



SAPIENZA  
UNIVERSITÀ DI ROMA

DOCTORAL THESIS

---

**Unified Multimodal 3D Reconstructions:  
Leveraging Parallels Between Camera and  
LiDAR**

---

*Author:*  
Luca Di Giammarino

*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)

December 2023

Thesis defended the 16th of January 2024.

---

**Unified Multimodal 3D Reconstructions: Leveraging Parallels Between Camera and LiDAR**

Ph.D. Thesis - Sapienza University of Rome

**Advisor:** Prof. Giorgio Grisetti

**Co-Advisor:** Prof. Francesco Leotta

**Reviewer:** Prof. Cyrill Stachniss

**Reviewer:** Prof. François Goulette

© 2023 Luca Di Giammarino. All rights reserved.

Author's email: [digiammarino@diag.uniroma1.it](mailto:digiammarino@diag.uniroma1.it)



# Abstract

In our modern digital era, taking photos with smartphones is now a common habit. However, these 2D photos show only a part of the whole scene. 3D reconstruction, on the other hand, provides a comprehensive, multi-faceted view of scenes, enriching experiences from personal memories to professional fields like urban planning and archaeology. This technique is fundamental in augmented reality and robotic navigation, where a deep grasp of the world's 3D structure is essential.

Current advancements, while significant, have yet to realize an “ideal” 3D reconstruction system. Such a system would consistently capture nearly all visible surfaces, being highly robust in camera tracking, and deliver intricate reconstructions rapidly. It would seamlessly scale from small to large environments without losing accuracy and perform optimally across diverse environmental and lighting conditions.

Beyond classic cameras, LiDAR has gained significant attention over the past decade as another tool to perceive and reconstruct our surroundings. While cameras provide visually rich data, their depth perception is often limited (hard to estimate), especially in low-light conditions. Contrastingly, LiDARs, relying on their emitted light, excel in various lighting scenarios and larger environments, but their measurements are sparse and lack colors. Hence, to achieve impressive results, an “ideal” 3D reconstruction pipeline should rely on both sensors to mitigate their limitations. This research exploits similarities between the two sensors, underscoring the value of integrating uniformly data from both to achieve a more comprehensive environmental understanding, ensuring accurate performance without extensive waiting.

However, integrating camera and LiDAR data presents challenges due to their distinct data natures, necessitating precise calibration, synchronization, and complex multimodal processing pipelines. Advances in technology have facilitated similarities between LiDAR generated images and those from passive sensors, opening avenues for visual place recognition. Our exploration in this domain yielded promising results, particularly highlighting LiDAR's consistent performance in diverse lighting.

Our research journey then transitioned to bridging the gap between LiDAR and RGB-D sensors. By devising a Simultaneous Localization and Mapping (SLAM) pipeline adaptable to both sensors and rooted in photometric alignment, our findings were comparable with specialized systems. Delving into Bundle Adjustment, our generalized strategy showcased remarkable efficiency, especially when merging data from both sensors. Further refinement incorporated geometric information, balancing robustness with precision and achieving impressive accuracy across varied environments.

In addition, we introduce a robotics perception dataset from Rome, encompassing RGB, dense depth, 3D LiDAR point clouds, IMU, and GPS data. Recognizing current dataset limitations and the proficiency of contemporary SLAM and 3D reconstruction methods,

our dataset offers a fresh challenge to push algorithm boundaries. We emphasize precise calibration and synchronization, using modern equipment to capture varied settings from indoor to highways. Collected manually and through vehicles, it is tailored for various robotic uses.

In essence, this thesis encapsulates the pursuit of enhancing SLAM and 3D reconstructions through multimodality. By harnessing the capabilities of diverse depth sensors, we have made significant progress in the domain, paving the way for more integrated, compact, robust and detailed systems in the future.

## Acknowledgments

*Mentre mi appresto a chiudere questo capitolo della mia vita, rifletto su anni che sono stati intensi, impegnativi, ma soprattutto sorprendentemente divertenti.*

*Il primo, caloroso ringraziamento va al prof. Giorgio Grisetti. Grazie per aver aperto per me la porta di questa avventura e per aver creduto nelle mie capacità. La tua disponibilità è stata senza pari, anche nei momenti più improbabili, come quando mi sono trovato con la batteria della macchina scarica!*

*Un ringraziamento speciale ad Ale. La tua presenza, il tuo sostegno e la tua “pazienza” durante questi anni sono stati fondamentali. Senza di te, probabilmente questa tesi non avrebbe visto la luce, e chi sa, forse la comunità scientifica avrebbe perso un contributo... (o forse sarebbe stata grata!)*

*Alla mia famiglia. Grazie per aver sempre assecondato le mie scelte, anche le più audaci, e per averlo fatto con un sorriso e con l’entusiasmo che solo una famiglia può avere.*

*Un caloroso grazie ai miei colleghi, che sono diventati più di semplici collaboratori: amici. Grazie per la vostra professionalità, genialità (a tratti) e per quei momenti di risate e divertimento che hanno reso tutto più leggero, sia durante le lunghe ore in ufficio che nei momenti di relax.*

*A tutti i miei amici, che nonostante tutto, sono sempre rimasti al mio fianco. La vostra amicizia è stata una delle costanti più preziose di questo percorso.*

*Un sentito grazie a Francesco Giardiello e Fabio Cottefoglie, per aver ampliato i miei orizzonti ed avermi dato la possibilità di guardare oltre l’accademia.*

*Un ringraziamento particolare al prof. Cyrill Stachniss, per avermi dato sempre degli spunti interessanti e formativi, durante le nostre collaborazioni.*

*Al prof. Marc Pollefeys e a tutto il team di @cvg, un immenso grazie per avermi accolto nel vostro gruppo all’ETH di Zurigo e per avermi dato l’opportunità di lavorare in un ambiente così stimolante. Scrivendo da qui, non posso fare a meno di pensare a quante altre avventure mi aspettano.*

*E infine, grazie a tutte le persone che mi hanno voluto bene lungo questo percorso e, sì, anche a quelle che forse hanno avuto opinioni diverse. Ogni interazione, ogni critica, ogni sostegno, ha contribuito a farmi essere la persona che sono oggi - purtroppo o per fortuna. Grazie di cuore a tutti voi per aver reso questo viaggio indimenticabile.*



# Contents

<b>Abstract</b>	<b>iii</b>
<b>Glossary</b>	<b>xxi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Overview in 3D reconstruction . . . . .	2
1.3 Thesis Outline . . . . .	4
<b>2 Background</b>	<b>7</b>
2.1 Distinguishing LiDAR and Camera Sensing . . . . .	7
2.2 Sensor Geometry . . . . .	8
2.2.1 Image poses . . . . .	8
2.2.2 Camera sensor . . . . .	9
2.2.3 LiDAR sensor . . . . .	12
2.2.4 Image types . . . . .	16
2.2.5 Image coordinate conventions . . . . .	17
2.2.6 Image interpolation . . . . .	18
2.2.7 Rolling shutter and “skewing” effects . . . . .	19
2.3 Non-linear Continuous Optimization . . . . .	20
2.3.1 Gauss-Newton method . . . . .	21
2.3.2 Levenberg-Marquardt method . . . . .	24
2.3.3 Optimizaton on manifolds . . . . .	24
2.3.4 Factor-graphs . . . . .	26
2.3.5 Efficiency and precision . . . . .	26
<b>3 Visual Place Recognition using LiDAR Intensity Information</b>	<b>29</b>
3.1 Related Work . . . . .	30
3.2 Visual Place Recognition applied to LiDAR Intensity Images . . . . .	32
3.2.1 Correcting LiDAR image for consistent feature extraction . . . . .	32
3.2.2 Feature extraction . . . . .	33
3.2.3 Feature-based VPR . . . . .	33
3.3 Experimental Evaluation . . . . .	34
3.4 Conclusion . . . . .	37

<b>4</b>	<b>MD-SLAM: Multi-cue Direct SLAM</b>	<b>39</b>
4.1	Related Work . . . . .	40
4.2	Multi-cue Direct SLAM . . . . .	42
4.2.1	Input preprocessing . . . . .	43
4.2.2	Photometric cost function . . . . .	44
4.2.3	Tracking . . . . .	44
4.2.4	Loop detection and validation . . . . .	45
4.2.5	Pose-graph optimization . . . . .	45
4.3	Experimental Evaluation . . . . .	46
4.3.1	RGB-D . . . . .	47
4.3.2	3D LiDAR . . . . .	48
4.4	Conclusion . . . . .	50
<b>5</b>	<b>Ca<sup>2</sup>Lib: Simple and Accurate LiDAR-RGB Calibration using a Small Common Marker</b>	<b>51</b>
5.1	Related Work . . . . .	52
5.2	LiDAR-RGB Calibration . . . . .	54
5.3	Experimental Evaluation . . . . .	57
5.3.1	Synthetic case . . . . .	57
5.3.2	Real case . . . . .	57
5.4	Conclusion . . . . .	59
<b>6</b>	<b>Photometric LiDAR and RGB-D Bundle Adjustment</b>	<b>61</b>
6.1	Related Work . . . . .	61
6.2	Photometric Bundle Adjustment . . . . .	64
6.2.1	Graph construction . . . . .	64
6.2.2	Photometric cost function . . . . .	65
6.3	Experimental Evaluation . . . . .	66
6.3.1	RGB-D . . . . .	68
6.3.2	3D LiDAR . . . . .	69
6.3.3	Photometric multimodal: RGB-D + 3D LiDAR . . . . .	71
6.4	Conclusion . . . . .	73
<b>7</b>	<b>VBR : A Vision Benchmark in Rome</b>	<b>75</b>
7.1	Related Work . . . . .	76
7.2	The Datasets . . . . .	79
7.2.1	Sensors setup . . . . .	79
7.2.2	Calibration . . . . .	79
7.2.3	Synchronization . . . . .	80
7.2.4	Ground truth generation . . . . .	81
7.2.5	Data selection . . . . .	82
7.3	Benchmark . . . . .	83
7.3.1	Evaluation . . . . .	85
7.4	Conclusion . . . . .	86

---

<b>8</b>	<b>Multimodal, fast and uniform 3D reconstruction</b>	<b>87</b>
8.1	Related Work . . . . .	87
8.2	Multimodal Bundle Adjustment . . . . .	89
8.2.1	Voxelhashing for camera dense depth generation . . . . .	89
8.2.2	Cost function . . . . .	91
8.3	Experimental Evaluation . . . . .	93
8.3.1	Qualitative results . . . . .	95
8.4	Conclusion . . . . .	96
<b>9</b>	<b>Conclusion</b>	<b>99</b>
9.1	Summary . . . . .	99
9.2	Outlook . . . . .	100
	<b>Appendices</b>	<b>103</b>
<b>A</b>	<b>Appendix</b>	<b>105</b>
A.1	Jacobian derivation of Photometric Bundle Adjustment . . . . .	105





# List of Figures

- 1.1 Generic high-level scheme of typical 3D reconstruction systems for sensors moving in a mainly static scene. First, sensor calibration is performed before the system is started, *extrinsic* calibration may be necessary for systems that works jointly for multiple sensors. *Tracking* (or odometry estimation) and *loop-closures* run in parallel, one estimates the ego-motion of the sensor and the other triggers optimization to reduce overall drift. In robotics and computer vision community this usually called SLAM. This block is usually real-time. For more accurate maps a final refinement is necessary, this can be done offline. By the computer vision community this is usually called Bundle Adjustment (BA). Note that this is just a high-level schema that accounts for most of the content presented in this thesis. This flow may be slightly different in other case (i.e. calibration is performed online during SLAM, BA may be part of the online SLAM estimation). . . . . 3
- 2.1 Sketch of  $\mathbb{SE}(3)$  transformations between a global frame  $W$  and local sensor frame  $S$ . Both of these frames provide different coordinate systems for the same 3D space.  $\mathbf{T}_{W,S}$  expresses  $S$  in  $W$ , hence transform point represented in frame  $S$  into the frame  $W$ .  $\mathbf{T}_{S,W}$  is the opposite. We suggest to read  $\mathbf{T}_{W,S}$  as "*transformation of S in W, sensor-in-world*" to avoid ambiguity. . . . 9
- 2.2 Different depth cameras. From left to right respectively Microsoft Kinect [5], Intel D405 [3] and Intel L515 [2]. Kinect, initially designed for gaming, utilizes both infrared and RGB cameras. It primarily leverages structured light technology, projecting a specific pattern onto a scene measuring the difference of the calibrated pattern to estimate depth. With its baseline of around 7 cm can reconstruct depth til 6/7 meters. Differently D405, does not rely on IR emitters, it is just equipped with two global shutter cameras and with its really small baseline is capable to reconstruct depth at sub-millimeters accuracy at an ideal range of 7 cm (till a maximum one of 50 cm). The one on the right, L515, is a depth camera that relies on ToF technology. As for LiDAR the good thing about active depth sensing is its capability to work in various lighting conditions, including total darkness. Its maximum depth is of 7 meters. . . . . 10

- 2.3 Illustration of pinhole camera model. The pinhole is located at the origin of the local camera coordinate system  $S$ . Note that most of the computer vision books refer with this quantity as  $C$ . However, we denote this with  $S$  to generalize the sensor. For convenience, the image sensor is imagined to be mirrored to the front of the camera (which would mirror the image at the principal point). In the illustration, intersections with the image plane are drawn with crosses, while those with the 3D world are highlighted with full circles. As common in computer vision, the camera reference frame is conventioned with z-axis pointing forward. An additional important aspect is the depth  $d$ , illustrated in green; this represents the distance between the pinhole and the point  $\mathbf{p}$  expressed over the z-axis. . . . . 11
- 2.4 Triangulation sketch.  $u$  and  $u'$  are the distances between points in image plane (from their origins) corresponding to the scene point 3D  $\mathbf{p}$  and their camera center. The baseline is  $b$  and  $f_x$  camera's focal length along  $u - x$  coordinates of the image, should be known quantity coming from intrinsics and extrinsics calibration. So in short, the above equation says that the depth  $d$  of a point in a scene is inversely proportional to the difference in distance of corresponding image points and their camera centers. With this information, once can derive the depth for all pixels in a image. . . . . 12
- 2.5 Different LiDARs available in the market. From left to right respectively Velodyne Puck [7], Ouster OS-128 [6] and Livox MID-70 [4]. Velodyne Puck, often referred to as the VLP-16, is a mechanical LiDAR equipped with 16 laser channels and is commonly used in various applications from drones to robotics due to its balance of performance, size, and price. It provides a 360 deg horizontal FoV and a vertical one of roughly 30 deg. Ouster OS-128 is always a mechanical LiDAR but with higher vertical resolution, as its name implies, it has 128 laser channels, offering a denser point cloud which is crucial for applications that require intricate detail. While the horizontal FoV is always 360 deg due to encoder rotation, its vertical FoV can reach up to 90 deg. It is the one we have used more in our work, due to its higher vertical capabilities. Livox MID-70, on the other hand, is a solid-state LiDAR and differentiates itself with its non-repetitive scanning pattern. Unlike traditional mechanical LiDARs that scan in a set, predictable pattern, Livox employs a unique approach where the scanning pattern is more random, aiming to achieve more uniform point cloud distribution. . . . 13
- 2.6 Illustration of a single beam emitting and receiving light pulse, as described in Eq. (2.9). . . . . 14
- 2.7 Illustration of the spherical model. Unlike the optical frame, the LiDAR reference frame is conventionally set with the z-axis pointing up. In the illustration, intersections with the image plane are drawn with crosses, while those with the 3D world are highlighted with full circles. The spherical coordinates are sketched in red, with  $\theta$  representing the azimuth,  $\phi$  the elevation, and  $\rho$  the range. . . . . 16
- 2.8 Differences between "projection by ID" and Spherical projection. . . . . 17
- 2.9 Image coordinate systems. Example coordinates values for *pixel-corner* in black and *pixel-center* in red, within a  $3 \times 3$  image. . . . . 18

- 2.10 Bilinear interpolation illustrated for a group of adjacent  $2 \times 2$  pixels. The control points  $\mathbf{u}_{00}$  to  $\mathbf{u}_{11}$  are located in the pixel centers. The value is interpolated for the blue cross at relative position  $(u, v)^T$ . Each control point's value is weighted with the area of the rectangle between the control point that is opposite to it and point  $(u, v)^T$ . . . . . 19
- 2.11 A brief overview of the Gauss-Newton algorithm across several iterations: beginning with the state  $\mathbf{x}_0$  (the initial-guess), the cost function (depicted in gray) is approximated using a parabola (in orange). The subsequent state,  $\mathbf{x}_1$ , is identified as the parabola's minimum, at which point a new parabola (shown in red) is employed for further approximation. This process is reiterated, leading to  $\mathbf{x}_2$  and ultimately  $\mathbf{x}_3$ . By this stage, the approximation is closely aligned with the cost function's true minimum. . . . . 23
- 2.12 Illustration of a Manifold space. Since the manifold is smooth, local perturbations - i.e.  $\Delta \mathbf{x}$  in the illustration - can be expressed with a suitable Euclidean vector. Illustration courtesy of Grisetti *et al.* [49]. . . . . 25
- 3.1 Qualitative results. Top: Example pairs of a query and a reference intensity image from our self-recorded dataset (IPB Car). Note that due to its high horizontal resolution (original size:  $64 \times 1024$ ), the intensity image has been divided in two parts, one above the other. The green line illustrate descriptor matches provided by HBST [105] on intensity data. Bottom: Newer College [98] (left) and IPB car (right) datasets used for evaluation, valid loops detected using HBST highlighted in green. . . . . 30
- 3.2 Empty lines removal. From top to bottom, original image after projection from 3D point cloud as explained in Sec. 3.2.1, detection of empty rows highlighted in red, result after image manipulation. Each image shown has been cropped to half of their horizontal size for better viewing. . . . . 33
- 3.3 Precision-Recall curves of the closures computed both with different combinations of feature extractors – image matchers on the LiDAR intensity image. Greater accuracy is reported in general by ORB-HBST, ORB-DBoW2 and Superpoint-DBoW2. Precision-Recall curves have been generated using different percentiles of query closures vector. . . . . 36
- 3.4 Localization timings of different combinations of feature extractors – image matchers. As expected binary descriptors take lower time to extract and match compared to floating points one. . . . . 36
- 3.5 Qualitative comparison between *IPB Car* LiDAR intensity image (up) and KITTI LiDAR intensity image (bottom) (both unprocessed). A lower vertical FoV and the uneven distribution of channels along the spinning axis makes KITTI intensity image unusable for this task. . . . . 37
- 3.6 Max  $F_1$  Score reached in full evaluation mode. Left our self-recorded dataset IPB Car and right Newer College [98]. The error bars indicate the  $F_1$  Score obtained using same approaches but over standard RGB images. Only DBoW2 ORB Voc-Train has not been replicated since the BoW training has been performed across LiDAR intensity images. . . . . 38

4.1	Scenes reconstructed using our pipeline. Top: results of a self-recorded dataset using Intel Realsense 455 RGB-D. Bottom: using LiDAR OS0-128 of the <i>cloister</i> sequence from the Newer College dataset [142]. . . . .	40
4.2	Illustration of our system. Depth $\mathcal{I}_t^d$ and $\mathcal{I}_t^g$ images are taken as input from the pipeline. An optimized trajectory within a map is produced as output. We do not spawn a keyframe $i$ each processed images $t$ (explained in Sec. 4.2.3). This system works independently both for RGB-D and LiDAR. Note that for LiDAR we take range image and not depth image as input. . .	42
4.3	Cues generated for LiDAR (top) and RGB-D (bottom) images. The first row/column shows the intensity $\mathcal{I}^i$ , the middle shows the depth $\mathcal{I}^d$ , and the last one illustrates the normals encoded by color $\mathcal{I}^n$ . The red pixels on the intensity cues are invalid measurements (i.e., depth not available). . . . .	43
4.4	Some scenes from Newer College dataset (LiDAR) - right, and self-recording data (RGB-D) - left, reconstructed with MD-SLAM. . . . .	46
4.5	MD-SLAM map on <i>long</i> sequence from Newer College [98] aligned with Google Earth. . . . .	47
4.6	Alignment of our estimate with the ground truth in <i>long</i> sequence of Newer College. The color bar on the right shows the translational error [m] over the whole trajectory. . . . .	49
5.1	Camera/LiDAR qualitative calibration results. The image shows the reprojection of a LiDAR point cloud on a image captured by a fisheye RGB camera rigidly attached to the former. The offset between the sensors leads to shadows on parts of the image. . . . .	52
5.2	LiDAR image generation for calibration. (a) shows a comparison between the spherical projection (top) used for LiDAR images and the projection by ID (bottom) used for this work (as already shown in Fig. 2.8). (b) shows the ring information before (left) and after (right) the projection. . . . .	54
5.3	Qualitative samples showing LiDAR cloud projection on RGB image. . . . .	55
5.4	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. . . . .	56
5.5	Acquisition system used for the real case experiments. . . . .	58
5.6	Average calibration errors evaluated on separate profiles using real-data. The plots shows how increasing the number of measurements lead to an enhancement in accuracy within our calibration approach. . . . .	59
6.1	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 [33] 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. . . . .	62

6.2	The flow of our approach. The input of our system contains the initial guess of the sensor pose, the intensity image $\mathcal{I}^s$ and the depth image $\mathcal{I}^d$ . 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. 6.2.1) and running a global optimization on the set of poses $\mathbf{X}_{1:N}$ , as detailed in Sec. 6.2.2. . . . .	64
6.3	An example of pose-pairs associated, 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. 4.2.1, i.e., images of different resolutions). Here, we show just one level for simplicity. The data used is from ETH3D. . . . .	65
6.4	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. The heatmap is normalized between 0 and 0.005 [m] for better visualization. The data is self-recorded using an Intel D455. . . . .	67
6.5	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. . . . .	68
6.6	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. . . . .	69
6.7	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. . . . .	72
6.8	Plots of trajectory estimates in different LiDAR sequences of Newer College [142]. . . . .	74
7.1	A summary of our dataset. Data illustrating some of the sequences recorded (top). 3D mapping done with of our ground truth (bottom). . . . .	76
7.2	Comparison between LiDAR clouds attached to ground truth trajectories of KITTI (up) and ours (down). The zoom shows the elevation view. . . . .	77
7.3	Projection of the KITTI LiDAR point cloud into an image plane (up), projection of our LiDAR into an image plane (down). The many holes of the up image due to uneven distribution of the LiDAR beams and calibration issues make the KITTI LiDAR image unusable for computer vision tasks. . . . .	77

