUANG, W., LIU, Q., AND WANG, B. Automatic targetless calibration for LiDAR and camera based on instance segmentation. *ral*, **8** (2022), 981. Available from: https://ieeexplore.ieee.org/document/9830826/, doi:10.1109/LRA.2022.3191242. 28

[122] TANG, J., CHEN, Y., NIU, X., WANG, L., CHEN, L., LIU, J., SHI, C., AND HYYPPÄ, J. Lidar scan matching aided inertial navigation system in gnss-denied environments. *Sensors*, **15** (2015), 16710. 55

[123] TAO, Y., MUÑOZ-BAÑÓN, M. Á., ZHANG, L., WANG, J., FU, L. F. T., AND FALLON, M. The Oxford Spires Dataset: Benchmarking Large-Scale LiDAR-Visual Localisation, Reconstruction and Radiance Field Methods (2024). arXiv:2411.10546, doi:10.48550/arXiv.2411.10546. xiv, xv, xviii, 82, 83, 95, 100

[124] TESCHNER, M., HEIDELBERGER, B., MÜLLER, M., POMERANTES, D., AND GROSS, M. H. Optimized Spatial Hashing for Collision Detection of Deformable Objects. In *8th International Fall Workshop on Vision, Modeling, and Visualization, VMV 2003, München, Germany, November 19-21, 2003* (edited by T. Ertl), pp. 47–54. Aka GmbH (2003). 92

[125] TOSI, F., ZHANG, Y., GONG, Z., SANDSTRÖM, E., MATTOCCIA, S., OSWALD, M. R., AND POGGI, M. How nerfs and 3d gaussian splatting are reshaping slam: a survey (2024). Available from: https://arxiv.org/abs/2402.13255, arXiv:2402.13255. 77

[126] TOTH, T., PUSZTAI, Z., AND HAJDER, L. Automatic LiDAR-camera calibration of extrinsic parameters using a spherical target. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 8580–8586. IEEE (2020). ISBN 978-1-72817-395-5. Available from: https://ieeexplore.ieee.org/document/9197316/, doi:10.1109/ICRA40945.2020.9197316. 29

[127] TRIGGS, B., MCLAUCHLAN, P. F., HARTLEY, R. I., AND FITZGIBBON, A. W. Bundle adjustment - a modern synthesis. In *International Workshop on Vision Algorithms*, pp. 298–372. Springer (2000). 39

[128] VELAS, M., SPANEL, M., AND HEROUT, A. Collar line segments for fast odometry estimation from velodyne point clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 4486–4495 (2016). 41

[129] VESPA, E., FUNK, N., KELLY, P. H., AND LEUTENEGGER, S. Adaptive-resolution octree-based volumetric slam. In *2019 International Conference on 3D Vision (3DV)*, pp. 654–662. IEEE (2019). 89

[130] VESPA, E., NIKOLOV, N., GRIMM, M., NARDI, L., KELLY, P. H. J., AND LEUTENEGGER, S. Efficient Octree-Based Volumetric SLAM Supporting Signed-Distance and Occupancy Mapping. *IEEE Robotics and Automation Letters*, **3** (2018), 1144. doi:10.1109/LRA.2018.2792537. 92

[131] VIZZO, I., CHEN, X., CHEBROLU, N., BEHLEY, J., AND STACHNISS, C. Poisson Surface Reconstruction for LiDAR Odometry and Mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 5624–5630 (2021). doi:10.1109/ICRA48506.2021.9562069. xiv, 82, 83, 84

[132] VIZZO, I., GUADAGNINO, T., BEHLEY, J., AND STACHNISS, C. VDBFusion: Flexible and Efficient TSDF Integration of Range Sensor Data. *Sensors*, **22** (2022), 1296. doi:10.3390/s22031296. 82, 87, 89, 95

[133] VIZZO, I., GUADAGNINO, T., MERSCH, B., WIESMANN, L., BEHLEY, J., AND STACHNISS, C. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *ral*, **8** (2023), 1029. doi:10.1109/LRA.2023.3236571. 73