7.4	Sensor setup and reference frames. Our ground truth is expressed in the LiDAR reference frame $RF_L$ . More details can be found in our website and supplementary materials. . . . .	78
7.5	Number of top 20 most frequent semantic instance for Ciampino (above) and Colosseum (below) sequences. The instances were counted using OneFormer [?] over a subset of images for each sequence and excluding the most predominant classes: sky, wall, road, grass, sidewalk, ground. . . . .	82
7.6	Our benchmark. Cumulative RPE [%] and ATE RMSE [m] across all training sequences in our dataset for KISS-ICP, F-LOAM and ORB-SLAM3. . . . .	85
7.7	The image shows the overlay of the 3D model obtained from our ground truth system and a view from Google Maps. . . . .	86
8.1	Streaming Scenario: On the left, Fig. 8.1a illustrates the rendering process over the integration area. The entire upper block is integrated, available in RAM for access, and ready for streaming to the GPU during rendering. The rendering block is processed in the middle of the integrated one, where model is denser. This procedure is repeated until all data has been processed and camera depths for all frames are rendered. On the right, as depicted in Fig. 8.1b, we demonstrate the RAM-GPU streaming process. In this situation, our LiDAR is moving from left-bottom to right-up. The magenta voxels (small squares) are streamed out to RAM since they fall outside the LiDAR's maximum radius, while the purple world chunks (big squares) are about to be streamed into the GPU because they are entering the LiDAR's maximum radius. While this happens in 3D space, we draw it in 2D for clearness. . . . .	90
8.2	The flow of our multimodal approach. The system's input comprises the initial estimate of the sensor pose, the intensity/grayscale image $\mathcal{I}^g$ , and the depth image $\mathcal{I}^d$ . The essence of our refinement strategy, highlighted in green in this sketch, involves creating a graph with geometric and photometric observations (Sec. 6.2.1) and executing a global optimization on the set of poses $\mathbf{X}_{1:N}$ , as elucidated in Sec. 8.2.2. It is worth noting that the input can originate from either RGB-D, LiDAR, or both sensors. In multimodal optimization, our findings suggest using LiDAR for geometric minimization and the camera for photometric error reduction. . . . .	91
8.3	Before and after global geometric optimization using LiDAR. . . . .	92
8.4	Qualitative evaluation of multimodal BA in one of our <i>Campus</i> sequences. The reconstruction has been done using LiDAR within geometric optimization (see Fig. 8.3b) and cameras for photometric, generating dense depth images from LiDAR model through voxelhashing. The sequence is about 1.5 Km long and only 791 RGB images have been used to render the reconstruction. . . . .	93
8.5	Qualitative samples of depth renderings. The samples have been captured from <i>Campus</i> sequence (motion by car) and rendered using technique discussed in Sec. 8.2.1. The depth is thresholded at 50 meters. . . . .	94
8.6	Voxelhashing runtimes. From left to right respectively integration, streaming and rendering runtimes. The $y$ -axis reports time in ms, the $x$ -axis the number of hash buckets involved in the process. . . . .	94

- 
- 8.7 Qualitative COLMAP results in one of the *Campus* sequence. It consistently fails, regardless of the number of images or the quality of matches. . . . . 96
- 9.1 Distorted images and maps. On the left, the impact of an image taken with a rolling shutter camera (at the bottom) is contrasted with one taken using a global shutter. To the right, the distinction between a “skewed” point cloud and its rectified counterpart is displayed, presented in 2D for clarity. Image acknowledgments go to [1] and [102], respectively. . . . . 100





# List of Tables

3.1	Configuration of keypoints detector and descriptors extractors used. . . . .	34
3.2	Datasets we used for evaluation. OD abbreviate outdoor-dynamic, Ext. Loc. stands for External Localization System. . . . .	35
3.3	Max $F_1$ score reached in full validation over the four datasets with combination of HBST [105] and ORB [101]. . . . .	37
4.1	MD-SLAM runtimes expressed in Hz. . . . .	46
4.2	ATE RMSE [m] results on TUM RGB-D [121] datasets. This was recorded with non-synchronous depth using a rolling shutter camera. . . . .	48
4.3	ATE RMSE [m] on ETH3D [108]. This was recorded with global shutter camera and synchronous streams. ElasticFusion fails in <i>table3</i> and <i>table7</i> . . . . .	48
4.4	ATE RMSE [m] results of all benchmarked approaches on the Newer College datasets [98, 142]. SuMA fails on <i>long</i> sequence. . . . .	49
5.1	Average translation error in millimeters with different noise levels and number of measurements (N). . . . .	57
5.2	Quantitative results on synthetic data achieved through calibration using $N = 3$ measurements. We choose this measurement count for parity with the methodology proposed in [14]. In [14], a single measurement is deemed sufficient for calibration determination, with 3 measurements considered the optimal scenario. Beyond 3 measurements accuracy does not improve. Both for our study and in alignment with the findings in [67], 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 [14] in rotation, while outperforming all methods on translation, using small commercial tags. . . . .	58
6.1	ATE RMSE [m] on ETH3D benchmark [108], recorded with global shutter camera and synchronous streams. ElasticFusion fails in <i>table3</i> and <i>table7</i> , BundleFusion fails in <i>table4</i> . . . . .	70
6.2	ATE RMSE [m] on Newer College dataset [142], recorded with OS0-128. We always improve the SLAM baseline, a part for <i>stairs</i> starting from LeGO-LOAM estimate. . . . .	71

7.1	The table summarizes the most important 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. . . . .	78
7.2	Summary of the devices in our sensor setup, and the accuracy of the temporal synchronization. . . . .	79
7.3	Summary of our sequences. . . . .	84
8.1	Multimodal 3D reconstruction runtimes compared to COLMAP (in minutes). On the left are our runtimes, where Bundle Adjustment refers to the combined times of geometric (first) and photometric (second) minimization, as detailed in Sec. 8.2.2. On the right, we have the runtimes for COLMAP. It is worth noting that 3D reconstruction relying solely on a camera is inherently more complex and time-consuming (i.e. structure optimization). The purpose here is not a direct comparison but rather to provide a general sense of how our multimodal approach stacks up against traditional vision techniques. COLMAP optimization with 791 images fails and results are reported in Fig. 8.7a. COLMAP optimization with 6807 images fails again and results are reported in Fig. 8.7b . . . . .	95

# Glossary

**AR** Augmented Reality

**ATE** Absolute Trajectory Error

**BA** Bundle Adjustment

**BMD** Basic Minimum Degree

**BST** Binary Search Tree

**CPU** Central Processing Unit

**EKF** Extended Kalman Filter

**GN** Gauss-Newton

**HBST** Hamming Distance Embedding Binary Search Tree

**ICP** Iterative Closest Point

**ILS** Iterative Least-Squares

**IMU** Inertial Measurement Unit

**IRLS** Iterative Reweighed Least-Squares

**KF** Kalman Filter

**LM** Levenberg-Marquardt

**LS** Least-Squares

**MAP** Maximum-A-Posteriori

**NDT** Normal Distributed Transform

**NeRFs** Neural Radiance Fields

**PCA** Principal Component Analysis

- PGO** Pose-Graph Optimization
- PnP** Perspective-n-Point
- PMF** Probability Mass Function
- PPS** Pulse Per Second
- PR** Place Recognition
- RAM** Random Access Memory
- RPE** Relative Pose Error
- SfM** Structure from Motion
- SGD** Stochastic Gradient Descent
- SLAM** Simultaneous Localization and Mapping
- TSDF** Truncated Signed Distance Function
- UAV** Unmanned Aerial Vehicle
- UGV** Unmanned Ground Vehicle
- UKF** Unscented Kalman Filter
- UMV** Unmanned Maritime Vehicle
- VIO** Visual-Inertial Odometry
- VO** Visual Odometry
- VPR** Visual Place Recognition

# Chapter 1

## Introduction

### 1.1 Motivation

Capturing moments with photos, especially through smartphones, is a daily routine for many. However, these photos offer just a fixed perspective of a scene, missing the depth and other viewpoints. This has sparked interest in 3D reconstruction, which provides a fuller view from various angles, potentially offering a richer way to remember moments than traditional photos.

3D reconstruction has broader applications than just enhancing memories. In Augmented Reality (AR), understanding the real-world's shape and layout is essential to integrate virtual objects seamlessly. Robots and autonomous vehicles need a detailed understanding of their environment for safe navigation. In fields like urban planning and archaeology, a comprehensive 3D view is invaluable, aiding in designing spaces and documenting sites. While cameras offer rich visual data, obtaining depth and distance can be challenging, especially in poor-light conditions.

In the evolving world of SLAM and 3D reconstructions, the integration of data from multiple sensors offers a more comprehensive understanding of environments. Different sensors bring their unique strengths to the table. For instance, while cameras are readily accessible and can capture rich visual details, they often struggle to reconstruct large-scale environments with high accuracy. In addition, they depend on ambient light, making them less effective in dark or featureless areas. On the other hand, LiDARs which base the distance measurements on their own light, can operate in varied lighting conditions and bigger environments. They can provide detailed spatial reconstructions but might result in sparser outputs and often lack the vibrant color information inherent to camera captures. In addition, different sensors can have different fields-of-view. By combining them, one can achieve a more comprehensive coverage of the environment and robustness in more challenging environments (i.e. dynamics in the scene) thanks to bigger static support. Overall, a multi-sensor system can be more versatile and adaptable to different applications and needs.

Nevertheless, fusing data from cameras and LiDARs is a non-trivial task, primarily due to the inherent differences in the nature of data each sensor provides. Cameras capture rich, high-resolution color information in 2D, while LiDARs provide sparse but accurate depth measurements in 3D. Aligning these heterogeneous data sources requires careful calibration and synchronization. Moreover, the data from each sensor comes with its own set

of noise characteristics and uncertainties. For instance, cameras might be affected by lighting conditions, motion blur, or lens distortions, while LiDARs can be influenced by reflectivity variations and distortion effects (skewing). Developing algorithms that can effectively combine these data sources often leads to complex multimodal processing pipelines. Some approaches might treat each sensor's data independently during initial stages of processing and then attempt to merge the results in later stages. This can introduce additional challenges, especially when trying to reconcile conflicts or inconsistencies between the two data types.

Thanks to recent technology enhancements, the images generated from LiDAR point cloud projections have started to show resemblances to those captured by passive sensors. This similarity led to our initial exploration into the use of LiDAR images for Visual Place Recognition (VPR). By applying well-known VPR techniques to LiDAR images, we found encouraging results. While the outcomes were slightly below those achieved with traditional images, the advantage of LiDAR's consistent performance, regardless of lighting conditions, was evident.

Building on this, our subsequent work aimed to create a bridge between LiDAR and RGB-D sensors. We developed a SLAM pipeline that worked in exactly the same way for both sensors (a projection function is the only difference). Relying on photometric odometry estimation and feature-based loop closures from the previous work, our system showed results that were in line with some of the specialized SLAM systems designed for either LiDAR or RGB-D but without an explicit data association and a representation of the model.

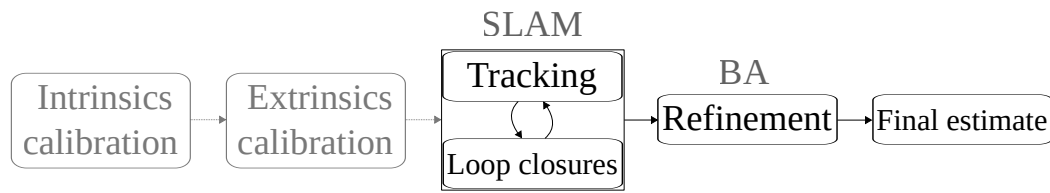
Our next step was to explore the realm of Bundle Adjustment. As for SLAM, we put forth a generalized strategy that was not tied to any specific sensor. Our approach, rooted in photometric alignment, showed results that were comparable or better compared to some existing Bundle Adjustment techniques built ad-hoc for RGB-D and LiDARs. Moreover, when we combined both LiDAR and RGB-D within our unified methodology, we noticed big improvements in the performance of our algorithm.

In our more recent work, we aimed to enhance our Bundle Adjustment methodology by incorporating geometric information. Starting with geometry alignment and then refining through photometric optimization, we developed a system that balances robustness with precision. The geometric part and the explicit point-to-plane data association, added stability to our approach, while the photometric refinement brought in finer detail. Notably, our method scaled well to larger environments, a benefit derived from the sparse geometrical information. Through extensive validation using a total-station, we found our approach to achieve a commendable accuracy of 3 centimeters across 1.5 km trajectories, in challenging environments (covering both indoor and outdoor settings, horizontal and vertical motion and different light settings).

To conclude, my research path has been motivated by the aspiration to understand and refine the functionalities in SLAM and 3D reconstructions using multimodality, while maintaining consistency. By tapping into the potential of various depth sensors, we have made some progress in the field, contributing to move slightly closer to more cohesive, stable and detailed systems in the future.

## 1.2 Overview in 3D reconstruction

This section provides a simplified overview of the typical steps involved in a 3D reconstruction pipeline, setting the stage for the contributions of this thesis. We will focus on



**Figure 1.1.** Generic high-level scheme of typical 3D reconstruction systems for sensors moving in a mainly static scene. First, sensor calibration is performed before the system is started, *extrinsic* calibration may be necessary for systems that works jointly for multiple sensors. *Tracking* (or odometry estimation) and *loop-closures* run in parallel, one estimates the ego-motion of the sensor and the other triggers optimization to reduce overall drift. In robotics and computer vision community this usually called SLAM. This block is usually real-time. For more accurate maps a final refinement is necessary, this can be done offline. By the computer vision community this is usually called BA. Note that this is just a high-level schema that accounts for most of the content presented in this thesis. This flow may be slightly different in other case (i.e. calibration is performed online during SLAM, BA may be part of the online SLAM estimation).

reconstructions derived from sequential data (i.e. images from video stream for cameras) and scenes captured by a moving sensor, as opposed to unordered data. A figure summarize this section is Fig. 1.1.

At the outset, any geometric vision pipeline requires *intrinsics* sensor calibration. For cameras this involves geometric calibration, which associates each camera image pixel with its corresponding observation ray, for LiDARs this involves estimating accurately vertical and horizontal field-of-view. In addition, for algorithms that require both the sensors an accurate estimate of the offset between the two sensors need to be computed (*extrinsics* calibration).

Once calibrated, the system begins to execute multiple tasks concurrently. The primary objective involves determining the sensor’s ego-motion through odometry or tracking. This facilitates the integration of data from various images, which can be achieved by utilizing a scene model or referencing anchor frames. As the system operates, errors might build up and, when revisiting a previously observed part of the scene, a distinct process, termed *loop-closure*, might be needed for corrections. These steps are done simultaneously, for this reason the name SLAM.

Odometry estimation methods can be *direct* or *indirect*. Direct methods align an image with other images or with a model by comparing pixel values, leveraging raw sensor data to average out noise and the whole set of pixels for better and more uniform estimate. However, this method requires a good initial pose estimate. This may for example be obtained by inertial measurements, by a motion model for frames in a video, with a hierarchical approach to incrementally increase the converge basin or with an indirect method. Indirect methods, on the other hand, match some extracted features between measurements or with a model. Usually features are marked with a unique identifier which encodes the local appearance (or local geometry).

If a high-quality estimate is desired as the final output, additional steps may be required which are usually done offline. This might involve motion refinement, structure refinement and meshing.

In summary, this section has outlined the foundational steps in a 3D reconstruction pipeline. The state-of-the art in related work will be discussed within the individual chapters that follow.

### 1.3 Thesis Outline

**Background.** Chapter 2 delves into the foundational concepts crucial for grasping the contributions of this thesis. Specifically, it explores image poses within the  $\mathbb{SE}(3)$  transformations, elucidates the elementary pinhole camera model and spherical model, used respectively for RGB and LiDAR and discuss image interpolation. Furthermore, this section highlights the powerful Gauss-Newton and Levenberg-Marquardt optimization techniques, touches upon optimization within manifolds, and examines strategies to enhance the runtime efficiency of these methods while maintaining accuracy.

**Visual Place Recognition on LiDAR data.** Chapter 3 examines the efficacy of established VPR methods when adapted to intensity data from a 3D LiDAR, transformed into an image format. Our assessments of various VPR processes across multiple robotic/vision datasets using 3D LiDARs indicate that these adapted techniques can provide dependable loop-detection. This suggests the viability of deploying LiDAR-only SLAM on a broader scale. While VPR methods tailored to LiDAR data might underperform slightly compared to conventional RGB images, they offer the benefit of being unaffected by external lighting conditions. This characteristic enables consistent SLAM performance across varying times, such as day or night, without necessitating complex algorithms in the RGB image domain.

**Uniform SLAM.** In Chapter 4, we introduce a novel direct SLAM system designed for both LiDAR and RGB-D. While it is typical in SLAM to have systems tailored for individual sensors, with separate communities addressing camera and LiDAR challenges, our approach bridges this divide. Our system integrates position tracking with an appearance-centric relocalization technique, effectively managing extensive loop closures. Notably, our work stands out as a pioneering effort to create a more universal approach to depth sensors, leveraging the LiDAR image for both photometric and feature-centric optimizations. Through different experiments on various public benchmarks, MD-SLAM has proven its mettle, showing results comparable with top-tier, sensor-specific systems. To benefit the community we release the system as open source at <https://github.com/rvp-group/mdslam>.

**LiDAR-RGB Calibration Using a Common Marker.** In Chapter 5, we introduce an extrinsic calibration method aimed at precisely identifying the offset between LiDAR and camera. Proper calibration is crucial when merging data from these two sensors. While the robotics domain frequently adopts distinct, large, and often costly tags to counter the sparseness of LiDAR, our approach rely on readily available, smaller markers (e.g., A3 chessboard). Comparative tests indicate that our method performs on par or better compared to several calibration techniques, many of which depend on specialized markers like CNC-printed tags. We release the toolbox as open source at <https://github.com/rvp-group/ca2lib>.

**Uniform Photometric Bundle Adjustment.** In Chapter 6, we introduce a cohesive photometric BA approach that seamlessly integrates with both RGB-D and LiDAR systems. This strategy is designed to enhance trajectories derived from SLAM/GNSS systems, aiming to maximize its photometric consistency across the entire collection of sensor positions. Our method implicitly manages data association and offers robust support for both RGB-D and LiDAR, either individually or jointly. Comparative evaluations reveal the expertise of our optimization framework, which consistently matches or outperforms dedicated methods for the two sensors. Furthermore, with precise calibration between the two sensors in place, our method can be augmented through the combined utilization of both LiDAR and camera. To facilitate broader adoption and research, we've made our CUDA/C++ implementation publicly accessible at <https://github.com/rvp-group/ba-mdslam>.



**Benchmark.** In Chapter 7, we introduce a comprehensive robotics perception research dataset recorded in Rome, which encompasses RGB data, dense depth images, 3D point clouds, IMU, and GPS data. Recognizing the limitations of current datasets, such as being confined to specific environments, having simplistic sequences, and the absence of benchmarks, and given the advancements in the fields of SLAM and 3D reconstruction, our dataset is primarily designed to cater to these domains and further the progress in autonomous robotics research. Our dataset stands as a significant contribution to the available resources, addressing a wide range of challenges from environment diversity and motion dynamics to sensor frequency. We have employed recent equipment and detailed our methods for ensuring accurate intrinsic and extrinsic sensor calibrations while maintaining precise temporal synchronization. Our recordings span a range of environments, including multi-floor buildings, gardens, urban settings, and highways. With a combination of handheld and vehicular data collection techniques, our setup is versatile and can emulate a variety of robotic applications, from quadrupeds and drones to autonomous vehicles. Notably, our dataset includes a precise 6-dof ground truth, which is derived from an innovative method that refines RTK-GPS estimates using LiDAR point clouds through BA. All sequences, divided into training and validation sets, and accessible at <https://rvp-group.net/slam-dataset>.

**Multimodal Uniform Bundle Adjustment.** In the concluding Chapter 8, we emphasized the combined strength of LiDAR and camera sensors in 3D reconstruction. By pairing LiDAR’s depth accuracy with the camera’s visual detail, we simplify the reconstruction process. Our integrated BA approach showcases the benefits of using both sensors. This chapter does not provide an in-depth evaluation, but its aims only to highlight the potential of this combined method in terms of efficiency and robustness. The message is clear: combining LiDAR and camera sensors paves the way for more effective 3D reconstruction techniques.

**Conclusion.** The final part of thesis (Chapter 9), sums up the overall contributions and discusses opportunities for future works and improvements.

**Publications.** The content of this thesis is based on publications [32, 33, 46, 34, 16].



## Chapter 2

# Background

In this section, we will explore several fundamental concepts that play a crucial role in comprehending the key contributions of this thesis. Sec. 2.1 briefly illustrates the main difference in sensing between camera and LiDAR. Successively, Sec. 2.2 will provide an introduction to camera and LiDAR geometry, encompassing topics such as 3D rigid body transformations, different projection models used (i.e. pinhole, spherical) and image interpolation. Following that, Sec. 2.3 will delve into the utilization of non-linear optimization techniques, specifically the Gauss-Newton and Levenberg-Marquardt methods.

**Notation.** We represent scalars using italic lowercase letters (e.g.,  $i$ ,  $c$ ), vectors using bold lowercase letters (e.g.,  $\mathbf{v}$ ,  $\mathbf{t}$ ), and matrices using bold uppercase letters (e.g.,  $\mathbf{X}$ ,  $\mathbf{T}$ ). The identity matrix is denoted as  $\mathbf{I}$ . Occasionally, ordinary multiplications, whether scalar or matrix, will be indicated with a central dot (e.g.,  $\mathbf{T} \cdot \mathbf{p}$ ). Please note that this notation is not intended to signify a dot product; dot products will always be expressed through transposition and matrix multiplication.

### 2.1 Distinguishing LiDAR and Camera Sensing

LiDARs and cameras are both essential tools in the realm of sensing and imaging, but they operate on fundamentally different principles in terms of being *active* or *passive*. LiDAR is an active sensor because it emits its own source of light, typically in the form of laser beams, to probe the environment. It then measures the time it takes for the emitted laser beams to bounce back after reflecting off objects, allowing it to determine distances and create detailed 3D maps of the environment.

On the other hand, a camera is a passive sensor. It does not emit any light of its own. Instead, it captures the ambient light that is either reflected from or emitted by objects in its field of view. The information it gathers is based entirely on the light available in the environment, whether it is sunlight, artificial light, or other sources. So, while LiDAR actively interacts with its surroundings by emitting and then detecting laser light, a camera passively records the light that is already present in the scene.

As we progress through this chapter and delve deeper into the thesis, the distinctions and parallels between these two sensors will be further clarified.

## 2.2 Sensor Geometry

To reconstruct a 3D scene from images captured by an optical sensor (i.e. LiDAR, camera), it is essential to establish a common coordinate system for these images, referred to as the "global" coordinate system. Additionally, it necessitates the capability to establish a connection between 3D points within this global coordinate system and the corresponding (sub)pixels in the images where these points are observed. This section will provide an overview of the fundamental concepts required to accomplish this task.

### 2.2.1 Image poses

The pose of an image refers to the sensor's position and orientation when capturing the image (excluding any camera movement during exposure time for simplicity). This pose can be expressed as the transformation from a stable global 3D coordinate system, denoted as  $W$ , to a sensor-local coordinate system, denoted as  $S$ , or conversely as its inverse. In practical applications, it is crucial to maintain clarity regarding the selected direction to avoid any potential misconceptions.

In any case, since optical sensors function as rigid entities moving through 3D space, this transformation belongs to  $\mathbb{SE}(3)$ , which represents the special Euclidean group in three dimensions—a group characterizing rigid body transformations in space. This space comprises six degrees of freedom: three for translation within 3D space and three for rotation in 3D space. Elements of  $\mathbb{SE}(3)$  can be represented in various ways, such as using quaternions along with translation vectors or as  $4 \times 4$  homogeneous transformation matrices. This section will explore the latter approach, removing the homogeneous row for simplicity, where the left  $3 \times 3$  sub-matrix of the complete  $3 \times 4$  matrix represents a rotation matrix, while the right  $3 \times 1$  sub-matrix represents a translation vector. For a transformation denoted as  $\mathbf{T}$  with rotation  $\mathbf{R}$  and translation  $\mathbf{t}$ , it can be expressed as follows:  $\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \end{pmatrix}$ . When dealing with a 3D point  $\mathbf{p}$  represented as a three-dimensional column vector, the transformation can be applied through homogeneous multiplication. In other words, this involves elevating the point to 4D space before performing the multiplication, achieved by introducing a fourth dimension with a value of 1:

$$\mathbf{T}\mathbf{p} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \end{pmatrix}\mathbf{p} = \mathbf{R}\mathbf{p} + \mathbf{t} \quad (2.1)$$

Because we define the transformation matrix as  $3 \times 4$  in size (as opposed to the frequently used  $4 \times 4$ ), there is no need for any supplementary operation to revert the projection to 3D space afterward.

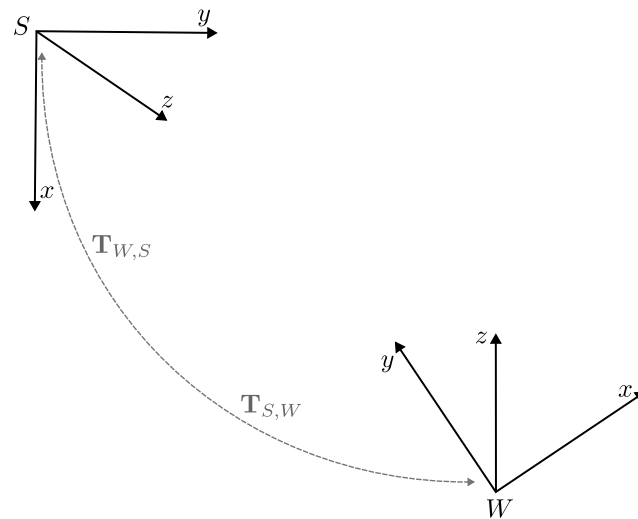
Concatenation of two  $\mathbb{SE}(3)$  transformations produces another transformation in  $\mathbb{SE}(3)$ :

$$\mathbf{T}_1\mathbf{T}_2\mathbf{p} = \mathbf{R}_1(\mathbf{R}_2\mathbf{p} + \mathbf{t}_2) + \mathbf{t}_1 \quad (2.2)$$

The same holds for inversion (note that  $\mathbf{R}^{-1} = \mathbf{R}^T$ ) since rotation matrices are orthogonal):

$$\mathbf{T}_1^{-1}\mathbf{T}_2\mathbf{p} = \mathbf{R}_1^T(\mathbf{R}_2\mathbf{p} + \mathbf{t}_2 - \mathbf{t}_1) \quad (2.3)$$

with  $\mathbf{T}^{-1} = \begin{pmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \end{pmatrix}$ . The following notation will be used for transformations: a transformation  $\mathbf{T}_{W,S}$  expresses the reference frame  $S$  into  $W$  (i.e. points are multiplied to it, from the right, from  $S$  into frame  $W$ ). Therefore, it holds:  $(\mathbf{T}_{W,S})^{-1} = \mathbf{T}_{S,W}$ . Fig. 2.1 illustrates the concept of transforming points between a global and a sensor-local coordinate system.



**Figure 2.1.** Sketch of  $\mathbb{SE}(3)$  transformations between a global frame  $W$  and local sensor frame  $S$ . Both of these frames provide different coordinate systems for the same 3D space.  $\mathbf{T}_{W,S}$  expresses  $S$  in  $W$ , hence transform point represented in frame  $S$  into the frame  $W$ .  $\mathbf{T}_{S,W}$  is the opposite. We suggest to read  $\mathbf{T}_{W,S}$  as "transformation of  $S$  in  $W$ , sensor-in-world" to avoid ambiguity.

### 2.2.2 Camera sensor

Every pixel within a camera captures light entering the camera lens from specific angles or directions. To simplify matters, we consistently assume that these directions can be approximated as a single ray, rather than a more realistic cone-shaped volume. The relationship between pixels and their associated observation rays is defined by a camera model, and the specific calibration parameters are often referred to as the *intrinsic parameters* or simply *intrinsic* of the camera. The most general camera model would allocate a distinct ray (or line, if the starting point of the ray is unknown) for every pixel. While this approach has been proposed [52], it is seldom employed directly in practice due to its complexity in terms of calibration and application.

Consequently, camera models commonly simplify the pixel-ray association by reducing the number of parameters and making certain smoothness assumptions. In fact, many traditional camera models use a minimal set of parameters, attempting to describe the camera's behavior based on fundamental geometric principles and a straightforward lens distortion function. For the purposes of this background section, we focus on the most fundamental of these traditional camera models: the *pinhole camera model*.

#### Pinhole camera model

This model assumes that all incoming light rays, which eventually reach the camera sensor, converge at a singular 3D point, commonly referred to as the *pinhole* or *optical center*. Specifically, this point serves as the focal point through which all light is channeled. In a broader context, it is known as the optical center.

Furthermore, the model assumes that the (planar) image sensor directly faces the *optical center*. In other words, the line extending from the optical center to the closest point on the image sensor plane is perfectly perpendicular to that plane. The precise location on the



**Figure 2.2.** Different depth cameras. From left to right respectively Microsoft Kinect [5], Intel D405 [3] and Intel L515 [2]. Kinect, initially designed for gaming, utilizes both infrared and RGB cameras. It primarily leverages structured light technology, projecting a specific pattern onto a scene measuring the difference of the calibrated pattern to estimate depth. With its baseline of around 7 cm can reconstruct depth til 6/7 meters. Differently D405, does not rely on IR emitters, it is just equipped with two global shutter cameras and with its really small baseline is capable to reconstruct depth at sub-millimeters accuracy at an ideal range of 7 cm (till a maximum one of 50 cm). The one on the right, L515, is a depth camera that relies on ToF technology. As for LiDAR the good thing about active depth sensing is its capability to work in various lighting conditions, including total darkness. Its maximum depth is of 7 meters.

image plane where this line intersects is termed the principal point.

With these assumptions, the camera can be adequately described using only four essential parameters: the  $x$  and  $y$  coordinates of the principal point in the image (denoted as  $c_x$  and  $c_y$ , respectively), and the focal lengths along both the axes, denoted as  $f_x$  and  $f_y$ . The focal lengths represent the distance between the optical center and the image plane. The reason why there might be two different focal lengths,  $f_x$  and  $f_y$ , is because of pixel non-squareness. While most modern cameras have square pixels, meaning  $f_x$  is approximately equal to  $f_y$ , in some cases (especially older cameras or specific imaging devices), the pixels might be rectangular. In this case, the focal lengths in the  $x$  and  $y$  directions would differ. This will be more evident for LiDAR, during spherical projection Sec. 2.2.3. We include these intrinsics parameters into the so-called calibration or camera matrix, which we will refer with  $\mathbf{K}$ . In the following paragraphs, we explore two operations associated with this camera model: *projection* and *inverse projection*.

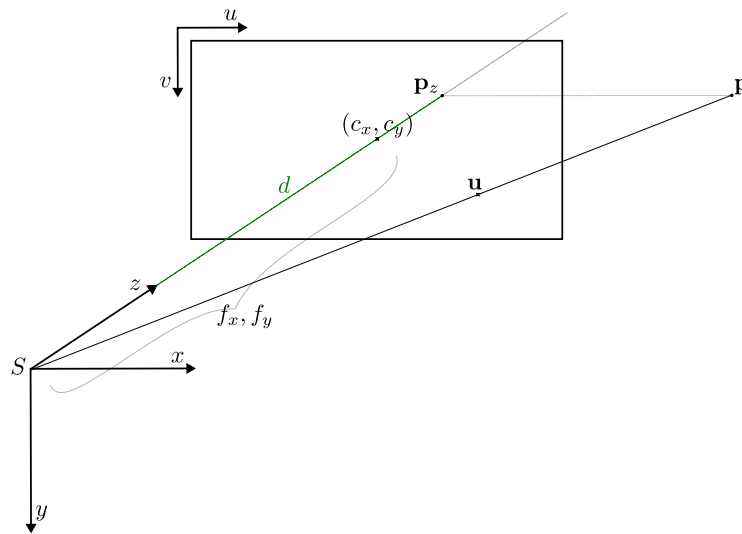
**Projection.** A general projection is a mapping  $\pi : \mathbb{R}^3 \rightarrow \Gamma \subset \mathbb{R}^2$  from a world point  $\mathbf{p} = [x, y, z]^T$  to image coordinates  $\mathbf{u} = [u, v]^T$ . Specifically, the pinhole projection of a point  $\mathbf{p}$  is computed as

$$\pi_p(\mathbf{p}) = \phi(\mathbf{K} \mathbf{p}), \quad (2.4)$$

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad (2.5)$$

$$\phi(\mathbf{v}) = \frac{1}{v_z} \begin{bmatrix} v_x \\ v_y \end{bmatrix}. \quad (2.6)$$

The result  $\pi_p(\mathbf{p})$  is a two-dimensional pixel coordinate. If this coordinate is outside of the image bounds, then the point is not visible in the image. The same applies if  $\mathbf{p}_z \leq 0$ , i.e., the point is not in front of the camera. The intermediate result  $\phi(\mathbf{v})$ , called homogeneous normalization with  $\mathbf{v} = [v_x, v_y, v_z]^T$ , indicates the direction from which the point is observed by the camera.



**Figure 2.3.** Illustration of pinhole camera model. The pinhole is located at the origin of the local camera coordinate system  $S$ . Note that most of the computer vision books refer with this quantity as  $C$ . However, we denote this with  $S$  to generalize the sensor. For convenience, the image sensor is imagined to be mirrored to the front of the camera (which would mirror the image at the principal point). In the illustration, intersections with the image plane are drawn with crosses, while those with the 3D world are highlighted with full circles. As common in computer vision, the camera reference frame is conventioned with  $z$ -axis pointing forward. An additional important aspect is the depth  $d$ , illustrated in green; this represents the distance between the pinhole and the point  $\mathbf{p}$  expressed over the  $z$ -axis.

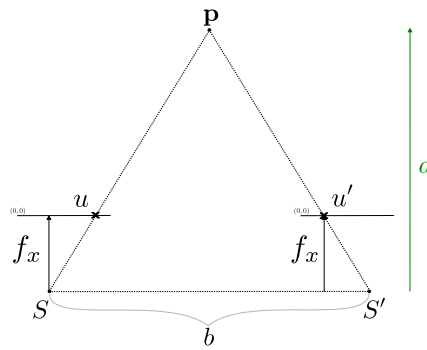
**Inverse projection.** The inverse-projection takes as input a (sub)pixel coordinate within the image and the distance of the observed point from the camera at that particular image position. The inverse projection determines the observation ray corresponding to the given pixel coordinate  $\mathbf{u}$ , and if also a distance is given, the observed 3D point along this ray. Therefore, we can calculate the inverse mapping  $\pi^{-1} : \Gamma \times \mathbb{R} \rightarrow \mathbb{R}^3$ , more explicitly  $\mathbf{p} = \pi^{-1}(\mathbf{u}, d)$ .

$$\mathbf{p} = d \left( \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \right) \quad (2.7)$$

Note that if  $d$  is removed from the above equation (i.e. unknown depth), the result is simply the normalized image coordinates  $\mathbf{v} = [v_x, v_y, 1]^T$  indicating the direction of the observation ray. These may be converted to a direction vector by normalizing the result to unit length.

### Camera depth estimation

Estimating depth using cameras, is a challenging task in computer vision. Several techniques have been developed to estimate depth from camera images []. Currently robust approaches require stereo vision, typically involving two cameras. Stereo vision uses two slightly offset cameras (we refer to *baseline* as the distance between the cameras) to capture images of the same scene. By finding corresponding points in the two images and calculating the disparity



**Figure 2.4.** Triangulation sketch.  $u$  and  $u'$  are the distances between points in image plane (from their origins) corresponding to the scene point 3D  $\mathbf{p}$  and their camera center. The baseline is  $b$  and  $f_x$  camera's focal length along  $u - x$  coordinates of the image, should be known quantity coming from intrinsics and extrinsics calibration. So in short, the above equation says that the depth  $d$  of a point in a scene is inversely proportional to the difference in distance of corresponding image points and their camera centers. With this information, once can derive the depth for all pixels in a image.

(horizontal shift) between them, it is possible to estimate depth using triangulation Eq. (2.8).

$$\text{disparity} = u - u' = \frac{b f_x}{d} \quad (2.8)$$

This technique relies on the principle that objects at different depths will have different disparities. As the baseline between camera increases (so the parallax), far objects distances can be determined. However, this well-known depth estimation technique, may suffer from illumination consistency and texturless surfaces. For this reason, often "depth cameras" [5, 3, 2] are equipped with an infrared (IR) pattern. This has different advantages: it can provide consistent illumination across the scene, reducing the impact of changing lighting conditions that often affect visible light stereo vision; it can enhance the visibility of textureless or low-texture surfaces, making it easier to extract feature points for stereo matching algorithms. These days, depth cameras might use ToF (LiDAR technology). But because they are small and need to be affordable, their ability to measure depth is limited, often to just 5 to 8 meters. Fig. 2.2 shows and details some different depth cameras available in the market at the time of writing.

### 2.2.3 LiDAR sensor

LiDAR technology, which measures distances using pulsed lasers, comes in two primary sensor forms: *mechanical* and *solid-state*.

Mechanical LiDAR systems use rotating parts (encoders) to steer the *laser beams* (for 3D sensing they are usually mounted vertically) granting them an unparalleled 360-degree field of view. This comprehensive scanning capability is a significant advantage, especially in applications requiring a complete environmental perspective, such as 3D reconstructions and autonomous vehicles. However, their moving components can make them bulkier, more prone to wear, and historically more expensive.

In contrast, solid-state LiDAR lacks moving parts. They use electronic methods, such as phased array optics, to steer the laser beam. This result in increased durability and a more





**Figure 2.5.** Different LiDARs available in the market. From left to right respectively Velodyne Puck [7], Ouster OS-128 [6] and Livox MID-70 [4]. Velodyne Puck, often referred to as the VLP-16, is a mechanical LiDAR equipped with 16 laser channels and is commonly used in various applications from drones to robotics due to its balance of performance, size, and price. It provides a 360 deg horizontal FoV and a vertical one of roughly 30 deg. Ouster OS-128 is always a mechanical LiDAR but with higher vertical resolution, as its name implies, it has 128 laser channels, offering a denser point cloud which is crucial for applications that require intricate detail. While the horizontal FoV is always 360 deg due to encoder rotation, its vertical FoV can reach up to 90 deg. It is the one we have used more in our work, due to its higher vertical capabilities. Livox MID-70, on the other hand, is a solid-state LiDAR and differentiates itself with its non-repetitive scanning pattern. Unlike traditional mechanical LiDARs that scan in a set, predictable pattern, Livox employs a unique approach where the scanning pattern is more random, aiming to achieve more uniform point cloud distribution.

compact design. While they are generally more affordable and resilient, their field of view can be limited compared to their mechanical counterparts.

In this thesis, we will focus exclusively on mechanical LiDAR due to its ability to provide a 360-degree scan. This comprehensive scan can be readily converted into an image, thanks to its vertically oriented beams and constant rotation. In contrast, solid-state LiDARs struggle to produce a coherent image from their point cloud, primarily because of the non-redundant steering of the laser beam.

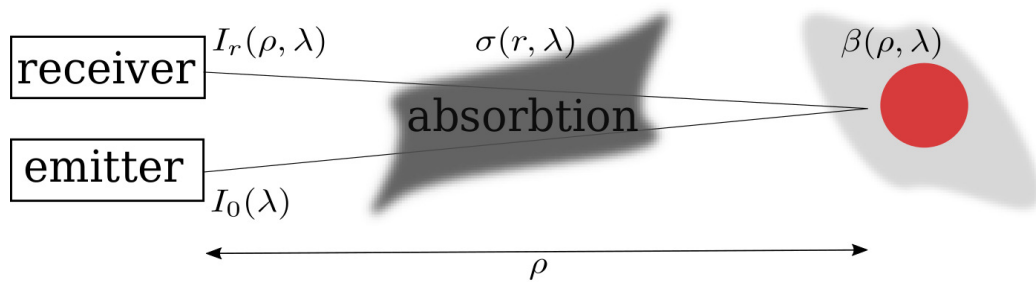
A typical LiDAR sensor emits a beam of pulsed light waves towards the measurement direction. The range to the obstacle along the beam is measured from the light pulse's round trip time. At a low level, a LiDAR senses the perceived light intensity  $I_r(\rho, \lambda)$  as a function of the range of the reflection  $\rho$  and the wavelength  $\lambda$ . The sensed intensity depends on the emitted intensity  $I_0(\lambda)$  at the same wavelength, as follows [99]:

$$I_r(\rho, \lambda) = I_0 \eta \frac{O}{4\pi\rho^2} \beta(\rho, \lambda) \exp\left(-2 \int_0^\rho \sigma(r, \lambda) dr\right) \quad (2.9)$$

Here,  $O$  denotes the beam aperture measured as a solid angle,  $\beta$  is the reflectance of the object, and  $\sigma$  is the absorption of the medium. Fig. 2.6 illustrates this aspect. The reflectivity  $\beta$  is affected by the composition, roughness and moisture content and incidence angle of the beam hitting the surface.

The accuracy of range measurements is primarily determined by the resolution of the clock that measures the signal's first return. However, by assessing the phase difference between the emitted and received signals, we can achieve even greater precision. A single detection might be prone to noise, so for enhanced accuracy, scanners often emit multiple pulses. This approach, however, limits the frequency at which range measurements can be taken.

Mechanical constraints dictate the rotational speed of the sensor. While a 2D sensor can deflect the beam using a small rotating mirror, a 3D sensor head houses multiple measuring



**Figure 2.6.** Illustration of a single beam emitting and receiving light pulse, as described in Eq. (2.9).

units, with modern ones accommodating up to 128 units.

Scanners also measure the intensity of measurements as the amount of light reflected from the surface. This intensity information is normalized and discretized to an 8 or 16 bit value. The intensity depends on surface characteristics, and several other factors impact the measurement. All terms in Eq. (2.9) are continuous. Thus, we can expect that mild changes of the viewpoint yield mild variations of the intensity when measuring the same 3D point.

Modern LiDAR point clouds are dense enough to be converted into images resembling those captured by conventional cameras. There are various methods to transform a LiDAR point cloud into an image. The most prevalent approach is to generate a panoramic image by horizontally aligning all vertical beams. For instance, if we have 128 vertical beams and our encoder completes 1024 steps for a full 360-degree rotation, we can easily construct a panoramic image with 128 rows and 1024 columns. This method is termed "projection by ID". However, a significant limitation of this technique is the absence of a continuous differentiable function, making it less suitable for state estimation. Due to this limitation and for symmetry with the classic pinhole projection, we define the spherical projection and its inverse as a function to map points from 3D point cloud to image and vice-versa. Some details and explanations about LiDAR available in the market (at the time of writing) are shown in Fig. 2.5.

### Spherical LiDAR model

The spherical model refers to the process of mapping or projecting points from a three-dimensional space onto a two-dimensional spherical surface and vice-versa mapping points from the image to the 3D space. It is a form of geometric transformation used in various fields, including cartography, computer graphics, and vision systems. Different from the pinhole model, this involves a conversion between Cartesian and spherical coordinates before flattening into the image space.

**Projection** The same quantities presented in the pinhole projection Sec. 2.2.2 apply here too. Let  $\mathbf{K}$  be a camera matrix in the form of Eq. (2.5), where  $f_x$  and  $f_y$  specify respectively the resolution of azimuth and elevation and  $c_x$  and  $c_y$  their offset in pixels. The function  $\psi$  maps a 3D point to azimuth and elevation. Thus, the spherical projection of a point is given

by

$$\pi_s(\mathbf{p}) = \mathbf{K}_{[1,2]}\psi(\mathbf{p}) \quad (2.10)$$

$$\psi(\mathbf{v}) = \begin{bmatrix} \text{atan2}(v_y, v_x) \\ \text{atan2}\left(v_z, \sqrt{v_x^2 + v_y^2}\right) \\ 1 \end{bmatrix} \quad (2.11)$$

In the spherical model  $\mathbf{K}_{[1,2]} \in \mathbb{R}^{2 \times 3}$ , being the third row in  $\mathbf{K}$  suppressed. We parameterize the spherical projection with  $\mathbf{K}$  for symmetry with respect to the pinhole model, however this camera matrix contains LiDAR azimuth and elevation resolution. These are calculated as follows:

$$\theta_{\text{res}} = \frac{\max h_{\text{FoV}} - \min h_{\text{FoV}}}{\text{rows}} \quad \phi_{\text{res}} = \frac{\max v_{\text{FoV}} - \min v_{\text{FoV}}}{\text{cols}} \quad (2.12)$$

where  $\min v_{\text{FoV}}$  and  $\max v_{\text{FoV}}$  are taken from the LiDAR datasheet, while  $\min h_{\text{FoV}}$  and  $\max h_{\text{FoV}}$  depends on the rotation done by the encoder (to acquire an omnidirectional point cloud, these are respectively 0 and 360 deg). The number of rows are equivalent to the number of beams vertically distributed, while the number of cols depends on the number of steps done by the encoder. The focal lengths and principal points are simply parameterized by:

$$f_x = -\frac{1}{\theta_{\text{res}}} \quad c_x = \frac{\text{cols}}{2} \quad (2.13)$$

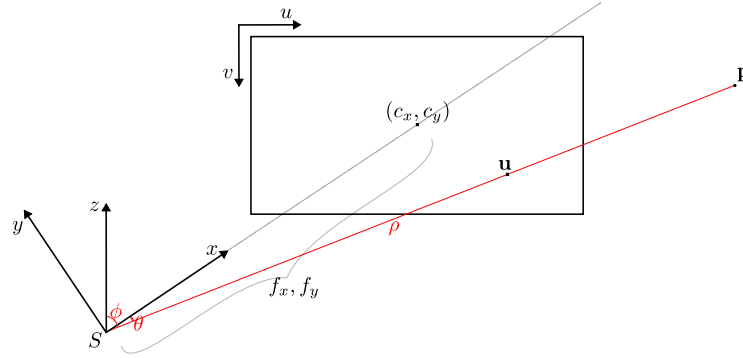
$$f_y = -\frac{1}{\phi_{\text{res}}} \quad c_y = \frac{\text{rows}}{2} \quad (2.14)$$

**Inverse projection.** Analogously to the pinhole inverse projection, the LiDAR one takes as input the (sub)pixel coordinate within the image. However, unlike the former, it uses the *range* of the observed point from the LiDAR at that specific image position instead of the *depth*. It is important to emphasize the difference between *range* and *depth*. While *depth* is measured along the z-axis of the optical frame (z-forward), the *range* is simply the distance between the origin  $S$  and the point  $\mathbf{p}$  expressed in the sensor reference frame (i.e., the L2-norm of the point) (see Fig. 2.7 for a visual explanation, range  $\rho$  highlighted in red). Mathematically, we can parameterize the spherical inverse projection as:

$$\mathbf{p} = \rho \psi^{-1} \left( \mathbf{K}_{[1,2]}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \right) \quad \psi^{-1} = \begin{bmatrix} \cos \theta \cos \phi \\ \sin \theta \cos \phi \\ \sin \phi \end{bmatrix} \quad (2.15)$$

### LiDAR model by ID

In previous section, we delved into a differentiable projection tailored for LiDAR, which is predominantly employed when state estimation came into play (essentially in all chapters except for Chapter 5). However, as shown in Fig. 2.8, one require the image for other applications. Let  $\mathbf{p}$  be a point detected by the LiDAR and expressed in its frame. Its



**Figure 2.7.** Illustration of the spherical model. Unlike the optical frame, the LiDAR reference frame is conventionally set with the z-axis pointing up. In the illustration, intersections with the image plane are drawn with crosses, while those with the 3D world are highlighted with full circles. The spherical coordinates are sketched in red, with  $\theta$  representing the azimuth,  $\phi$  the elevation, and  $\rho$  the range.

projection is computed as:

$$\pi_{\text{id}}(\mathbf{p}) = \mathbf{A}\psi(\mathbf{p}) \quad (2.16)$$

$$\mathbf{A} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & 1 & 0 \end{bmatrix} \quad (2.17)$$

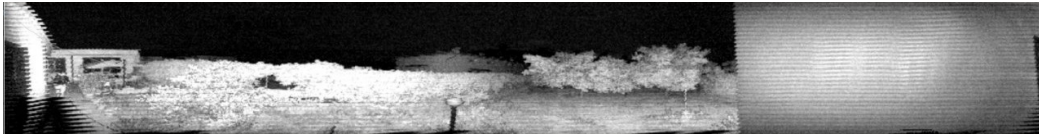
$$\psi(v) = \begin{bmatrix} \text{atan2}(v_y, v_x) \\ \text{ring}(v) \\ 1 \end{bmatrix} \quad (2.18)$$

where  $f_x$  and  $c_x$  are respectively focal length and principal offsets along azimuthal coordinates. The  $\text{ring}(v)$  function is either obtained directly from the LiDAR sensor, which augment every measured point with a number that represents the beam that detected it or, assuming the cloud is ordered, by dividing the point index by the horizontal resolution of the sensor. Compared with the spherical projection, the projection by ID does not preserve the geometric consistency of the scene. Still, it provides an image with no holes, hence what kind of projection using depends always on the application. The element of the point clouds (equivalent to perform an inverse projection) can be retrieved simply by look-up table.