[134] WANG, G., WU, X., LIU, Z., AND WANG, H. PWCLO-Net: Deep LiDAR Odometry in 3D Point Clouds Using Hierarchical Embedding Mask Optimization. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 15905–15914 (2021). doi:10.1109/CVPR46437.2021.01565. 77

[135] WANG, H., WANG, C., CHEN, C.-L., AND XIE, L. F-loam : Fast lidar odometry and mapping. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4390–4396 (2021). 73

[136] WANG, P., LIU, L., LIU, Y., THEOBALT, C., KOMURA, T., AND WANG, W. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, (2021). 20

[137] WANG, Z., ZHANG, L., SHEN, Y., AND ZHOU, Y. D-liom: Tightly-coupled direct lidar-inertial odometry and mapping. *IEEE Transactions on Multimedia*, (2022), 1. doi:10.1109/TMM.2022.3168423. 55

[138] WEI, J. AND LEUTENEGGER, S. GSFusion: Online RGB-D Mapping Where Gaussian Splatting Meets TSDF Fusion. *IEEE Robotics and Automation Letters*, **9** (2024), 11865. doi:10.1109/LRA.2024.3502065. 94, 95, 97

[139] WELFORD, B. P. Note on a Method for Calculating Corrected Sums of Squares and Products. *Technometrics*, **4** (1962), 419. doi:10.1080/00401706.1962.10490022. 93

[140] WHELAN, T., LEUTENEGGER, S., SALAS-MORENO, R., GLOCKER, B., AND DAVISON, A. Elasticfusion: Dense slam without a pose graph. In *rss* (2015). 46

[141] WU, C., DUAN, Y., ZHANG, X., SHENG, Y., JI, J., AND ZHANG, Y. MM-Gaussian: 3D Gaussian-based Multi-modal Fusion for Localization and Reconstruction in Unbounded Scenes. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 12287–12293 (2024). doi:10.1109/IROS58592.2024.10801605. 76, 77

[142] WU, G., YI, T., FANG, J., XIE, L., ZHANG, X., WEI, W., LIU, W., TIAN, Q., AND WANG, X. 4d gaussian splatting for real-time dynamic scene rendering. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 20310–20320 (2024). 103

[143] WU, S., BASU, S., BROEDERMANN, T., VAN GOOL, L., AND SAKARIDIS, C. PBR-NeRF: Inverse rendering with physics-based neural fields. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2025). 20

[144] WU, S., LV, Z., ZHU, Y., FROST, D., LI, Z., YAN, L.-Q., REN, C., NEWCOMBE, R., AND DONG, Z. Monocular online reconstruction with enhanced detail preservation (2025). Available from: https://arxiv.org/abs/2505.07887, arXiv:2505.07887. 102

[145] XU, W. AND ZHANG, F. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robotics and Automation Letters*, **6** (2021), 3317. 55

[146] YAN, Z., SUN, L., KRAJNIK, T., AND RUICHEK, Y. EU long-term dataset with multiple sensors for autonomous driving. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (2020). 63

[147] YE, H., CHEN, Y., AND LIU, M. Tightly coupled 3d lidar inertial odometry and mapping. In *2019 International Conference on Robotics and Automation (ICRA)*, vol. abs/1904.06993, pp. 3144–3150. IEEE (2019). 55

[148] YOON, B.-H., JEONG, H.-W., AND CHOI, K.-S. Targetless multiple camera-LiDAR extrinsic calibration using object pose estimation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 13377–13383. IEEE, IEEE (2021). ISBN 978-1-72819-077-8. Available from: https://ieeexplore.ieee.org/document/9560936/, doi:10.1109/ICRA48506.2021.9560936. 28

[149] YU, M., LU, T., XU, L., JIANG, L., XIANGLI, Y., AND DAI, B. Gsdf: 3dgs meets sdf for improved rendering and reconstruction. *arXiv preprint*, (2024). 87

[150] YU, Z., CHEN, A., HUANG, B., SATTLER, T., AND GEIGER, A. Mip-splatting: Alias-free 3d gaussian splatting. *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, (2024). 102

[151] YUGAY, V., LI, Y., GEVERS, T., AND OSWALD, M. R. Gaussian-SLAM: Photo-realistic Dense SLAM with Gaussian Splatting (2024). arXiv:2312.10070, doi:10.48550/arXiv.2312.10070. 76, 77, 79, 80, 81