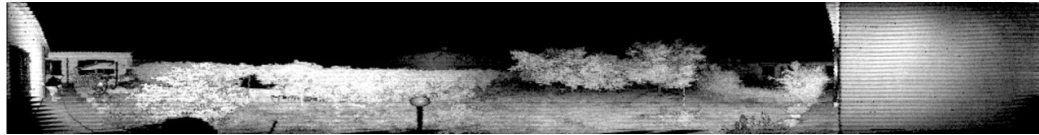
### 2.2.4 Image types

In computer vision, an image is a representation of a visual scene and stored digitally. Images can be captured by directly by cameras or generated by projection, as illustrated in previous sections. It is typically represented as a matrix (or a two-dimensional array) of pixel values. Each pixel value, depending on the image type, can represent various things:

- *grayscale image*: each pixel value represents a shade of gray, typically ranging from 0 (black) to 255 (white);



(a) Projection by ID. While the image appears without gaps, the function is non-differentiable. Moreover, horizontal stacking of beams distorts the world points, leading to artifacts manifested as small triangular shapes. More details in Sec. 2.2.3.



(b) Spherical projection. While the function is differentiable and the world is represented in the image how is perceived, the image may display empty gaps due to potential miscalibration of the LiDAR intrinsics. More details in Sec. 2.2.3.

**Figure 2.8.** Differences between "projection by ID" and Spherical projection.

- *color image* (often in RGB format): each pixel is usually composed of three values corresponding to the Red, Green, and Blue color channels. Each channel typically ranges from 0 to 255, and together, they define the color of the pixel;
- *binary image*: Each pixel has one of two possible values, often 0 (representing black) or 1 (representing white);
- *depth image*: in some applications, an image might represent distances instead of color. Each pixel value in a depth image corresponds to a distance from the sensor (depth or range).
- *surface normal image*: often simply referred to as a "normal map" or "normal image" represents the orientation of surfaces in a scene. Each pixel in a normal image encodes the direction that the surface at that location, relative to some coordinate system. In 3D these are commonly encoded through RGB.

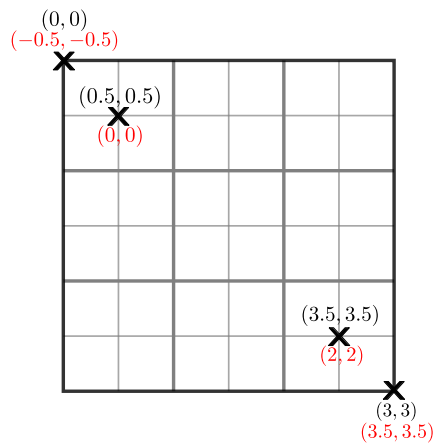
Images are the main data source for computer vision tasks. While there are many types beyond the ones listed earlier, including more complex ones (i.e. hyperspectral images), this thesis focuses on the most commonly used types, especially for 3D reconstructions.

### 2.2.5 Image coordinate conventions

Intrinsic calibration has a nuance that can be easily missed: there are two prevalent definitions for image coordinate systems. This distinction matters when exchanging or scaling intrinsic calibrations. In both definitions, as common in standard digital image representations the x-axis in the image (denoted with  $u$  in Fig. 2.3 and Fig. 2.7) points right, and the y-axis points down (denoted with  $v$  in Fig. 2.3 and Fig. 2.7).

The first system, termed the "pixel-corner convention," places the origin at the top-left *corner* of the top-left pixel. Conversely, the "pixel-center convention" positions the origin at the *center* of the top-left pixel. Fig. 2.9 depicts both conventions using a sample image.

While coordinates in both systems can be inter-converted, each is best suited for specific applications. The pixel-corner convention is handy when projecting points to pixels to



**Figure 2.9.** Image coordinate systems. Example coordinates values for *pixel-corner* in black and *pixel-center* in red, within a  $3 \times 3$  image.

determine the exact pixel a point maps to. On the other hand, the pixel-center convention is ideal for tasks like bilinear interpolation with control points at image centers or when inverse projecting pixel centers.

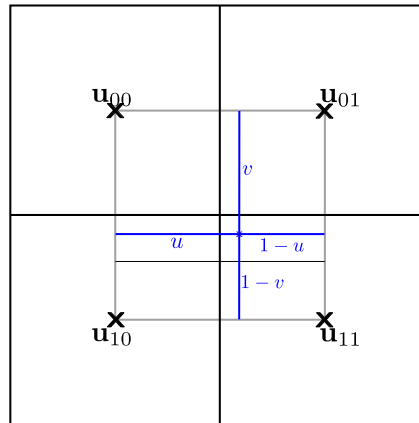
It is crucial to recognize the significance of the chosen convention when scaling intrinsic calibrations, especially for multi-resolution methods working on varied image scales. With the pixel-corner convention, pinhole parameters can be linearly scaled using a straightforward multiplication by the scaling factor. However, in the pixel-center convention, the principal point's location does not scale linearly. Instead, it should first be converted to the pixel-corner convention by adding 0.5 to its coordinates, scaled, and then reverted to the pixel-center convention by subtracting 0.5.

Given that many implementations do not specify their convention, there is a risk of confusion, leading to a half-pixel error in intrinsic calibration. Although this typically has minimal real-world impact, it is an unnecessary source of inaccuracy.

### 2.2.6 Image interpolation

The preceding sections detailed the mapping between 3D points and (sub)pixels in an image. What remains is understanding how to interpret sub-pixel image coordinates resulting from projection.

At its core, each image pixel typically captures a singular intensity measurement (passive light for cameras, active for LiDAR). Ideally, to best mirror reality, each pixel should be viewed as an individual discrete intensity sensor, capturing specific light frequencies and directions of incoming light. While this perspective is essential and should be pursued for optimal results, practical runtime performance often necessitates approximations to enhance processing speed. A frequent requirement, such as for continuous optimization (as mentioned in Sec. 2.3), is the computation of image gradients, viewing the image as a continuous function rather than discrete pixels. This can be achieved through pixel interpolation, with bilinear interpolation being the most prevalent method, which we will delve into in this section.



**Figure 2.10.** Bilinear interpolation illustrated for a group of adjacent  $2 \times 2$  pixels. The control points  $\mathbf{u}_{00}$  to  $\mathbf{u}_{11}$  are located in the pixel centers. The value is interpolated for the blue cross at relative position  $(u, v)^T$ . Each control point's value is weighted with the area of the rectangle between the control point that is opposite to it and point  $(u, v)^T$ .

### Bilinear interpolation

Imagine a set of  $2 \times 2$  neighboring pixels in an image. Each pixel's intensity value is linked to a control point situated at the pixel's center. Bilinear interpolation provides interpolated values for the square region between these four control points. For a subpixel position within this area, we determine its  $u$  and  $v$  offsets from the top-left control point. This offset would be  $(0, 0)$  directly at the top-left control point and  $(1, 1)$  at the bottom-right control point. The values of the control points are labeled as  $\mathbf{u}_{00}$ ,  $\mathbf{u}_{01}$ ,  $\mathbf{u}_{10}$ , and  $\mathbf{u}_{11}$ , which can be either vectors or scalars. Using these, the desired interpolated value is then derived:

$$(1 - u)(1 - v)\mathbf{u}_{00} + u(1 - v)\mathbf{u}_{01} + (1 - u)v\mathbf{u}_{10} + uv\mathbf{u}_{11} \quad (2.19)$$

As depicted in Fig. 2.10, to interpolate across the entire image, one would first identify the relevant control point square and then utilize the aforementioned formula for that square. This method ensures continuity throughout the image and allows for differentiation within each square interpolation region.

Bilinear interpolation, as the term indicates, is not purely linear. However, it exhibits linear (i.e. affine) behavior when examined along lines parallel to either  $u$  or the  $v$  direction, equivalently if  $u$  or  $v$  is held constant. Along any other straight line, the interpolant is quadratic. Even though the interpolation is not linear in the position ( $u$  and  $v$ ), at a fixed point it is linear in the interpolation values.

The result of bilinear interpolation is independent of which axis is interpolated first and which second. If we had first performed the linear interpolation in the  $v$  direction and then in the  $u$  direction, the resulting approximation would be the same.

#### 2.2.7 Rolling shutter and “skewing” effects

Camera shutters are categorized into two primary types. The global shutter captures an image by simultaneously exposing all its pixels to the incoming light, resulting in a consistent snapshot of the scene throughout the exposure period. On the other hand, the rolling shutter exposes pixels in a sequential manner, either row-by-row or column-by-column.

This sequential exposure means that different pixels record the scene at varying times. Movements within the scene or shifts of the camera during this process can introduce image distortions. These distortions can adversely affect 3D reconstruction if not properly managed. While the distortion’s impact lessens when the camera is moved gently, maintaining such stability with hand-held devices is difficult due to unavoidable minor tremors. Notably, these minor movements, especially rotational ones compared to slight positional shifts relative to the scene’s depth, can considerably modify the captured image, heightening its vulnerability to the rolling shutter effect.

Mechanical LiDARs exhibit a similar effect to the camera’s rolling shutter. As previously discussed, mechanical LiDARs capture the world vertically "simultaneously" but sense the scene horizontally over the time it takes for the encoder to complete the intended rotation. This leads to a “skewing” effect, and, as for rolling shutter cameras, the magnitude of this effect is directly proportional to the sensor’s movement speed. Various techniques, such as those mentioned in [102, 28], can be employed to deskew point clouds. The most effective methods often depend on an inertial sensor to gauge the sensor’s speed at each encoder step. To ensure broader applicability across sensors, the research presented in this thesis does not factor in the rolling shutter effect or point cloud deskewing.

### 2.3 Non-linear Continuous Optimization

This section delves into the utility of non-linear optimization for continuous optimization challenges. In many Computer Vision tasks, while direct solutions might be elusive, it is often straightforward to design a function that gauges the quality of a potential solution. Take, for instance, the task of aligning overlapping photos for a panorama. Directly devising an algorithm for this might be challenging, but assessing the quality of an alignment is simpler. One could measure the quality by calculating the sum of squared differences between corresponding pixels in overlapping image sections. Leveraging image interpolation, this cost function (or objective function) can be smoothly varied, enabling the computation of informative Jacobians. With a rough initial alignment, perhaps from keypoint matching or gyroscope data, non-linear optimization can refine this to a locally optimal alignment. If the starting point is decent, the outcome is likely satisfactory. Essentially, with a clear cost function, we can tackle tasks by defining the desired outcome. The optimization techniques discussed here can address various problems, provided there is a measurable cost function and a reasonable initial estimate. They offer a way to specify the ideal solution criteria and then find a solution that aligns with it.

#### Cost function

The problems addressed in this thesis typically adhere to the following structure. We have a cost function, that indicates the progress in solving a task. This function relies on a *state* vector,  $\mathbf{x}_k$ , which captures the current status of the task (for instance, the current alignment of images as mentioned earlier). The cost function is essentially a sum of squared *residuals*,  $\mathbf{e}_k$ , that change based on the state.  $\|\cdot\|_{\Omega}$  represents the squared Mahalanobis distance. These squared residuals can be tweaked by a *robust loss function*,  $L$ , which might be employed to lessen the influence of outliers:



$$E(\mathbf{x}) = \sum_k L \left\| \mathbf{e}_k(\mathbf{x}_k)^T \mathbf{e}_k(\mathbf{x}_k) \right\|_{\Omega} = \sum_k L \left\| \mathbf{e}_k(\mathbf{x}_k) \right\|_{\Omega}^2 \quad (2.20)$$

Here,  $k$  refers to the  $k^{\text{th}}$  measurement, which is only influenced by a subset  $\mathbf{x}_k \in \mathbf{x}$  of the overall state vector  $\mathbf{x}$ .  $L$  should be symmetric with a unique minimum at zero. The requirements that a robust loss function (as formulated on non-squared scalar residuals, i.e.,  $\bar{L}(e(\mathbf{x})) = L(e(\mathbf{x})^2)$ ) are:

- for robust loss functions  $\bar{L}$ , the *influence function*  $\frac{d\bar{L}(x)}{dx}$ ;
- $\bar{L}(e(\mathbf{x}))$  should be convex in  $\mathbf{x}$ ;
- whenever  $\frac{d^2\bar{L}(e(\mathbf{x}))}{d^2\mathbf{x}}$  is singular, it should hold:  $\frac{d\bar{L}(e(\mathbf{x}))}{d\mathbf{x}} \neq 0$ .

For optimal convergence, it is essential that the residuals are minimal and have a zero mean at their best point. This means that the values of the various  $\mathbf{e}_k$  should hover around zero, both positively and negatively, when in a solution state. For instance, rather than utilizing a vector's norm as a residual (which is invariably positive), it might be more beneficial to use the vector directly as the residual. The rationale behind this will be evident in the subsequent derivation.

### 2.3.1 Gauss-Newton method

In this section, we will delve into the Gauss-Newton (GN) method, a strategy for optimizing states concerning the cost functions outlined earlier. This method's exposition draws inspiration in part from the `srrg2_solver` documentation [49]. We start with an initial state estimate,  $\mathbf{x}$ , aiming to adjust it to minimize the cost function  $E(\mathbf{x})$ .

For the algorithm to work effectively, it needs insight into the shape of the cost function. Given that evaluating this function is typically resource-intensive, it is crucial to limit the number of evaluations. If evaluations were cheap and quick, we could assess the function for all possible state values  $\mathbf{x}$  and choose the one yielding the lowest cost. However, this approach is often impractical. Instead, the algorithm incrementally adjusts the state to reduce the cost progressively.

To grasp the local shape of the function, we employ linearization on the individual residuals within the cost. Near the current state  $\mathbf{x}$ , specifically for  $\mathbf{x} + \Delta\mathbf{x}$  where  $\Delta\mathbf{x}$  is a minor change, we approximate the residuals in the following manner:

$$\mathbf{e}_k(\mathbf{x} + \Delta\mathbf{x}) \approx \mathbf{e}_k(\mathbf{x}) + \mathbf{J}_k \Delta\mathbf{x} \quad (2.21)$$

In the equation  $\mathbf{J}_k$  denotes the Jacobian of the residual  $\mathbf{e}_k(\mathbf{x})$  with respect to  $\mathbf{x}$  at the current value of  $\mathbf{x}$ . We set in these approximations into the cost function  $E$  from Eq. (2.20) to obtain a local approximation for the complete cost:

$$E(\mathbf{x} + \Delta\mathbf{x}) = \sum_k L \left\| (\mathbf{e}_k(\mathbf{x} + \Delta\mathbf{x}))^T \mathbf{e}_k(\mathbf{x} + \Delta\mathbf{x}) \right\|_{\Omega} \quad (2.22)$$

$$\approx \sum_k L \left\| (\mathbf{e}_k(\mathbf{x} + \mathbf{J}_k \Delta\mathbf{x}))^T \mathbf{e}_k(\mathbf{x} + \Delta\mathbf{J}_k \mathbf{x}) \right\|_{\Omega} \quad (2.23)$$

$$= \sum_k L \left\| \mathbf{e}_k(\mathbf{x})^T \mathbf{e}_k(\mathbf{x}) + 2(\mathbf{e}_k(\mathbf{x}))^T \mathbf{J}_k \Delta\mathbf{x} + \mathbf{x}^T \mathbf{J}_k^T \mathbf{J}_k \Delta\mathbf{x} \right\|_{\Omega} \quad (2.24)$$

We use the concept of iteratively re-weighted optimization to manage the loss functions, denoted as  $L$ . To determine the appropriate weight for each residual, consider that the optimal solution to the minimization issue should result in a zero derivative of the cost with respect to every component of  $\mathbf{x}$ . Using the chain rule and dropping the  $\Omega$ -norm for simplicity, for each component  $m$  of  $\mathbf{x}$ , we can derive the following:

$$\sum_k \frac{dL(x)}{dx} \frac{\partial(\mathbf{e}_k(\mathbf{x}))^T \mathbf{e}_k(\mathbf{x})}{\partial x_m} = 0 \quad (2.25)$$

Therefore, for weighted least squares:

$$\sum_k w_k \frac{\partial(\mathbf{e}_k(\mathbf{x}))^T \mathbf{e}_k(\mathbf{x})}{\partial x_m} = 0 \quad (2.26)$$

Hence, to obtain the desired solution we need to set  $w_k$  as:

$$w_k := \left. \frac{dL(x)}{dx} \right|_{(\mathbf{e}_k(\mathbf{x}))^T \mathbf{e}_k(\mathbf{x})} \quad (2.27)$$

In the iteratively re-weighted optimization approach, these weights are treated as fixed during the update calculation. Incorporating these weights, we derive the expression to minimize as follows:

$$E(\mathbf{x} + \Delta \mathbf{x}) = \sum_k L \left\| \mathbf{e}_k(\mathbf{x})^T w_k \mathbf{e}_k(\mathbf{x}) + 2(\mathbf{e}_k(\mathbf{x}))^T w_k \mathbf{J}_k \Delta \mathbf{x} + \mathbf{x}^T \mathbf{J}_k^T \mathbf{J}_k w_k \Delta \mathbf{x} \right\|_{\Omega} \quad (2.28)$$

As common, we group the different terms using standard notation of non-linear optimization:

$$\mathbf{c} = \sum_k w_k \mathbf{e}_k(\mathbf{x})^T \Omega_k \mathbf{e}_k(\mathbf{x}) \quad (2.29)$$

$$\mathbf{b} = \sum_k w_k \mathbf{J}_k^T \Omega_k \mathbf{e}_k(\mathbf{x}) \quad (2.30)$$

$$\mathbf{H} = \sum_k w_k \mathbf{J}_k^T \Omega_k \mathbf{J}_k \quad (2.31)$$

Note that in Eq. (2.30) terms have been inverted due to transposition. Ignoring the weighting,  $c$  represents the present value of the cost function. Meanwhile,  $\mathbf{b}$  is half the gradient of  $E(\mathbf{x})$  at point  $\mathbf{x}$ , and  $\mathbf{H}$  stands for half of the symmetric and semi-positive-definite 'Gauss-Newton' approximation to the Hessian of the cost at  $\mathbf{x}$ . The accuracy of this approximation will be explored in subsequent discussions. Reframing the cost approximation with the newly introduced variables gives:

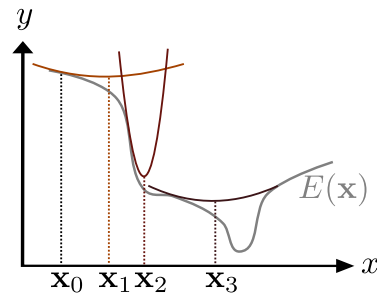
$$c + 2\mathbf{b}^T \Delta \mathbf{x} + \Delta \mathbf{x}^T \mathbf{H} \Delta \mathbf{x} \quad (2.32)$$

This quadratic vector expression can be minimized by taking the derivative with respect to  $\Delta \mathbf{x}$  and finding the point where it is zero,  $2\mathbf{b}^T + 2(\mathbf{H} \Delta \mathbf{x})^T = 0$ , thus leading to:

$$\mathbf{H} \Delta \mathbf{x} = -\mathbf{b} \quad (2.33)$$

$$\Delta \mathbf{x} = -\mathbf{H}^{-1} \mathbf{b} \quad (2.34)$$

The state-update can be done using the one which minimizes the approximated cost by iteratively setting  $\mathbf{x} := \mathbf{x} + \Delta \mathbf{x}$ . This process of iterative approximation is illustrated in Fig. 2.11



**Figure 2.11.** A brief overview of the Gauss-Newton algorithm across several iterations: beginning with the state  $\mathbf{x}_0$  (the initial-guess), the cost function (depicted in gray) is approximated using a parabola (in orange). The subsequent state,  $\mathbf{x}_1$ , is identified as the parabola's minimum, at which point a new parabola (shown in red) is employed for further approximation. This process is reiterated, leading to  $\mathbf{x}_2$  and ultimately  $\mathbf{x}_3$ . By this stage, the approximation is closely aligned with the cost function's true minimum.

Keep in mind that the update relies on a linear approximation of the residuals at the current state  $\mathbf{x}$ . If this approximation does not capture the entirety of the cost function, it is improbable that a single update will pinpoint the cost's global minimum. However, the hope is that it will enhance the cost, provided the local approximation is reasonably accurate, though this is not always guaranteed. After implementing the update, the linearization and solution process is reiterated from the updated state, aiming to refine the solution progressively. If this iterative method converges, it will land on a stationary point of the original non-linear cost function, such as a local minimum or a saddle point with a zero derivative. This technique is recognized as the Gauss-Newton method. The iterative approximation process is illustrated in Fig. 2.11. The accuracy of the Gauss-Newton approximation to the Hessian plays a crucial role in the algorithm's convergence. This approximation will be beneficial only if it is sufficiently accurate. To evaluate its effectiveness, one should examine the complete Hessian. Without considering the weighting, where  $j$  goes over all elements in vector  $\mathbf{e}_k$ , and  $\nabla^2(\mathbf{e}_k(\mathbf{x}))_j$  is the Hessian of the  $j$ -th element in  $\mathbf{e}_k(\mathbf{x})$ :

$$2 \sum_k \left( \mathbf{J}_k^T \mathbf{J}_k + \sum_j (\mathbf{e}_k(\mathbf{x}))_j \nabla^2(\mathbf{e}_k(\mathbf{x}))_j \right) \quad (2.35)$$

The Hessian approximation primarily focuses on the terms  $\mathbf{J}_k^T \mathbf{J}_k$  on the left side. Given that the term on the right is influenced by a specific element of the residuals, it is likely small when the residuals themselves are small. If these residuals are centered around zero, there is a good chance these terms balance each other out to some degree. Such characteristics of the residuals enhance the convergence efficiency of the Gauss-Newton method, as highlighted in the initial discussions about the cost function requirements at the beginning of this section. The convergence is also better when the residual  $\mathbf{e}_k$  behaves almost linearly at their minimum, leading to a near-zero value for that specific element  $\nabla^2(\mathbf{e}_k(\mathbf{x}))_j$ .

### 2.3.2 Levenberg-Marquardt method

As highlighted in the preceding section, the GN method determines update steps that do not always result in a reduced cost. Depending on the nature of the cost function, this can pose problems, potentially causing divergence or state oscillations. The Levenberg-Marquardt (LM) method offers a solution to enhance performance in such scenarios. This method introduces a parameter  $\lambda \geq 0$ , which is always non-negative. One might initialize  $\lambda$  with a nominal small positive number or perhaps a factor of the average diagonal component from the initial  $\mathbf{H}$  matrix in the Gauss-Newton process. Consequently, the calculation of the update step  $\Delta \mathbf{x}$  (as in Eq. (2.28)) is adjusted as follows:

$$\Delta \mathbf{x} = -(\mathbf{H} + \lambda \mathbf{I})^{-1} \mathbf{b} \quad (2.36)$$

In essence,  $\lambda$  is added to each diagonal of  $\mathbf{H}$ . The Levenberg-Marquardt method then calculates the cost at the updated state,  $E(\mathbf{x} + \Delta \mathbf{x})$ . If the new cost is lower,  $\lambda$  is reduced (e.g., halved) and the new state is accepted. If the cost rises, the update is undone,  $\lambda$  is increased (e.g., doubled), and another attempt is made. If no steps reduce the cost and  $\lambda$  keeps growing, the algorithm stops, indicating likely convergence. This method ensures no steps increase the cost.

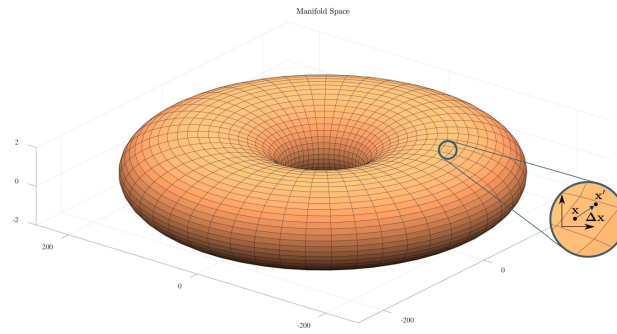
Let's break down the implications of adding a constant to the diagonal of  $\mathbf{H}$ . Clearly, when  $\lambda = 0$ , the update mirrors the Gauss-Newton step. As  $\lambda$  approaches infinity,  $\mathbf{H}$ 's influence diminishes, making the update roughly  $-(\lambda \mathbf{I})^{-1} \mathbf{b} = -\frac{1}{\lambda} \mathbf{b}$ . Given that  $\mathbf{b}$  is half the gradient of  $E(\mathbf{x})$  at  $\mathbf{x}$ , the update step becomes a scaled version of the negative gradient, with the scale factor inversely related to  $\lambda$ . This mirrors the gradient descent approach. Hence, Levenberg-Marquardt acts as a bridge between Gauss-Newton and gradient descent. It leans on GN when the Hessian approximation  $\mathbf{H}$  is reliable, and reverts to gradient descent, adjusting step size through  $\lambda$ , when it is not. The method targets either a saddle point or a local minimum, which is only assured to be the global minimum if the cost function is convex.

In Computer Vision, a frequently encountered but seldom discussed challenge with the Levenberg-Marquardt method is residual validity. Take the example of stitching two images (as presented previously), where the cost function sums up pixel value differences in the overlapping regions. As images are repositioned during optimization, their overlapping areas shift, altering the set of pixels contributing to the residuals. Ideally, we'd want maximum pixels to aid in achieving the best alignment. But, the method must compare costs across different states to ensure if a step is beneficial. If the residual sets vary between states, how can a fair comparison be made? A simplistic approach might be to assign a constant cost (like zero) to residuals present in only one state. However, this can mistake updates that alter residual validities.

### 2.3.3 Optimizaton on manifolds

Sometimes, the components we aim to optimize are not in a Euclidean space but exist on a manifold. A prevalent example in Computer Vision is camera orientations, which belong to the 3D special orthogonal group,  $\mathbb{S}\mathbb{O}(3)$ . A more straightforward example is 3D unit-length directions, represented by 3D vectors but constrained to have a length of one.

Directly optimizing these elements' representations is problematic because the optimization might overlook these constraints, leading to incorrect updates. One workaround is to



**Figure 2.12.** Illustration of a Manifold space. Since the manifold is smooth, local perturbations - i.e.  $\Delta \mathbf{x}$  in the illustration - can be expressed with a suitable Euclidean vector. Illustration courtesy of Grisetti *et al.* [49].

use local Euclidean spaces that serve as tangent spaces to the elements' current state. This allows updates to be calculated in the tangent space, free from the original constraints. The updated result can then be translated back to the element's original representation. Think of this in terms of optimizing 3D directions with a unit length.

Optimizing the components of such a direction vector might breach the unit-length requirement. For a specific direction, we can establish a tangent space, symbolized by a plane with the direction vector as its normal. Within this plane, we can identify two orthogonal direction vectors to form a two-dimensional coordinate system. This serves as the tangent space. Instead of a three-dimensional update to the direction vector, the optimization computes a two-dimensional update within this system. After determining an update step, which is a vector in the plane, we can combine it with the initial direction vector and then adjust the result to unit length. For subsequent steps, a new tangent space is defined based on the updated direction. While this example might seem overly simplistic, as one could also re-normalize after a full 3D update, it showcases the concept clearly. Moreover, 3D updates would introduce an unnecessary degree of freedom.

Let's structure our problem more clearly, focusing on the challenges highlighted in this thesis. The primary variables we deal with are homogeneous transformation matrices, denoted as  $\mathbb{SE}(3)$ . While these can be multiple, for the sake of clarity, let's consider a single variable represented by  $\mathbf{X}$  (we use  $\mathbf{X}$  and not  $\mathbf{T}$  since this is the quantity we are going to estimate). Instead of optimizing  $\mathbf{X}$  directly, we introduce a tangent space perturbation or *Lie algebra*,  $\Delta \mathbf{x}$ , which is of a smaller dimension, specifically  $\Delta \mathbf{x} \in \mathbb{R}^6$ . This perturbation,  $\Delta \mathbf{x}$ , encodes the translation and the imaginary part of the quaternion for us. To apply this perturbation to our variable, we use the update function  $\mathbf{X} \boxplus \Delta \mathbf{x}$ . This function leverages the exponential map,  $\exp(\cdot)$ , to elevate the element from the tangent space to the manifold  $\mathbb{SE}(3)$ . Formally, the update is given by:

$$\mathbf{X}' = \mathbf{X} \boxplus \Delta \mathbf{x} = \mathbf{X} \cdot \exp(\Delta \mathbf{x}) \quad (2.37)$$

In a similar vein, we introduce another operator,  $\boxminus$ , which computes the vector perturbation between two points on the manifold. It is defined as:

$$\Delta \mathbf{x} = \mathbf{X}' \boxminus \mathbf{X} = \log(\mathbf{X}' \mathbf{X}^{-1}) \quad (2.38)$$

Here,  $\exp$  and  $\log$  represent the *exponential* and *logarithmic map* at the identity, respectively. For compactness, we have discussed the scenario where only one variable is part of the

optimization. However, when multiple variables are involved, adjustments will be necessary. We will delve into the specifics later in the next chapters.

### 2.3.4 Factor-graphs

When the state dimension is composed by multiple variables (i.e., sensor poses, structure) it is important to exploit the relationship and connectivity of the variables to efficiently solve the problem: for this we employ factor-graphs. Factor-graphs are *undirect graphs*, usually used in probability theory and information theory to represent factorized representations of joint probability distributions. These unify and generalize several other graphical models, such as Bayesian networks and Markov random fields. In fields like SLAM and 3D reconstruction they are extremely important because, when multiple variables are involved, the problem is usually *sparse*. In other words, given a state composed by multiple entities  $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ , the measurements (constraints)  $\mathbf{z}$  relate only a subset of the whole state. For instance, in a graph composed by only  $\mathbb{SE}(3)$  poses (called usually *pose-graph*) the first pose may not be related to the third and fourth pose and so on, while it is related only to the second [62].

We extensively use factor-graphs in the works presented in this thesis. A factor-graph usually is composed by *variables* and *factors*. Variables represent usually the state we want to estimate, hence the position of the sensor  $\mathbb{SE}(3)$  with respect to a fixed reference frame and the 3D structure, factors (edges) represent the measurements, connecting the variables of the graph (i.e. camera measurements, GNSS readings, dead reckoning).

### 2.3.5 Efficiency and precision

In this thesis, three specific steps within the Gauss-Newton and Levenberg-Marquardt methods emerged as performance-critical in the research projects:

- with a high count of residuals, the computation of Jacobians  $\mathbf{J}_k$  and the accumulation of the matrix  $\mathbf{H}$  often lead to performance bottlenecks;
- when optimizing a significant number of variables, determining the update  $(\mathbf{H}^{-1}\mathbf{b})$  becomes a complex task;
- accumulating numerous numeric values in  $\mathbf{H}$  and  $\mathbf{b}$  can result in floating point errors and precision loss.

To mitigate the first challenge, strategies such as minimizing the residual count or utilizing GPU for  $\mathbf{H}$  computation can be effective. For the second issue, leveraging the sparsity of  $\mathbf{H}$  or seeking approximate solutions for the update can be beneficial. To address the third concern, it is essential to streamline the summation process, ensuring that values of similar magnitudes are summed together, thereby maintaining numerical stability. In our case, given the large number and independence of each pixels contributing to the final estimate we discuss how we mitigate these issues.

#### Minimum degree ordering

In numerical analysis, especially when dealing with sparse matrices, the order in which we process the rows and columns of a matrix can significantly affect the efficiency of certain

operations, such as matrix factorization. Consider a large  $n \times n$  system of equations given by

$$\mathbf{Ax} = \mathbf{b} \quad (2.39)$$

similar to one presented in Eq. (2.33), something we could possibly get in 3D reconstruction problems. In this section we just give a brief insight on how these kind of systems are efficiently solved, more details can be found in [45]. Here,  $\mathbf{A}$  is a symmetric positive definite matrix that is *sparse*. For a large  $n$ , when we factor  $\mathbf{A}$  using Cholesky's method, we often encounter some "fill", which is the introduction of non-zero entries in positions that were originally zero during matrix factorization. An interesting aspect is that  $\mathbf{PAP}^T$  remains symmetric and positive definite for any permutation matrix  $\mathbf{P}$ . This allows us to solve a reordered system:  $(\mathbf{PAP}^T)(\mathbf{Px}) = \mathbf{Pb}$ . The selection of  $\mathbf{P}$  can significantly influence the fill that arises during factorization. Therefore, reordering the rows and columns of the matrix before factorization has become a standard practice.

Typically, solving a sparse positive definite system involves four key, independent steps:

1. determining an appropriate ordering  $\mathbf{P}$  for  $\mathbf{A}$ ;
2. setting up a structure for  $\mathbf{L}$ , which will be the Cholesky factor of  $\mathbf{PAP}^T$ ;
3. numerically factoring  $\mathbf{PAP}^T$  into  $\mathbf{LL}^T$ ;
4. solving the equation  $\mathbf{LL}^T(\mathbf{Px}) = \mathbf{Pb}$ .

It is important to note that steps (1) and (2) are solely based on the structure of  $\mathbf{A}$ , and do not depend on its numerical values. One of the primary concerns is to minimize the fill, however, the problem of finding a best ordering for  $\mathbf{A}$  in the sense of minimizing the fill is computationally intractable: an NP-complete problem. We are therefore obliged to rely on heuristic algorithms. One of the most effective of these is the Basic Minimum Degree (BMD). The BMD ordering is one such heuristic method to reorder the rows and columns of a matrix to achieve this. We will give a brief explanation to get the intuition.

The main idea behind BMD is straightforward. It involves selecting the node (or row/column in the matrix context) with the minimum degree and eliminating it. The "degree" of a node in this context refers to the number of non-zero entries in its corresponding row or column.

---

#### Algorithm 1 Basic minimum degree ordering

---

**Data:**  $\mathcal{G}$  ▷ given a symmetric graph (original sparse matrix  $\mathbf{A}$ )  
determine the degree of each node  
**while**  $\mathcal{G} \neq \emptyset$  **do**  
    select the node with the smallest degree  $y$   
    delete  $y$  ▷ perform Gaussian elimination on the corresponding row/col  
    update the degrees of the remaining nodes

---

#### Floating-point stability and GPU

Floating-point arithmetic in computers is inherently imprecise due to the finite representation of numbers. When summing a sequence of floating-point numbers, especially if they vary

widely in magnitude, the accumulated round-off errors can lead to significant inaccuracies in the result. This is very common, in our circumstances where all the pixels of the image contribute to determine the quantities  $\mathbf{H}$  and  $\mathbf{b}$ . In single-thread implementation, the concept of compensated summation is a technique used to improve the numerical accuracy of summation operations. One of the most well-known algorithms for compensated summation is *Kahan's summation* algorithm [55]:

---

**Algorithm 2** Kahan summation algorithm
 

---

<b>Data:</b> <code>input-vec</code>	▷ input vector
<b>Result:</b> $S$	▷ sum
$S \leftarrow 0, c \leftarrow 0$	▷ the compensation $c$ will store and correct the error
<b>for all</b> $x$ <b>in</b> <code>input-vec</code> <b>do</b>	
$temp = S + y$	▷ temporarily store the sum
$c = (temp - S) - y$	▷ update the compensation
$S = temp$	▷ sum update

---

By using compensated summation techniques like Kahan's algorithm, one can achieve a result that is closer to the exact sum than would be obtained with straightforward summation, especially for long sequences of numbers.

In our multi-threaded GPU implementation, making the straightforward application of compensated sum is challenging, nonetheless, GPU sum reduction techniques can be beneficial. Sum reduction on GPUs efficiently aggregates large datasets by leveraging the platform's parallelism. This process involves dividing the data into smaller chunks, with each thread computing a local sum. These sums are stored in shared memory (part of the GPU memory accessible efficiently by all threads) and then combined hierarchically through a tree-reduction method. This method successively reduces the active threads by half until a single thread remains with the final sum. Finally, the results from all blocks are combined to get the overall sum.

Optimizing this process involves techniques like coalesced memory access, avoiding bank conflicts in shared memory, loop unrolling, and allowing threads to process multiple data points. Though, challenges like warp divergence and the potential bottleneck of atomic operations need careful handling. A detailed explanation can be found at [54].



## Chapter 3

# Visual Place Recognition using LiDAR Intensity Information

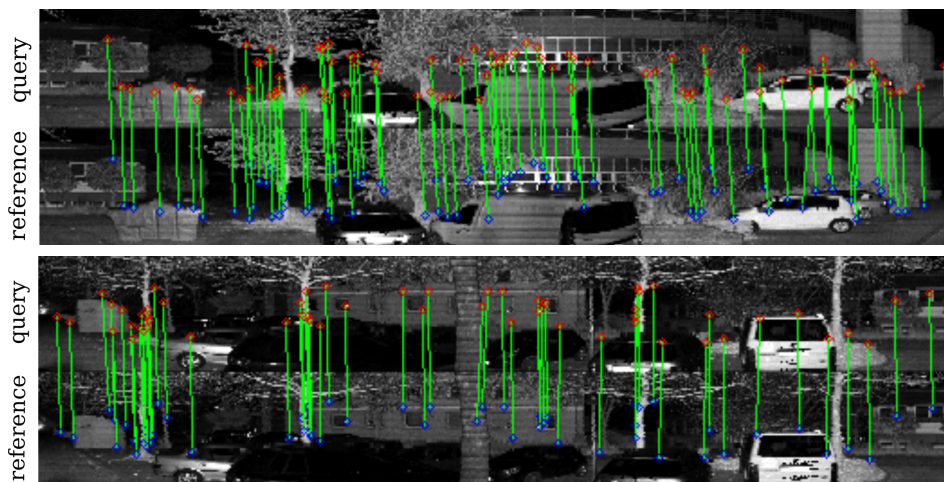
Place Recognition (PR) is a technique in robotics and computer vision that enables systems to recognize previously visited places by identifying specific locations from the environment. This capability is essential in many applications, including robotics, autonomous vehicles, and augmented reality, as it facilitates navigation and provides a deeper understanding of an environment's spatial context.

Within the domain of SLAM, the role of PR is essential. As robots navigate, errors in motion estimation can accumulate over time, leading to a drift in their perceived position. Place recognition comes into play here, identifying previously visited places, termed as *loops*. This identification allows the system to rectify these accumulated errors, aligning its current perception with the previously mapped environment. Moreover, when a robot is placed in an unfamiliar location within a known map, PR assists in determining its position within that map. This is particularly crucial in scenarios like the "kidnapped robot problem," where a robot, if moved to a different location, could lose its positional bearings. Furthermore, another advantage of PR is its ability to maintain the consistency and accuracy of SLAM systems, even across extensive operations or vast environments. By recognizing places it has already visited, PR ensures that the map remains streamlined, avoiding redundant expansions.

While existing 3D-LiDAR SLAM systems are adept at delivering accurate maps in real-time on standard computers, there are still challenges. A significant number of these systems either do not take into account loops [140, 141, 30] or depend on resource-intensive operations for potential loop-detection [13], such as ICP combined with outlier rejection mechanisms. This is because, detecting loops using only LiDAR data remains an open challenge. However, when it comes to camera images, this task is addressed as VPR, and there are already effective solutions in place [79, 131, 87].

As established in the introductory and background parts, it is evident that modern 3D LiDARs have made significant strides, outperforming their predecessors in accuracy and vertical resolution. When data from these LiDARs is processed into a panoramic image Sec. ??, the resulting picture is comparable to that of a grayscale camera.

This chapter contribution is to analyze the performance of existing visual place recognition techniques when they are applied to the intensity data derived from a 3D LiDAR. Through our evaluations of various VPR pipelines on multiple robotic/vision datasets using



**Figure 3.1.** Qualitative results. Top: Example pairs of a query and a reference intensity image from our self-recorded dataset (IPB Car). Note that due to its high horizontal resolution (original size:  $64 \times 1024$ ), the intensity image has been divided in two parts, one above the other. The green line illustrate descriptor matches provided by HBST [105] on intensity data. Bottom: Newer College [98] (left) and IPB car (right) datasets used for evaluation, valid loops detected using HBST highlighted in green.

3D LiDARs, we have found that adapting existing VPR techniques can indeed yield reliable loop-detection, making it feasible to implement LiDAR-only SLAM on a larger scale. A visual representation of the application of VPR approaches to LiDAR intensity cues can be seen in Fig. 3.1.

### 3.1 Related Work

The early loop-detection detection systems for 3D scans extracted features from the raw data. Several feature extractors have been proposed, each capturing some traits of a local neighborhood of the scene. Early studies in this direction were made by Johnson [61] and later by Huber [57]. The former extracted some local 3D features from local point cloud patches, describing the local surface around points with orientation. The latter built on top of Johnson’s Spin Images a methodology to perform global registration exploiting these features. In this sense, each *query* frame is compared with a database, and if the surfaces of the local descriptors are “similar” between query and reference, then a potential loop-closure is detected. Steder *et al.* [115] investigated novel point features that are extracted directly from range images, and later, they applied them in the context of loop-detection [114]. Finally, Steder *et al.* proposed to use more robust NARF features [117] together with Bag-of-Words-based search to increase the efficiency and the accuracy of the detection [116]. Orthogonally, Magnusson *et al.* investigated the use of Normal Distributed Transform (NDT) as features to match 3D scans [82]. This approach has been originally developed to perform registration

between scans; still, the authors demonstrated that NDT-based features capture enough structure to be used in the context of place recognition. Röhling *et al.* [100] investigated the use of histograms computed directly from the 3D point cloud to define a measure of the *similarity* of two scans. Novel types of descriptors have been investigated, exploiting additional data gathered by the LiDAR sensor – i.e., light remission of the beams [23, 53]. However, despite being very attractive, these descriptors are time-consuming to extract and match, resulting in a slower system overall.

More recently, deep learning approaches are spreading thanks to the increased computing power of today’s computers. Dubé *et al.* [36] proposed the detection and matching of segments to recognize whether we are observing an already visited place. Uy *et al.* [126] employed a CNN based on PointNet [96] to compute NetVLAD holistic descriptors [10] out of range images. Zaganidis *et al.* [138] used semantic information extracted from the point cloud [97] to enrich NDT features, resulting in more accurate and robust place recognition. Chen *et al.* [21], instead, developed an end-to-end solution to evaluate the overlap of two 3D scans together with a raw estimate of the yaw angle. Still, all deep-learning-based approaches require a great amount of data to perform training (most of the times also labeled) and a lot of computing power to work properly.

A lot of visual place recognition systems exploit features such as SURF [11] or SIFT [77] and several approaches apply bag-of-words techniques, i.e., they perform matching based on the appearance statistics of such features. To improve the robustness of appearance-based place recognition, Stumm *et al.* [120] consider the constellations of visual words and keeping track of their covisibility. Another popular approach for visual place recognition proposed by Galvez-Lopez *et al.* [42] proposes a bag of words approach using binary features for fast image retrieval. Single image visual localization in real-world outdoor environments is still an active field of research, and one popular approach used in robotics is FAB-MAP2 [25]. For across season matching using SIFT and SURF, Valgren and Lilienthal [127] propose to combine features and geometric constraints to improve the matching.

To deal with substantial variations in the visual input, it is useful to exploit sequence information for the alignment, compare [75, 83, 84, 87, 130, 131]. SeqSLAM [84] aims at matching image sequences under seasonal changes and computes a matching matrix that stores the similarity between the images in a query sequence and a database. Milford *et al.* [83] present a comprehensive study about the SeqSLAM performance on low-resolution images. Related to that, Naseer *et al.* [87] focus on sequence matching using a network flow approach and Vysotska *et al.* [130] extended this idea towards an online approach with lazy data association and build up a data association graph online on-demand, also allowing flexible trajectories in a follow-up work [131].

In this chapter, we investigate how to perform fast and accurate place-recognition using additional channels available in modern 3D-LiDAR sensors. Our approach applies well-known methodologies originally designed to work with camera images to 3D LiDARs data, exploiting the increased descriptiveness of such sensors. We perform multiple experiments with different combinations of features – image retrieval tools. Among the features, we picked computationally efficient binary ones like BRISK [72] and ORB [101]. Instead, as floating point descriptors, we selected SURF [11] and Superpoint, a more recent neural extractor that shows impressive results compared to older geometrical features [31]. Among image-retrieval tools, we use a Hamming Distance Embedding Binary Search Tree (HBST) [105], a tree-like structure that allows for descriptor search and insertion in logarithmic time by exploiting particular properties of binary feature descriptors, and DBoW2 [42]

(Bags of Binary Words for Fast Place Recognition in Image Sequences) that allows fast image retrieval based on the histogram of the distribution of words appearing in the image both for floating point and binary descriptors.

## 3.2 Visual Place Recognition applied to LiDAR Intensity Images

Popular VPR approaches often store a database of places in the form of a collection of image keypoints and descriptors (and potentially a coordinate in some world frame). The *keypoints* are *salient points* in the image, possibly corners and edges, while the *descriptors* encode the *appearances* around keypoints.

In the process of finding similar places, two images are regarded as similar if a substantial part of their keypoints' descriptors are close to each other. Performing VPR using this paradigm requires first to convert a query image into a set of keypoints and descriptors and second to efficiently find images with similar descriptors. Effective solutions are available to quickly find the potential matches in the database, see [79]. Among all, we focus specifically on HBST [105] and DBoW2 [42] as two prominent approaches.