[152] ZHANG, G., SANDSTRÖM, E., ZHANG, Y., PATEL, M., VAN GOOL, L., AND OSWALD, M. R. Glorie-slam: Globally optimized rgb-only implicit encoding point cloud slam. *arXiv preprint arXiv:2403.19549*, (2024). 77

[153] ZHANG, J., KAESS, M., AND SINGH, S. On degeneracy of optimization-based state estimation problems. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 809–816 (2016). 42

[154] ZHANG, J. AND SINGH, S. LOAM: Lidar Odometry and Mapping in Real-time. In *Robotics: Science and Systems X* (edited by D. Fox, L. E. Kavraki, and H. Kurniawati). Robotics: Science and Systems Foundation (2014). ISBN 978-0-9923747-0-9. Available from: http://www.roboticsproceedings.org/rss10/p07.html, doi:10.15607/RSS.2014.X.007. 41, 54, 76

[155] ZHANG, J. AND SINGH, S. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 2174–2181 (2015). 41

[156] ZHANG, L., CAMURRI, M., AND FALLON, M. Multi-Camera LiDAR Inertial Extension to the Newer College Dataset. *arXiv preprint*, (2021). arXiv:2112.08854. xiv, xv, xviii, 47, 82, 83, 86, 95, 97, 100

[157] ZHANG, L., HELMBERGER, M., FU, L. F. T., WISTH, D., CAMURRI, M., SCARAMUZZA, D., AND FALLON, M. Hilti-oxford dataset: A millimeter-accurate benchmark for simultaneous localization and mapping. *ral*, **8** (2022), 408. 63, 65, 67, 68

[158] ZHENG, J., BARATH, D., POLLEFEYS, M., AND ARMENI, I. MAP-ADAPT: Real-Time Quality-Adaptive Semantic 3D Maps. In *Computer Vision – ECCV 2024* (edited by A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol), pp. 220–237. Springer Nature Switzerland, Cham (2024). ISBN 978-3-031-72933-1. doi:10.1007/978-3-031-72933-1_13. 89, 93

[159] ZHENG, X. AND ZHU, J. Efficient LiDAR Odometry for Autonomous Driving. *ral*, **6** (2021), 8458. doi:10.1109/LRA.2021.3110372. 77

[160] ZHONG, X., PAN, Y., BEHLEY, J., AND STACHNISS, C. SHINE-Mapping: Large-Scale 3D Mapping Using Sparse Hierarchical Implicit Neural Representations. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 8371–8377 (2023). doi:10.1109/ICRA48891.2023.10160907. 76, 77

[161] ZHOU, L., LI, Z., AND KAESS, M. Automatic extrinsic calibration of a camera and a 3d LiDAR using line and plane correspondences. pp. 5562–5569. IEEE (2018). ISBN 978-1-5386-8094-0. Available from: https://ieeexplore.ieee.org/document/8593660/, doi:10.1109/IROS.2018.8593660. 29

[162] ZHU, L., LI, Y., SANDSTRÖM, E., HUANG, S., SCHINDLER, K., AND ARMENI, I. LoopSplat: Loop Closure by Registering 3D Gaussian Splats. In *Proc. of the International Conference on 3D Vision (3DV)* (2025). 76, 79, 80, 81

[163] ZHU, Z., PENG, S., LARSSON, V., CUI, Z., OSWALD, M. R., GEIGER, A., AND POLLEFEYS, M. NICER-SLAM: Neural Implicit Scene Encoding for RGB SLAM. In *Proc. of the International Conference on 3D Vision (3DV)*, pp. 42–52. IEEE (2024). doi:10.1109/3DV62453.2024.00096. 89

[164] ZHU, Z., PENG, S., LARSSON, V., XU, W., BAO, H., CUI, Z., OSWALD, M. R., AND POLLEFEYS, M. NICE-SLAM: Neural Implicit Scalable Encoding for SLAM. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 12776–12786 (2022). doi:10.1109/CVPR52688.2022.01245. 76, 77