An intensity image constructed from a laser scan has a number of rows equal to the number of vertical beams and a number of columns equal to the number of scanning steps along the azimuth. Unfortunately, most public datasets provide the scans as annotated point clouds, and recovering the beam measurements needed for image formation requires a cylindrical projection. Due to vehicle motion, round-offs, or unknown parameters, this projection will likely result in missing data in some parts of the image. These phenomena may hinder the straightforward feature extraction process.

In the following, we will first discuss how we handle image formation from a laser scan, and then we review the structure of a straightforward pipeline for VPR.

### 3.2.1 Correcting LiDAR image for consistent feature extraction

The uneven distribution of the vertical beams as well as calibration errors in the scanner's vertical FoV may lead to empty gaps in the resulting image, usually whole horizontal rows. Holes in an image can be detrimental to feature extraction due to a variety of reasons. These gaps introduce discontinuities, which can be problematic for algorithms that rely on smooth transitions, such as those based on gradients or edge detection.

Addressing this issue can be approached in various ways. One method involves using the "projection by ID" technique, as discussed in Sec. 2.2.3. However, this method is not differentiable. An alternative is to directly measure and determine the exact intrinsic parameters of the LiDAR ( $\max v_{\text{FoV}}$ ,  $\min v_{\text{FoV}}$ ), since datasheets might not always be accurate. Yet, this does not always guarantee a resolution to the problem. A third solution is vertical interpolation. In our experiments, we opted for this third approach. To eliminate the empty rows in the LiDAR image, we first identify them using a binary threshold combined with a horizontal kernel spanning the image's width. For each pixel in these void rows, we compute an interpolated value based on the values of the nearest valid rows above and below. This interpolation ensures smoother feature extraction, minimizing gradient spikes. However, for geometric verification, we avoid using these interpolated values to prevent potential inaccuracies. Fig. 3.2 shows the result of this procedure.



**Figure 3.2.** Empty lines removal. From top to bottom, original image after projection from 3D point cloud as explained in Sec. 3.2.1, detection of empty rows highlighted in red, result after image manipulation. Each image shown has been cropped to half of their horizontal size for better viewing.

### 3.2.2 Feature extraction

As stated at the beginning of this section, the feature extraction process aims at compressing an image in a set of interest points or *keypoints*. A *descriptor* vector captures the appearance of the image in the neighborhood of the keypoint. The detector outputs a set of keypoint  $\{\mathbf{k}_i = (u_i, v_i)^\top\}$ , in image coordinates. A key quality of a keypoint detector is its ability to identify points that are “salient” or “locally distinct”. In other words, a good detector will identify the projection of the same point in the world upon small changes in the viewpoint. Typical approaches consider the image gradient at different scales to compute keypoints. Thus, to successfully operate, a detector requires the gradients in the image to capture the local intensity difference at nearby regions of the world. Accordingly, these approaches do not work when the vertical resolution is too low, since in this case, changes in the gradient are dominated by sampling effects. Similarly, typical feature detectors operate on a small image patch from which they compute some quantity that is as invariant as possible to mild warpings of the patch itself. This ensures that regions of the image that look alike will result in similar descriptors. For each keypoint  $\mathbf{k}_i$ , the extractor computes a descriptor vector  $\mathbf{d}(\mathbf{k}_i)$ . This vector consists of either floating point or binary values.

We directly employed well-known combinations of feature detectors and extractors[72, 101, 12] whose C++ implementation is publicly available [15]. We also tested a more recent neural feature extractor by Detone et al. [31].

### 3.2.3 Feature-based VPR

Two images of the same scene acquired with similar viewpoints will have a high number of descriptors that have a small distance. To this extent, we should define a suitable metric  $e_d$  for this comparison. For floating point descriptors,  $e_d$  a standard choice is the Euclidean distance in  $\mathbb{R}^n$  (other metrics such as the *cos-similarity* could be employed instead). For binary ones, the Hamming distance is commonly employed.

Relying on the metric  $e_d$  and the invariant properties of the descriptors, we can find corresponding points between two images  $\mathcal{I}_q$  and  $\mathcal{I}_r$  by finding for each keypoint  $\mathbf{k}_q \in \mathcal{I}_q$

FAST	threshold	40
ORB	nFeatures	300
	scaleFactor	1.2
	scaleFactor	8
	nLevels	8
	edgeThreshold	15
BRISK	threshold	30
	nOctaves	3
	patternScale	1
SURF	hessianThreshold	400
	nOctaves	4
	nOctaveLayers	3
Superpoint	minProbability	0.05
	nFeatures	300

**Table 3.1.** Configuration of keypoints detector and descriptors extractors used.

the closest keypoint  $\mathbf{k}_r \in \mathcal{I}_r$  in the descriptor space:

$$\mathbf{k}_r^* = \underset{\mathbf{k}_r}{\operatorname{argmin}} (e_d(\mathbf{d}(\mathbf{k}_q), \mathbf{d}(\mathbf{k}_r))) : \mathbf{k}_q \in \mathcal{I}_q \quad \mathbf{k}_r \in \mathcal{I}_r. \quad (3.1)$$

A straightforward way to solve Eq. (3.1) is by *exhaustive search*. This process is complete since it returns all neighbors according to the distance metric. However, it quickly becomes prohibitive as the size of the database increases, thus preventing online operations. Efficient approaches that perform an approximate search are available. These methods usually organize the features in the database in a search structure. Common choices are search trees such as KD-trees or binary trees. The splitting criterion and the parameters of the tree control the completeness of the search.

Alternative methods preprocess the features in the image by describing each image as a histogram of “words”. The words are computed by determining a priori a “dictionary” from a training image set. The elements of the dictionary are the clusters of features in the training set. Each feature in an image will contribute to its histogram based on the “word” in the dictionary closest to the feature. As a representative for tree-based approaches, we use HBST [105], while for BoW, we used DBoW2 [42]. In the next section, we will discuss in more detail the experimental configuration.

### 3.3 Experimental Evaluation

This evaluation analyzes our combinations of feature extractors and VPR pipelines on intensity images generated from LiDAR scanner point clouds and intensity data. In more detail, we evaluate the following combinations:

- FAST - ORB - HBST

Dataset	Description	LiDAR	FoV [deg]	GT
Newer College (long) [98]	OD, campus-park	OS1-64 (Gen 1)	33.2	Ext. Loc.
IPB Car (self-recorded)	OD, urban	OS1-64 (Gen 2)	45	RTK-GPS
Ford Campus (00) [91]	OD, urban	HDL 64-E	26.9	RTK-GPS
KITTI (00) [44]	OD, urban	HDL 64-E	26.9	RTK-GPS

**Table 3.2.** Datasets we used for evaluation. OD abbreviate outdoor-dynamic, Ext. Loc. stands for External Localization System.

- FAST - BRISK - HBST
- FAST - ORB - DBoW2
- Superpoint - DBoW2
- FAST - SURF - DBoW2

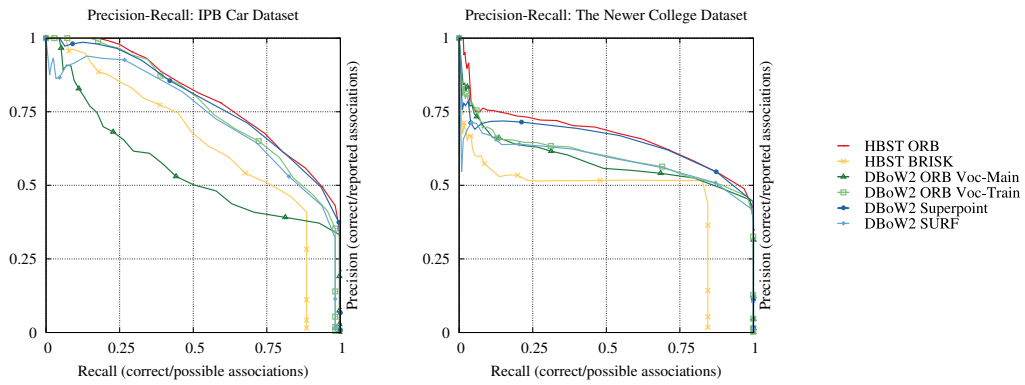
We test these configurations on four datasets, three publicly available, and one self-recorded dataset using an automated car, see Tab. 3.2 for dataset details. All datasets provide some cue of the robot position independent from the LiDAR sensor, which we use to construct ground truth place information. This is usually done with an RTK-GPS or an external reference system.

The ground truth is a set of matching pairs of scans acquired at nearby locations. A pair is a match if the scans’ recording locations are close according to the external system, and an ICP registration succeeds (up to 1 m in translation and 20° in rotation). We processed the datasets sequentially by adding the query image  $\mathcal{I}_q$  to the database at each step. We obtain a set (potentially empty) of images similar to  $\mathcal{I}_q$  at each query, and we verify these matches against the ground truth. From the returned images, we disregard those added within the most recent 120 steps. We do this to not positively bias the evaluation.

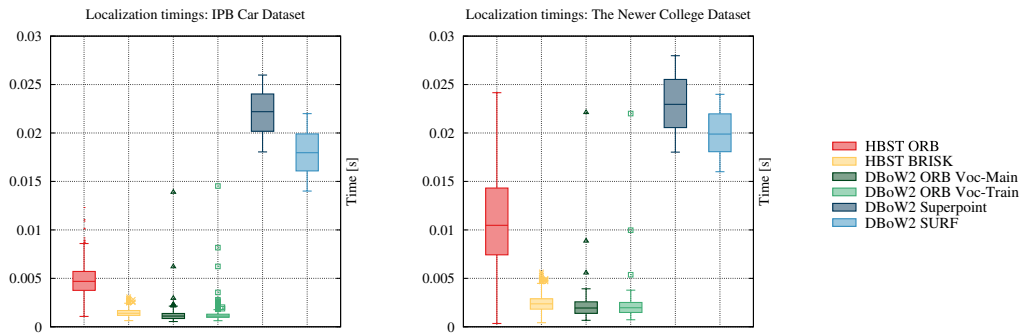
For each dataset, we tested the combination of feature extractor and image retrieval systems mentioned before. To quantify the performances of one run, we evaluated common statistical quantities – i.e., *Precision* and *Recall*. To this end, we introduce the terms *true positive* to indicate a loop-closure that is present in the ground truth database and *false positive* to indicate a wrong loop-closure. Analogously, a *false negative* represents a loop-closure that is present in the ground truth database but has not been reported by the method in analysis; a *true negative* represents its contrary. Hence, we can define Precision, Recall and F<sub>1</sub> Score using the number of true positives  $T_p$ , false positives  $F_p$ , true negatives  $T_n$  and false negatives  $F_n$  as follows:

$$P = \frac{T_p}{T_p + F_p} \quad R = \frac{T_p}{T_p + F_n} \quad F_1 = 2 \frac{P \cdot R}{P + R}. \quad (3.2)$$

We use FAST as the keypoint detector apart from Superpoint that outputs pairs of keypoints and descriptors directly for all experiments. For each dataset, we extracted the following descriptors: ORB, BRISK as binary and Superpoint and SURF as floating point. As retrieval methods, we use HBST with parameters  $\delta_{\max} = 0.1$  and  $N_{\max} = 50$  for the binary features. We use DBoW2 for all features but BRISK, which is not supported in by



**Figure 3.3.** Precision-Recall curves of the closures computed both with different combinations of feature extractors – image matchers on the LiDAR intensity image. Greater accuracy is reported in general by ORB-HBST, ORB-DBoW2 and Superpoint-DBoW2. Precision-Recall curves have been generated using different percentiles of query closures vector.



**Figure 3.4.** Localization timings of different combinations of feature extractors – image matchers. As expected binary descriptors take lower time to extract and match compared to floating points one.

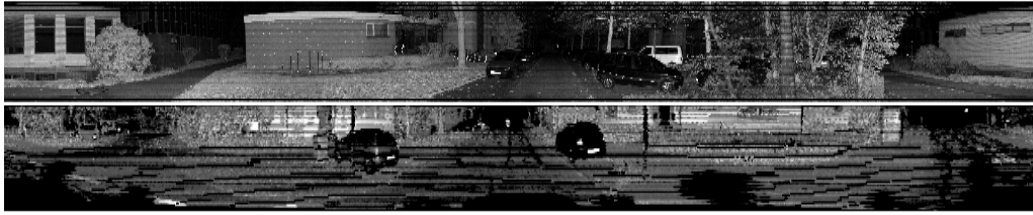
default package, and we extend it to operate with Superpoint. Configuration of each feature extractor reflects Tab. 3.1.

Since BoW approaches require a dictionary that depends on the sensor characteristics, we train such dictionaries using a portion of the datasets not used for the evaluation. In the plots, these curves are labeled with Voc-Train. More in detail, we train the vocabulary by using 3000 intensity images, a branching factor of 10, and a depth level of 5, using the classic euclidean distance to measure descriptor similarity over floating points and the Hamming distance for the binary ones. For comparison, we also report the results obtained with the image-based dictionaries packaged in the software release of DBoW2 (Voc-Main).

We conduct several experiments, varying the type of descriptor and VPR matcher. We report the precision-recall curves (Fig. 3.3), maximum harmonic mean (Tab. 3.3, Fig. 3.6) reached with LiDAR intensity images against normal camera images, localization timings (Fig. 3.4) and valid loops detected drawn on the trajectories (Fig. 3.1) of the most significant experiments.

Overall, existing VPR approaches shows usable results at negligible computation over the two most recent datasets (IPB Car and Newer College [98]) (Fig. 3.3 – Fig. 3.4). Results obtained in KITTI [44] and Ford Campus [91] are not comparable with the other two





**Figure 3.5.** Qualitative comparison between *IPB Car* LiDAR intensity image (up) and KITTI LiDAR intensity image (bottom) (both unprocessed). A lower vertical FoV and the uneven distribution of channels along the spinning axis makes KITTI intensity image unusable for this task.

Newer College [98]	IPB Car	Ford Campus [91]	KITTI [44]
0.6751	0.7088	0.115	0.097

**Table 3.3.** Max  $F_1$  score reached in full validation over the four datasets with combination of HBST [105] and ORB [101].

modern datasets. The HDL-64E utilized to record this data has irregularly distributed vertical laser beams. This is most likely due to a calibration offset that was not taken into consideration during the point cloud creation process (from raw spherical LiDAR measurements to Cartesian coordinates). This, along with the reduced vertical FoV (26.9 deg), results in two major issues:

- weakness to viewpoint invariancy throughout different vertical locations in the cylindrical image. The same item may seem different depending on the LiDAR orientation;
- many false positive features would be detected close to the empty gaps (black horizontal lines) due to radical change in intensity.

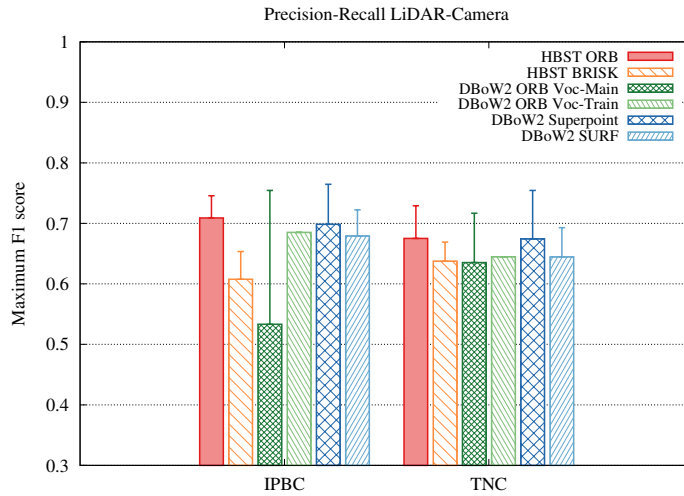
As a result, we were not able to produce acceptable outcomes for the two datasets (KITTI and Ford Campus) (Tab. 3.3), see also the qualitative comparison of the intensity images Fig. 3.5.

We obtain the best results by combining ORB with HBST and DBoW2. The combination Superpoint-DBoW2 shows a comparable performance. However, floating point descriptor comes with a higher computational cost, see Fig. 3.4. The accuracy on Newer College dataset [98] is inferior to the one obtained on our self-recorded dataset called IPB Car. This is due to the small changes on the roll axis since this data has been recorded walking in the campus, which, in turn, translates into a higher viewpoint variation within the same dataset.

For the experimental campaign, we used a PC running Ubuntu 20.04, equipped with an Intel i7-10750H CPU@2.60GHz and 16GB of RAM. We run neural network-based feature detection on a NVIDIA GeForce GTX 1650Ti.

## 3.4 Conclusion

In this chapter, we provide an analysis of the performance of visual place recognition techniques for loop closing, applied to the intensity information of a 3D LiDAR scanner.



**Figure 3.6.** Max  $F_1$  Score reached in full evaluation mode. Left our self-recorded dataset IPB Car and right Newer College [98]. The error bars indicate the  $F_1$  Score obtained using same approaches but over standard RGB images. Only DBoW2 ORB Voc-Train has not been replicated since the BoW training has been performed across LiDAR intensity images.

We evaluated gold standard visual place recognition approaches on four different datasets. Except for one outdated LiDAR, which does not provide stable intensity measurements, this transfer was proved to be successful and very close to results obtained with passive sensors (Fig. 3.6). On modern sensors with a high vertical resolution, we obtained encouraging results. Despite not proposing a new approach in this work, we believe that existing LiDAR-based mapping systems can easily benefit from our findings. We furthermore expect that one can improve the performance further by designing or learning descriptors that are specifically optimized for intensity cues of LiDAR scanners. In the following chapter, we will delve deeper into the capabilities of LiDAR imaging, utilizing both feature-based and photometric techniques. Throughout, we will approach the LiDAR as we would a camera.

## Chapter 4

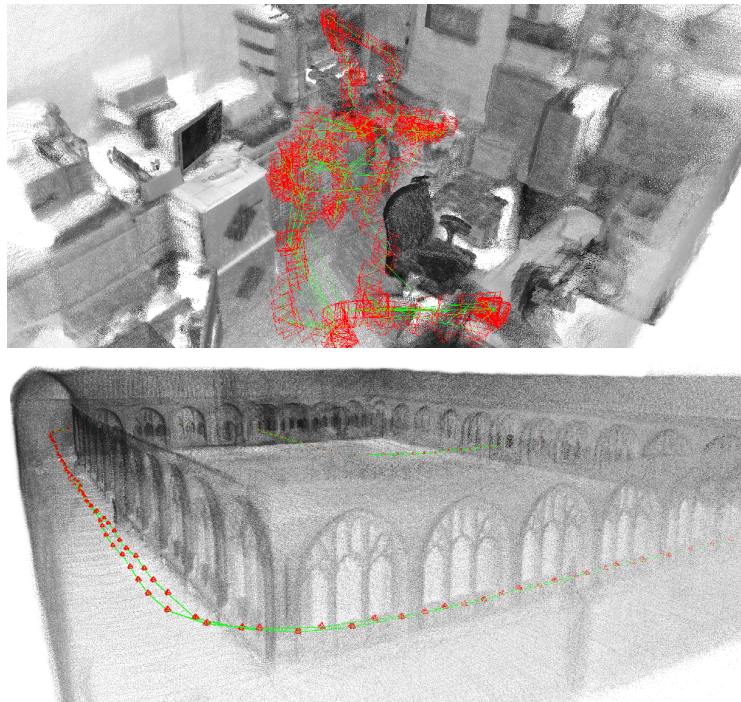
# MD-SLAM: Multi-cue Direct SLAM

SLAM is a popular field in robotics, and after roughly three decades of research, effective solutions are available. As many sectors rely on SLAM, such as autonomous driving, augmented reality and space exploration, it still receives much attention in academia and industry. The advent of robust machine learning systems allowed the community to enhance purely geometric maps with semantic information or replace hard-coded heuristics with data-driven ones. Within the computer vision community, we have seen direct (or photometric) approaches used to tackle the SLAM Structure from Motion (SfM) problem. As already mentioned in the introductory part, the direct techniques address registration by minimizing the pixel-wise error between image pairs or between an image and a model. 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 offer the possibility to boost registration accuracy [108]. Whereas these methods have been successfully used on monocular, stereo, or RGB-D images, their use on 3D LiDAR data is less prominent—probably due to the comparably limited vertical resolution relation to cameras.

Della Corte *et al.*[24] introduced a multi-cue photometric registration method for RGB-D cameras. This system expands photometric techniques to various projective models and improves robustness by including additional channels, like normals, in the cost function. As mentioned in previous chapters (Chapter 2 and Chapter 3), LiDAR provides not just range data but also information on intensity or reflectivity. Additionally, newer versions of LiDARs come with up to 128 vertical beams, enhancing the overall laser FoV. This development brings the image formed from the point cloud closer to what a traditional camera captures. In the previous chapter, we showed that classic optical features can be extracted and matched in LiDAR imaging; in this chapter, we present that direct registration methods can be effectively applied to LiDAR images.

The dominant paradigm for modern SLAM systems today is graph-based SLAM. A graph-based SLAM system works by constructing a SLAM graph where each node represents the sensor position or a landmark, while edges encode a relative displacement between nodes. Pose-graphs are a particular case in which only poses are stored in the graph. These local transformations stored in the edges are commonly inferred by comparing and matching sensor readings. This chapter investigates the fusion of multi-cue direct registration with graph-based SLAM.

The main contribution of this chapter is a flexible, direct SLAM pipeline for 3D data. To the best of our knowledge, our approach is the only SLAM system that can deal with



**Figure 4.1.** Scenes reconstructed using our pipeline. Top: results of a self-recorded dataset using Intel Realsense 455 RGB-D. Bottom: using LiDAR OS0-128 of the *cloister* sequence from the Newer College dataset [142].

RGB-D and LiDAR in a unified manner. Inspiring from [24], we adapted and revised the approach to compute the incremental motion for sensors that handle both RGB-D and LiDAR data. For the detection of loop-closures, we use an appearance-based algorithm that leverages a Binary Search Tree (BST) structure introduced by Schlegel *et al.* [105], filled with binary feature descriptors as described in [101]. All components that require the solution of an optimization problem rely on the same framework [49], resulting in a compact implementation. It is designed for flexibility, hence not optimizing the SLAM system to a specific sensor. Our system has been tested on both, RGB-D and LiDAR data, using benchmark datasets. The accuracy is competitive concerning other sensor-specific SLAM systems, while it outperforms them if some assumptions about the structure of the environment are violated. We release a CUDA/C++ implementation that runs in realtime for both sensors: <https://github.com/rvp-group/mdslam>. Fig. 4.1 illustrates some estimates of MD-SLAM.

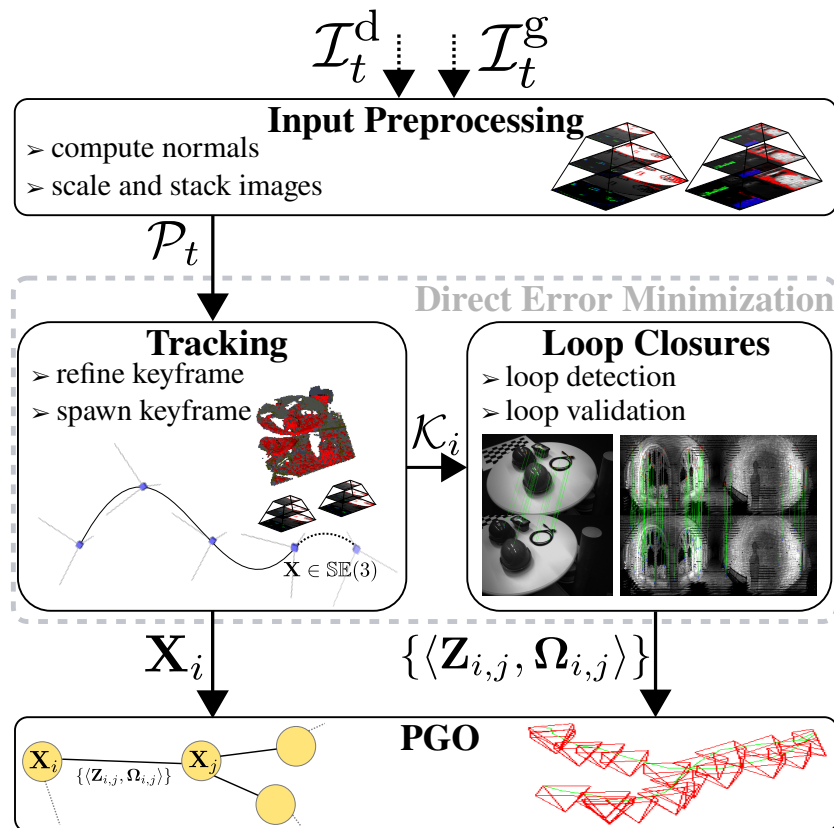
## 4.1 Related Work

3D SLAM has been widely addressed by the computer vision and robotics community and a large number of valid SLAM systems are available. Whereas many deserve mention, for compactness we focus only on the most seminal one, strictly related to our work. The available computational resources limited early approaches to operate offline [90] in fairly limited environments[134]. After the Kinect sensor became available about 15 years ago, we observed a revamped interest in RGB-D SLAM. Newcombe *et al.* [58] were

the first to leverage a dense tracking on a Truncated Signed Distance Function (TSDF) stored in the GPU while using massively parallel implementation to render the surface of the local scene perceived by the sensor. Meanwhile, Segal *et al.* [109] proposed a robust variant of Iterative Closest Point (ICP) relying on a point-to-plane metric. These initial methods addressed the open-loop registration approaches, tracking the pose of the sensor in a small neighbourhood. The advent of efficient optimization systems such as iSAM [62] and g2o [51], made it possible to build an effective full-fledged 3D SLAM system supporting loop closures and providing an online globally consistent estimate. Novel efficient salient floating point [12] and binary image descriptors [101], paired with bag-of-words retrieval methods inspired from web search engines, lead to impressive place recognition approaches [42]. These methods were then employed within visual SLAM systems, ORB-SLAM by Mur Artal *et al.* [86] being one of the most popular ones. The pipeline fully relied on the stability of features (keypoints), minimizing the reprojection error of the reconstructed landmarks within the image. In contrast to these indirect methods, another line of research aimed at photometric error minimization. Keller *et al.* [65] use projective data matching in a dense model, relying on a surfel-based map for tracking. Others rely on keyframe-based technique [66]. As it happened for feature-based approaches, these works were assembled into full visual SLAM systems [37]. More recently, BAD-SLAM, a surfel-based direct BA system that combines photometric and geometric error [108] using feature-based loop closures, shows that, for well-calibrated data, dense BA outperforms sparse BA. The accuracy and elegance shown by photometric approaches lead to further developments such as MPR [24] aiming at unifying both LiDAR and RGB-D devices into a unique registration method. Built maps can then be used for robot navigation [123], even in hazardous environments [119].

In parallel, the community approached LiDAR-based odometry by seeking alternative representations for the dense 3D point clouds. These include 3D salient features [140, 59], subsampled clouds [128] or NDT [118]. Nowadays, LiDAR Odometry and Mapping (LOAM) is perhaps one of the most popular methods for LiDAR odometry [140, 141]. 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 ground optimized version (Lego-LOAM) method has been later proposed [110], as it leverages the presence of a ground plane in its segmentation and optimization steps. In contrast to sparse methods, dense approaches suffer less in a non-structured environment [13]. Compared to RGB-D images, 3D LiDARs offer lower support for appearance-based place recognition. It is common for dense LiDAR SLAM systems to attempt a brute force registration with all neighborhood clouds to seek loop closures. Thanks to the typically small drift, this strategy is most successful; however, computational costs grow significantly in large environments. LiDAR loop closures have been addressed in different ways compared to RGB-D. Magnusson *et al.* proposed an approach suitable for NDT representations [82]. Röhling *et al.* [100] investigated the use of histograms computed directly from the 3D point cloud to define a measure of the similarity of two scans. Novel types of descriptors have been investigated, exploiting additional data gathered by the LiDAR sensor – i.e., light emission of the beams [23, 53]. However, despite being very attractive, these descriptors are time-consuming to extract and match, resulting in a slower system overall. Recent works address loop-closures detection in a RGB-D fashion, relying on the visual feature matching extracted from the image obtained by using the LiDAR intensity channel [32].

Building on top of prior work [24, 32], this paper presents a flexible and general SLAM



**Figure 4.2.** Illustration of our system. Depth  $\mathcal{I}_t^d$  and  $\mathcal{I}_t^g$  images are taken as input from the pipeline. An optimized trajectory within a map is produced as output. We do not spawn a keyframe  $i$  each processed images  $t$  (explained in Sec. 4.2.3). This system works independently both for RGB-D and LiDAR. Note that for LiDAR we take range image and not depth image as input.

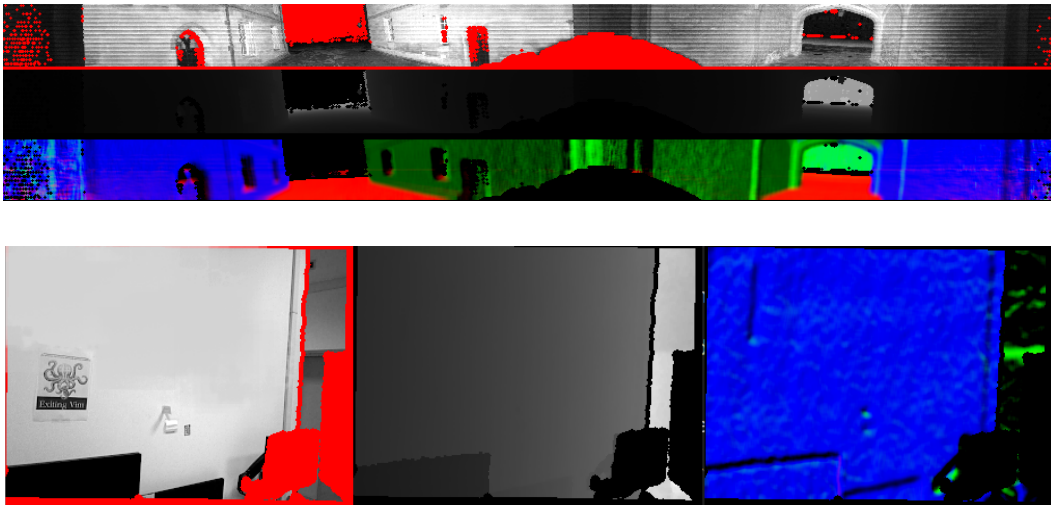
approach. It is a direct method working on RGB-D and 3D LiDAR data alike providing a unified approach. Our results show that it is competitive with other sensor-specific systems.

## 4.2 Multi-cue Direct SLAM

Our approach relies on a pose-graph to represent the map. Nodes of the pose-graph store keyframes in the form of multi-cue image pyramids. Our pipeline takes as input intensity (grayscale) and depth images for RGB-D or intensity and range images for LiDAR. For compactness, we will generalize, mentioning only depth images. The pyramids are generated from the inputs images each time a new frame becomes available. By processing the depth information, our system computes the surface normals and organizes them into a three-channel image, which is then stacked to the original input to form a five-channel image. Pyramids are generated by downscaling this input. This process is described in Sec. 4.2.1.

The pyramids are fed to the tracker, which is responsible for estimating the relative transform between the last keyframe and the current pyramid through the direct error minimization strategy summarized in Sec. 4.2.2. The tracker is in charge of spawning new keyframes and adding them to the graph when necessary, as discussed in Sec. 4.2.3.

Whenever a new keyframe is generated, the loop closure schema, described in Sec. 4.2.4,



**Figure 4.3.** Cues generated for LiDAR (top) and RGB-D (bottom) images. The first row/column shows the intensity  $\mathcal{I}^i$ , the middle shows the depth  $\mathcal{I}^d$ , and the last one illustrates the normals encoded by color  $\mathcal{I}^n$ . The red pixels on the intensity cues are invalid measurements (i.e., depth not available).

seeks for potential relocalization candidates between the past keyframes by performing a search in appearance space. Candidate matches are further pruned by geometric validation and direct refinement. Successful loop closures result in the addition of new constraints in the pose-graph and trigger a complete graph optimization as detailed in Sec. 4.2.5.

### 4.2.1 Input preprocessing

As discussed in the background chapter, specifically Sec. 2.2.4, images can represent multiple quantities. The input of our SLAM pipeline can be a pair of intensity/grayscale  $\mathcal{I}^g$  and depth  $\mathcal{I}^d$  (for RGB-D) or range  $\mathcal{I}^p$  (for LiDAR) images for each sensor pose. For simplicity we will discuss only about depth  $d$  which is suitable for RGB-D, but the same concept applies for LiDAR range  $\rho$ . 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 pixel  $\mathbf{u}$  we inverse project 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 depth at the pixel  $\mathcal{I}^d(\mathbf{u})$ . The normal  $\mathbf{n}_{\mathbf{u}}$  is the one of the plane that best fits the inverse projected points from the set  $\mathcal{U}$ , and is oriented towards the observer. All valid normals are assembled in a normal image  $\mathcal{I}^n$ , so that  $\mathcal{I}^n(\mathbf{u}) = \mathbf{n}_{\mathbf{u}}$ . Hence, the final five-channel image is obtained by stacking together  $\mathcal{I}^g$ ,  $\mathcal{I}^d$ , and  $\mathcal{I}^n$ . In the remainder, we will refer to the generic channel as a *cue*  $\mathcal{I}^c$ .

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 (top of Fig. 4.2 shows the “image pyramid structure”).

Each level of a pyramid consists of a multi-cue image generated from  $\mathcal{I}^g$ ,  $\mathcal{I}^d$  and  $\mathcal{I}^n$ , by



downscaling at user-selected resolutions. In our experiments, we typically use three scaling levels, each half the resolution of the previous level.

### 4.2.2 Photometric cost function

As in direct error minimization approaches, our method seeks to find the transform  $\mathbf{X}^* \in \mathbb{SE}(3)$  that minimizes the photometric distance between the two images:

$$\mathbf{X}^* = \operatorname{argmin}_{\mathbf{X} \in \mathbb{SE}(3)} \sum_{\mathbf{u}} L \left\| \underbrace{\mathcal{I}_{i+1}^g(\mathbf{u}) - \mathcal{I}_i^g \left( \underbrace{\pi(\mathbf{X}\pi^{-1}(\mathbf{u}, \rho))}_{\mathbf{u}'} \right)}_{\mathbf{e}_{\mathbf{u}}} \right\|^2 \quad (4.1)$$

where  $\mathbf{e}_{\mathbf{u}}^g$  denotes the error between corresponding pixels and  $L$  is the Huber robust loss. The evaluation point  $\mathbf{u}'$  of  $\mathcal{I}^g$  is computed by inverse projecting the pixel  $\mathbf{u}$ , applying the transform  $\mathbf{X}$  and projecting it back. To carry out this operation, the depth at the pixel  $d = \mathcal{I}^d(\mathbf{u})$  needs to be known, as explained in Sec. 2.2.

Eq. (6.2) models classical photometric error minimization assuming that the cues are not affected by the transform  $\mathbf{X}$ . In our case, depth and normal are affected by  $\mathbf{X}$ . Hence, we need to account for the change in these cues, and we will do it by introducing a mapping function  $\zeta^c(\mathbf{X}, \mathcal{I}^c(\mathbf{u}))$ . This function calculates the *pixel* value of the  $c^{\text{th}}$  cue after applying the transform  $\mathbf{X}$  to the original channel value  $\mathcal{I}^c(\mathbf{u})$ . We can thus rewrite a more general form of Eq. (4.1) that accounts for all cues and captures this effect as follows:

$$\mathbf{X}^* = \operatorname{argmin}_{\mathbf{X} \in \mathbb{SE}(3)} \sum_c \sum_{\mathbf{u}} L \left\| \underbrace{\zeta^c(\mathbf{X}, \mathcal{I}_{i+1}^c(\mathbf{u})) - \mathcal{I}_i^c(\mathbf{u}')}_{\mathbf{e}_{\mathbf{u}}^c} \right\|_{\Omega^c}^2 \quad (4.2)$$

The squared Mahalanobis distance  $\|\cdot\|_{\Omega^c}^2$  is used to weight the different cues (i.e. from noise). Further details (but not fundamental to understand this chapter) and analytic Jacobians will be provided when we introduce the photometric error minimization for Bundle Adjustment in Sec. 6.2.2. While the error may vary slightly, the concept remains similar.

While approaching the problem in Eq. (4.2) with the iteratively re-weighted optimization method described in Sec. 2.3, particular care has to be taken to the numerical approximations of floating-point numbers, as detailed in Sec. 2.3.5. In particular, since each pixel and cue contribute to constructing the quadratic form with an independent error  $\mathbf{e}_{\mathbf{u}}^c$ , the summations of  $\mathbf{H}$  and  $\mathbf{b}$  (Eq. (2.30), Eq. (2.31)) might accumulate millions of terms. Hence, to lessen the effect of these round-offs, the summation has to be computed using a stable algorithm, either in GPU or for single-thread implementations Sec. 2.3.5.

We use multi-cue direct alignment in incremental position tracking, explained in next section (Sec. 4.2.3) and in loop closure refinement and validation (Sec. 4.2.4). Eq. (4.1) and Eq. (4.2) emphasize the incremental nature of the problem with indices  $\langle i, i+1 \rangle$ , which are apt for tracking/odometry estimation. However, for loop validation, this approach is not applicable, and it is more appropriate to use image indices  $\langle i, j \rangle$  as mentioned in Sec. 4.2.4.

### 4.2.3 Tracking

This module is in charge of estimating the open-loop trajectory of the sensor. To this extent, it processes new pyramids as they become available by determining the relative transform between the last pyramid  $\mathcal{P}_t$ , and the current keyframe  $\mathcal{K}_i$ . A keyframe stores a



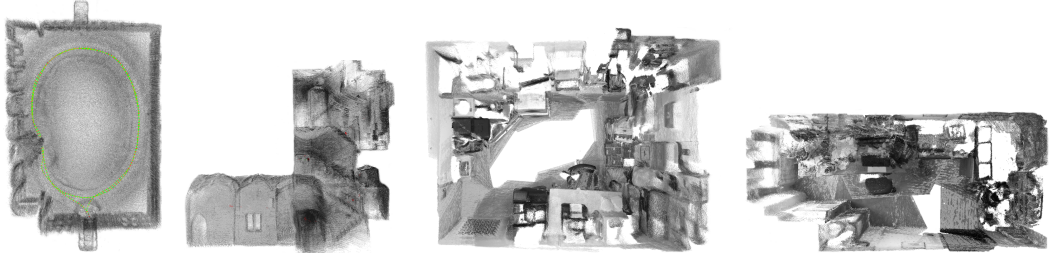
global transform  $\mathbf{X}_i$ , and a pyramid  $\mathcal{P}_i$ . The registration algorithm of Sec. 4.2.2 is used to compute a relative transform  $\mathbf{Z}_{i,t}$  between the last two pyramids. Whenever the magnitude of such a transform exceeds a given threshold or the overlap between  $\mathcal{P}_i$  and  $\mathcal{P}_t$  becomes too small, the tracker spawns a new keyframe  $\mathcal{K}_{i+1}$ , with transform  $\mathbf{X}_{i+1} = \mathbf{X}_i \mathbf{Z}_{i,i+1}$ . Furthermore, it adds to the graph a new constraint between the nodes  $i$  and  $i + 1$ , with transform  $\mathbf{Z}_{i,i+1}$ , and information matrix  $\mathbf{\Omega}_{i,i+1}$ . The latter is set to  $\mathbf{H}$  matrix of the direct registration at the optimum. The generation of the new keyframe triggers the loop detection described in the next section. Using keyframes reduces the drift that would occur when performing subsequent pairwise registration since the reference frame stays fixed for a longer time. Potentially, if the sensor hovers at a distance smaller than the keyframe threshold, all registrations are done against the same pyramid, and no drift would occur.

#### 4.2.4 Loop detection and validation

This module is responsible for relocalizing a newly generated keyframe with respect to previous ones. More formally, given a query frame  $\mathcal{K}_i$ , it retrieves a set of tuples  $\{\langle \mathcal{K}_j, \mathbf{Z}_{i,j}, \mathbf{\Omega}_{i,j} \rangle\}$ , consisting of a past keyframe  $\mathcal{K}_j$ , a transform  $\mathbf{Z}_{i,j}$  between  $\mathcal{K}_i$  and  $\mathcal{K}_j$  and an information matrix  $\mathbf{\Omega}_{i,j}$  characterizing the uncertainty of the computed transform. Our system approaches loop closing in multiple stages. At first, we carry on visual place recognition on the intensity channels. This approach leverages the results of previous work [32]. For visual place recognition, we rely on ORB feature descriptors, extracted from the  $\mathcal{I}^1$  of each keyframe. Retrieving the most similar frame to the current one results in looking for the images in the database having the closest descriptor “close” to the one of the current image. To efficiently conduct this search, we use a hamming distance embedding binary search tree (HBST) [105], a tree-like structure that allows for descriptor search and insertion in logarithmic time by exploiting particular properties of binary descriptors. A match from HBST also returns a set of pairs of corresponding points between the matching keypoints. Having the depth and inverse projecting the points, we can carry on a straightforward RANSAC registration. Finally, each candidate match is subject to direct refinement (Sec. 4.2.2). This step enhances the accuracy and it provides information matrices on the same scale as the ones generated by the tracker. The above strategy is applied independently to RGB-D or LiDAR data. These surviving pairs  $\{\langle \mathcal{K}_j, \mathbf{Z}_{i,j}, \mathbf{\Omega}_{i,j} \rangle\}$ , constitute potential loop closing constraints to be added to the graph. However, to handle environments with large sensor aliasing, we introduced a further check to preserve topological consistency. Whenever a loop closure is found, we carry on a direct registration between all neighbors that would result *after* accepting the closure. If the resulting error is within certain bounds, the closure is finally added to the graph, and a global optimization is triggered.

#### 4.2.5 Pose-graph optimization

The goal of this module is to retrieve a configuration of the keyframes in the space that is maximally consistent with the incremental constraints introduced by the tracker and the loop closing constraints by the loop detector. A pose-graph is a special case of a factor-graph, presented in the preliminaries (Sec. 2.3.4). The nodes of the graph are the keyframe poses  $\mathbf{X} = \{\mathbf{X}_i\}_{i=1:N}$ , while the constraints encode the relative transformations between the connected keyframes, together with their uncertainty  $\{\langle \mathbf{Z}_{i,j}, \mathbf{\Omega}_{i,j} \rangle\}$ . Optimizing



**Figure 4.4.** Some scenes from Newer College dataset (LiDAR) - right, and self-recording data (RGB-D) - left, reconstructed with MD-SLAM.

RGB (480x640)		LiDAR (128x1024)	
CPU	GPU	CPU	GPU
8	30	15	60

**Table 4.1.** MD-SLAM runtimes expressed in Hz.

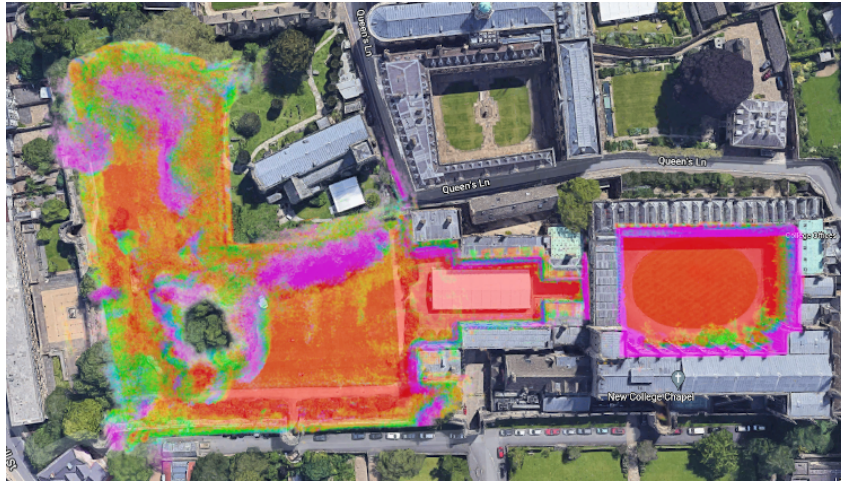
a factor-graph consists in solving the following optimization problem:

$$\mathbf{X}^* = \underset{\mathbf{X} \in \mathbb{S}\mathbb{E}(3)^N}{\operatorname{argmin}} \sum_{i,j} \underbrace{\| \mathbf{X}_i^{-1} \mathbf{X}_j \boxminus \mathbf{Z}_{i,j} \|_{\Omega_{i,j}}^2}_{\mathbf{e}_{i,j}} \quad (4.3)$$

Here, the error  $\mathbf{e}_{i,j}$  is the difference between predicted displacement  $\mathbf{X}_i^{-1} \mathbf{X}_j$  and result of the direct alignment  $\mathbf{Z}_{i,j}$ . The total perturbation vector  $\Delta \mathbf{x} \in \mathbb{R}^{6N}$  results from stacking all variable perturbations  $\{\Delta \mathbf{x}_i\}$ . For the reasons illustrated in Sec. 2.3.1, for pose-graph optimization we employ Gauss-Newton.

### 4.3 Experimental Evaluation

In this section, we report the results of our pipeline on different public benchmark datasets. To the best of our knowledge, our approach is the only open-source SLAM system that can deal with RGB-D and LiDAR in a unified manner. Therefore, to evaluate our system, we compare with state-of-the-art SLAM packages developed specifically for each of these sensor types. For RGB-D we consider DVO-SLAM [66] and ElasticFusion [133] as direct approaches and ORB-SLAM2 [86] as indirect representative. For LiDAR we compare against LeGO-LOAM [110] as feature-based and SuMA [13] representing the dense category. We specify that our pipeline is purely photometric namely, not IMU or odometry source have been used as assistance. To run the experiments, we used a PC with an Intel Core i9-10900KF CPU @ 5.30GHz with 64GB of RAM and a RTX 3070. Since this work is focused on SLAM, we perform our quantitative evaluation using the RMSE on the absolute trajectory error (ATE) with  $\mathbb{S}\mathbb{E}(3)$  alignment. The alignment for the metric is computed by using the Horn method [56], and the timestamps are used to determine the associations. Then, we calculate the RMSE of the translational differences between all matched poses. The tracking module dominates the runtime of our approach since loop closures are detected and validated asynchronously within another thread. Hence, we report the average frequency at which the tracker runs for each sensor. At the core of the tracker, we have the photometric



**Figure 4.5.** MD-SLAM map on *long* sequence from Newer College [98] aligned with Google Earth.

registration algorithm, whose computation is proportional to the size of the images. Runtimes summarized in Tab. 4.1.

### 4.3.1 RGB-D

We conducted several experiments with RGB-D sensor. Qualitative analysis have been done using self-recorded data and are shown in Fig. 4.4. As public benchmarks we used the TUM-RGB-D [121] and the ETH3D [108]. The TUM RGB-D dataset contains multiple real datasets captured with handheld Xbox Kinect. A rolling shutter camera provides RGB data. Further, the camera’s depth and color streams are not synchronized. Every sequence accompanies an accurate ground truth trajectory obtained with an external motion capture system. ETH3D benchmark is acquired with global shutter cameras and accurate active stereo depth. Color and depth images are synchronized. We select several indoor sequences for which ground truth, computed by external motion capture, is available.

On these datasets, we compare with DVO-SLAM, ElasticFusion and ORB-SLAM2. These three approaches are representative of different classes of SLAM algorithms. Tab. 4.2 shows the results on the TUM RGB-D datasets, while Tab. 4.3 presents the outcome on the ETH3D datasets. DVO SLAM implements a mixed geometry-based and direct registration. Internally the alignment between pairs of keyframes 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. In contrast to these two 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. This map is constantly optimized as the camera moves by performing local BA. Loop closures are detected through DBoW2 [42] and a global optimization on a Sim(3) pose-graph to enforce global consistency is used.

The TUM dataset provides images  $640 \times 480$  pixels, while ETH3D  $740 \times 460$  pixels. From these images, we compute a 3 level pyramid with scales  $1/2$ ,  $1/4$ , and  $1/8$ . Our system runs respectively at 8 and 7.5 Hz at these resolutions on CPU. On GPU, differently, operates at 30 Hz and 28.5 Hz, making the graphic-card implementation suitable for online estimation.

	fr1/desk	fr1/desk2	fr2/desk
DVO-SLAM [66]	0.021	0.046	0.017
ElasticFusion [133]	0.020	0.048	0.071
ORB-SLAM2 [86]	<b>0.016</b>	<b>0.022</b>	<b>0.009</b>
<b>Ours</b>	0.041	0.064	0.057

**Table 4.2.** ATE RMSE [m] results on TUM RGB-D [121] datasets. This was recorded with non-synchronous depth using a rolling shutter camera.

	table3	table4	table7	cables1	plant2	planar2
DVO-SLAM [66]	0.008	0.018	<b>0.007</b>	<b>0.004</b>	0.002	0.002
ElasticFusion [133]	–	0.012	–	0.018	0.017	0.011
ORB-SLAM2 [86]	<b>0.007</b>	<b>0.008</b>	0.010	0.007	0.003	0.005
<b>Ours</b>	0.021	0.022	0.036	0.015	<b>0.001</b>	<b>0.001</b>

**Table 4.3.** ATE RMSE [m] on ETH3D [108]. This was recorded with global shutter camera and synchronous streams. ElasticFusion fails in *table3* and *table7*.

In Tab. 4.2 we can see that ORB-SLAM2 clearly outperforms all other pipelines. DVO-SLAM and ElasticFusion provide comparable results, and our approach is the worst in terms of accuracy. Yet, the largest error is 6.4 cm, which results in a usable map. As stated before, this dataset is subject to rolling shutter and asynchronous depth effects. ORB-SLAM2, being feature-based, is less sensitive to these phenomena. DVO-SLAM and ElasticFusion explicitly model these effects. Our approach does not attempt to address these issues since it would render the whole pipeline less consistent between different sensing modalities.

Tab. 4.3 presents the results on the ETH3D benchmark. In this case, our performances are on par with other methods, since intensity and depth are synchronous, and the camera is global shutter.

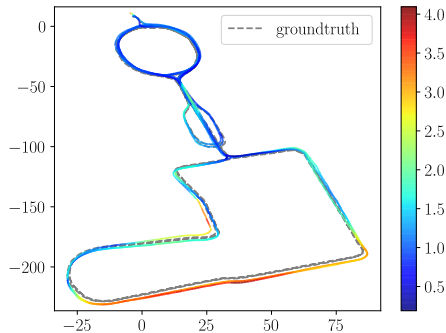
These results highlight the strength and weaknesses of a purely direct approach not supported by any geometric association. While being compact, it suffers from unmodeled effects and requires a considerable overlap between subsequent frames.

### 4.3.2 3D LiDAR

We conducted different experiments on public LiDAR benchmarks to show the performances of our SLAM implementation. For the LiDAR we use the Newer College Dataset [98, 142] recorded at 10 Hz with two models of Ouster LiDARs: OS1 and OS0. We conducted our evaluation on the *long*, *cloister*, *quad-easy* and *stairs* sequences. The OS1 has 64 vertical beams. We selected the *long* sequence that lasts approximately 45 minutes. It consists of multiple loops with viewpoint changes between buildings and a park.

The other three shorter sequences are recorded with the OS0, which has 128 vertical beams. The LiDAR *quad-easy* sequence contains four loops that explore quad, *cloister* mixes outdoor and indoor scenes while *stairs* is purely indoor and based on vertical motion through different floors.

Qualitative analysis have been performed to show the results obtained by our pipeline. Fig. 4.4 illustrates some reconstructions obtained with MD-SLAM from Newer College



**Figure 4.6.** Alignment of our estimate with the ground truth in *long* sequence of Newer College. The color bar on the right shows the translational error [m] over the whole trajectory.

	OS0-128		OS1-64	
	cloister	quad	stairs	long
LeGO-LOAM [110]	<b>0.20</b>	<b>0.09</b>	3.20	<b>1.30</b>
SuMA [13]	3.34	1.74	0.67	-
<b>Ours</b>	0.36	0.25	<b>0.34</b>	1.74

**Table 4.4.** ATE RMSE [m] results of all benchmarked approaches on the Newer College datasets [98, 142]. SuMA fails on *long* sequence.

sequences. Fig. 4.5 and Fig. 4.6 show the global consistency of our estimate on *long* sequence.

Quantitatively, we compare against LeGO-LOAM and SuMA. These represent two different classes of LiDAR algorithms, respectively sparse and dense. Tab. 4.4 summarizes the results of the comparison. LeGO-LOAM is currently one of the most accurate LiDAR SLAM pipelines and represents a sparse class of LiDAR algorithms. In contrast to our approach, LeGO-LOAM is a pure geometric feature-based frame-to-model LiDAR SLAM work, 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 in a frame-to-frame fashion. For both the pipelines loop closures are handled through ICP. Being ground optimized, LeGO-LOAM shows impressive results mainly in chunks where ground occupies most of the scene, yet our approach provides competitive accuracy. The situation becomes challenging for LeGO-LOAM when its assumptions are violated, such as in the *stairs* sequence. In this case, our pipeline is the most accurate since it does not impose any particular structure on the environment being mapped. SuMA performances are the worst in terms of accuracy. We tried this pipeline both in a frame-to-frame and frame-to-model mode. The one reported in Tab. 4.4 represents SuMA frame-to-frame that always outperforms the frame-to-model on these datasets.

We use the OS1 to produce images of  $64 \times 1024$  pixels while the OS0 to produce images of  $128 \times 1024$  pixels. Since the horizontal resolution is much larger than the vertical one, to balance the aspect ratio for direct registration, initially, we downscale the horizontal

resolution by  $1/2$  for OS0 and by  $1/4$  for OS1. Our approach generates a pyramid with the following scales: 1,  $1/2$  and  $1/4$ . With these settings, our system operates in CPU at around 15 Hz on the OS0 and at approximately 24 Hz on the OS1. In GPU respectively at 60 Hz and 120 Hz. This makes both implementation suitable for online estimation.

## 4.4 Conclusion

In this chapter, we presented a direct SLAM system that operates both with RGB-D and LiDAR sensors. These two heterogeneous sensor modalities are addressed exclusively by changing the projection models. To the best of our knowledge, our approach is the only SLAM system that can deal with RGB-D and LiDAR in a unified manner. Furthermore, thanks to the data independence given by photometric nature, our implementation runs in realtime in commercial GPUs. Comparative experiments show that our generic method can compete with sensor-specific state-of-the-art approaches. Being purely photometric and without making any assumption of the environment, our pipeline shows consistent results on different types of datasets. The independence of the internal representation from the sensor source paves the way to SLAM systems that operate jointly on both RGB-D and LiDAR, within the same algorithm. In Chapter 6, we'll explore a universal BA approach that remains consistent regardless of the sensor type, functioning uniformly for both LiDAR and camera, whether used separately or together. Such integration hinges on the precise and versatile calibration framework presented in Chapter 5.

## Chapter 5

# Ca<sup>2</sup>Lib: Simple and Accurate LiDAR-RGB Calibration using a Small Common Marker

As established in previous chapters, the ability to fuse readings from heterogeneous sensors is often beneficial in many robotics and perception applications. In particular, LiDAR and RGB sensors exhibit a strong compatibility: the former being able to capture high precision sparse range readings while the latter measure dense color intensity measurements. Recent advancements in sensor technology have enabled the fusion of LiDAR and RGB data for enhanced 3D reconstruction. Several researchers have proposed pipelines that leverage the depth information from LiDAR and the rich color details from RGB cameras to produce high-fidelity 3D models. In the work of Smith *et al.* [113], a novel approach was introduced that seamlessly integrates depth maps from LiDAR with RGB images. Their method employs a deep learning framework to refine the depth maps, ensuring consistency with the RGB data. Another contribution by Johnson *et al.* [60] presents a hybrid pipeline that uses both LiDAR point clouds and RGB images for dense 3D reconstruction. Their method focuses on aligning the LiDAR and RGB data in a common reference frame and then fusing them using a voxel-based approach. Others, simply limit fusing sensors for depth estimation tasks, taking advantage of a more accurate and robust LiDAR depth (see Sec. 2.2.2). However, to accomplish any of these tasks, and merge the strengths from both sensors, one would require to accurately know the relative offset between them.

In this chapter, we introduce a straightforward calibration schema designed to precisely estimate the relative displacement between two sensors. Central to this multimodal calibration is the identification of spatial correspondences between the heterogeneous measurements. Most approaches rely on one or more calibration patterns to establish common features between the sensors. The main problem is that these patterns are often complex or expensive to produce [14].

Our primary contribution is the development of a flexible calibration toolbox. This toolbox facilitates the estimation of the extrinsic parameters between LiDAR and RGB using a basic commercial calibration target (e.g. Checkerboard, ChAruCO [43]), requiring minimal user intervention. We leverage a joint non-linear formulation of the problem to achieve high accuracy results even with a minimum of three measurements. The only requirement for our method is that measurements must be observed by both sensors during the acquisition.





**Figure 5.1.** Camera/LiDAR qualitative calibration results. The image shows the reprojection of a LiDAR point cloud on a image captured by a fisheye RGB camera rigidly attached to the former. The offset between the sensors leads to shadows on parts of the image.

We exploit the planarity of the target to find a common observation used to estimate the extrinsic parameters. Moreover, we release an open-source implementation of our toolbox at <https://github.com/rvp-group/ca2lib>.

## 5.1 Related Work

This section delves into LiDAR-RGB calibration and explores the two main classes of approaches: *target-based* and *target-less*. 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 use from this task. The core idea of calibration is common in the two classes of approaches: computing common features between heterogeneous measurements and estimating the transformation that minimizes the distance between corresponding features.

First, an overview of *target-less* approaches is presented: Pandey *et al.* presents an automatic data-driven approach based upon the maximization of mutual information between the sensor-measured surface intensities [92]. The authors exploits the correlation coefficient for the reflectivity and intensity values of many scan-image pairs using different calibration parameters. However, shadows of objects or colored surfaces that completely absorb infrared lights might result in weaker correlation between scan-image pairs. Yoon *et al.* proposes a calibration method using region-based object pose estimation. Objects are segmented in both measurements, then a 3D mesh is generated from the LiDAR measurements, while images are used to reconstruct the scene using 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 [137]. In recent years, the development of learning based methods have also spanned in this field: Lv *et al.* proposes a real-time self-calibration network that predict the extrinsic parameters by constructing a cost volume between RGB and LiDAR features [80], while Sun *et al.* first estimates an initial guess by solving an hand-eye calibration method [122].



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

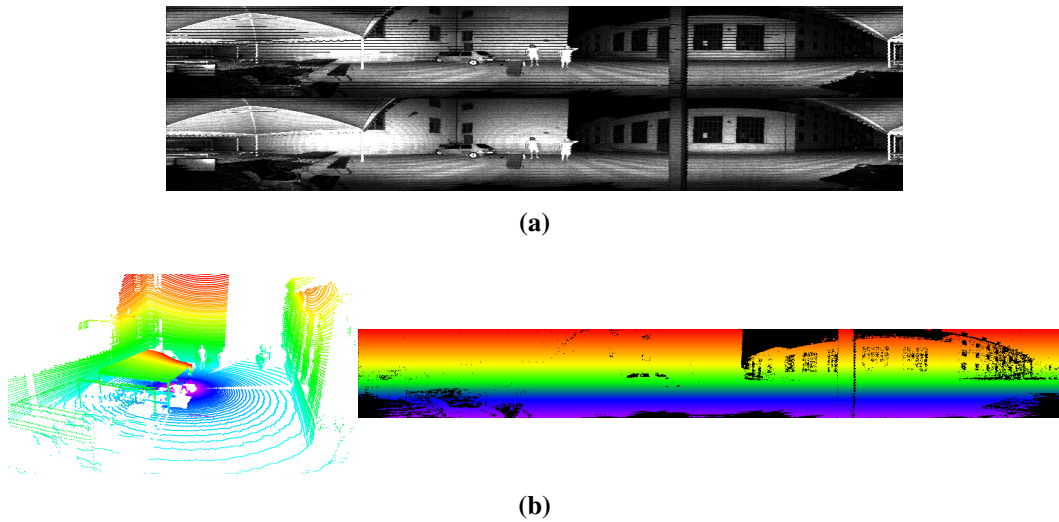
*Target-based* methods estimate the relative pose using an observed known structure. Given the difference of resolution for the two sensors, it is high unlikely that correspondences within the measurements can be established directly. For this reason, *point-to-point* methods tends to process LiDAR measurements to implicitly obtain virtual points<sup>1</sup> easily detectable from an RGB sensor. For instance, Park *et al.* utilizes a specially designed polygonal planar calibration board with known lengths of adjacent sides [93]. 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.* introduces a methodology that utilizes cubic objects with predetermined side lengths [95, 94]. The corners of the cubes are estimated by initially detecting each side of the box and subsequently determining their intersection points. Furthermore, the corners along with their corresponding RGB image are employed to calibrate the system by solving ICP. Zhou *et al.* proposes a single-shot calibration method requiring a checkerboard [144]. The target is detected both in the RGB image, and LiDAR measurement, using RANSAC [40] for the latter. Furthermore, the four edges of the checkerboard are estimated and aligned to compute the relative offset between the two sensors. 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 [125]. Upon successful detection, the algorithm aligns the set of sphere centers using SVD. Beltrán *et al.* presents a methodology that utilizes a custom calibration target equipped with 4 holes and AruCO markers specifically designed for monocular detection [14]. The methodology employs a set of techniques for each sensor to estimate the center points of the holes. Subsequently, the relative offset between sensors are determined by aligning the set of centers obtained from each sensor, Li *et al.* adopt a similar approach while using a checkerboard with 4 holes [73]. Fan *et al.* propose a two-stage calibration method using an auxiliary device with distinctive geometric features [39]. 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.*, the authors first extract planar information from RGB and LiDAR measurements, then, two grid of points are extracted from the computed planar patches and aligned using a customized ICP algorithm [111].

Albeit these approaches provides relatively 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 requires precise construction or expensive manufacturing.

Another group of approaches does not directly solve the calibration problem using point-to-point correspondences, but rather exploit the planarity of the target to reduce the 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 [85]. The authors then refine these estimates through iterative minimization. They also discuss the identifiability

---

<sup>1</sup>Points that are not explicitly detected, but estimated from the LiDAR measurement.



**Figure 5.2.** LiDAR image generation for calibration. (a) shows a comparison between the spherical projection (top) used for LiDAR images and the projection by ID (bottom) used for this work (as already shown in Fig. 2.8). (b) shows the ring information before (left) and after (right) the projection.

conditions for accurate parameter estimation. Finally, a method similar to our proposal, Kim *et al.* combine observed normals to first estimate the relative orientation with SVD and then iteratively estimates an initial guess of the relative translation by minimizing the pairwise planar distances between measurements [67]. Finally, the translation is refined using a non-linear optimization problem using LM. Despite its simplicity, this method decouples the estimation of orientation and translation, thus leading to potential losses in accuracy while also increasing the number of required measurements.

Compared with the state-of-the-art, 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 supports Checkerboard and *ChARuCO* patterns of typical A3-A4 sizes, easily obtainable from commercial printers;
- the possibility to handle different camera models and distortion;
- an open-source C++ implementation.

## 5.2 LiDAR-RGB Calibration

In this section, we will provide a detailed and comprehensive description of our method. First we describe the preliminaries required to understand our approach, then every component of the pipeline is described, following the procedure from the acquisition of the measurements up to the computation of the relative poses between the two sensors (extrinsic parameter). Note that the background to understand this sections has been presented in Chapter 2, here



Figure 5.3. Qualitative samples showing LiDAR cloud projection on RGB image.

we briefly re-adapt some of the notation to this specific problem. **Plane Representation:** Let  $\gamma = (\hat{\mathbf{n}}, d)$  be a 3D plane, where  $\hat{\mathbf{n}} \in \mathbb{S}^2$  represents the unit vector orthogonal to the plane and  $d \in \mathbb{R}$  is the shortest distance of the plane respect to the origin. Applying a transform  $\mathbf{X} \in \mathbb{SE}(3)$  to a plane  $\gamma$  yields new coefficients  $\gamma'$ :

$$\mathbf{X}\gamma = \begin{cases} \hat{\mathbf{n}}' = \mathbf{R}\mathbf{n} \\ d' = d + (\mathbf{R}\mathbf{n})^T \mathbf{t} \end{cases} \quad (5.1)$$

Since the transformation is modified by a local perturbation in the tangent space  $\Delta \mathbf{x} \in \mathbb{R}^6$ , we can rewrite:

$$(\mathbf{X} \boxplus \Delta \mathbf{X})\gamma = \begin{cases} \tilde{\mathbf{n}} = \Delta \mathbf{R} \mathbf{R} \mathbf{n} \\ \tilde{d} = d' + \mathbf{n}^T \mathbf{R}^T \Delta \mathbf{R}^T \Delta \mathbf{t} \end{cases} \quad (5.2)$$

Deriving the result with respect to  $\Delta \mathbf{x}$  leads to the following Jacobian:

$$\frac{\partial (\mathbf{X} \boxplus \Delta \mathbf{x})\gamma}{\partial \Delta \mathbf{x}} = \begin{bmatrix} \mathbf{0}_{3 \times 3} & -[\mathbf{R}\mathbf{n}]_{\times} \\ \mathbf{n}^T \mathbf{R}^T & \mathbf{0}_{1 \times 3} \end{bmatrix}_{4 \times 6} \quad (5.3)$$

The distance between two planes depends both 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:

$$\mathbf{p}(\gamma_k) = -\mathbf{n}_k d_k \quad (5.4)$$

$$\mathbf{e}_p(\gamma_i, \gamma_j) = \begin{bmatrix} \mathbf{n}_i^T (\mathbf{p}(\gamma_i) - \mathbf{p}(\gamma_j)) \\ \mathbf{n}_j - \mathbf{n}_i \end{bmatrix} \quad (5.5)$$

Here  $\mathbf{p}(\gamma_k)$  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 distance  $d$ . In all the works presented in this thesis, we mostly used spherical projection (Sec. 2.2.3), however, during the calibration process, our primary concern in representing the LiDAR image is to allow the user to precisely select the checkerboard plane. Thus we generate the LiDAR image through projection by ID, more details in Sec. 2.2.3

We now explain the steps done in order to achieve a reliable calibration. First, we process the incoming raw LiDAR and RGB measurements to acquire planar information. Assuming the scene to remain static throughout the acquisition of a single joint measurement, the LiDAR cloud is embedded in an image using the projection by ID. 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 attempted on the RGB image. Once the target is detected, the RGB plane is computed by solving the ICP. If the user is satisfied with both LiDAR and RGB planes, they are stored for processing.

Whereas a straightforward rank analysis of the Jacobians reveals that just 3 measurements are sufficient to constrain a solution, it is well known from the estimation theory that the accuracy grows with “good” number of measurements.

Once the set of measurements are acquired, we jointly estimate the relative orientation and translation of the LiDAR with respect to the RGB sensor  $\mathbf{X} \in \mathbb{SE}(3)$  by solving the following nonlinear minimization problem:

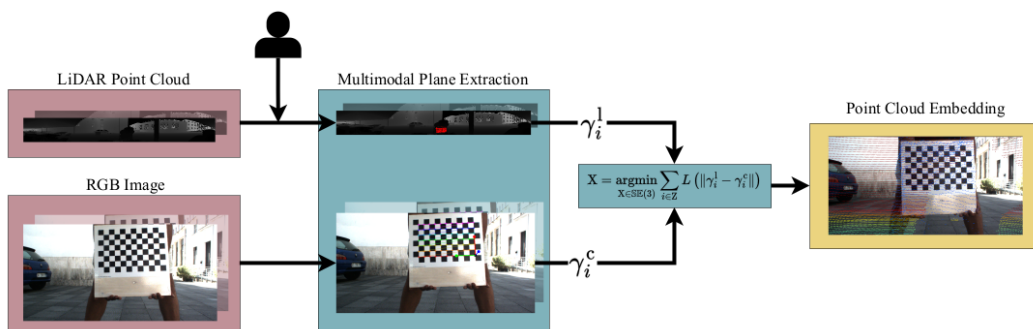
$$\mathbf{X} = \underset{\mathbf{X} \in \mathbb{SE}(3)}{\operatorname{argmin}} \sum_i \underbrace{\|\mathbf{X}\gamma_i^l - \gamma_i^c\|}_{e_p}^2 \quad (5.6)$$

where  $e_p$  represent the plane-to-plane error.

During acquisition, it may happen that the user accepts one or more wrongly estimated measurements. Due to the quadratic nature of the error terms, these *outliers* are often over-accounted, resulting in wrong estimations. To account for this factor, as described in [50], we employ an Huber robust estimator  $L$  that treats differently measurements based on their error. We rewrite Eq. (5.6) as follows:

$$\mathbf{X} = \underset{\mathbf{X} \in \mathbb{SE}(3)}{\operatorname{argmin}} \sum_i L(\|\mathbf{X}\gamma_i^l - \gamma_i^c\|). \quad (5.7)$$

To resolve Eq. (5.7) we employ the GN algorithm implemented in the `srrg2_solver` [50].



**Figure 5.4.** 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.

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

**Table 5.1.** Average translation error in millimeters with different noise levels and number of measurements ( $N$ ).

## 5.3 Experimental Evaluation

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 [14] to compare our estimates with ground truth while we also conduct qualitative and quantitative experiments on real scenarios using our acquisition system. We directly compare our results with [67], as it is the work which is closest to ours. In addition, we compare to [14] that produce accurate results relying on a very complex target (CNC printed).

### 5.3.1 Synthetic case

We conducted experiments on *Gazebo* simulator [70] to evaluate the accuracy and robustness of our approach, injecting different noise figures to the sensor measurements. We also experiment how the number of observations affect the final results. The setup of the scene includes a Velodyne HDL-64 LiDAR, a BlackFly-S RGB sensor and a  $6 \times 8$  checkerboard target with corner size of 0.2 meters. We randomly generate and acquire 53 valid<sup>2</sup> 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  $\sigma_L$  (0 mm, 7 mm and 14 mm). For every  $w_s$ , we sample 40 sets of measurements.

From Tab. 5.1, we observe a steady decrease of error for every noise level, reaching an average of 2.6mm translation error in the intermediate noise case. In case of 3 measurements the high uncertainty is due to the potentially poorly conditioned system when using planes that have similar normals. Nonetheless, we compare our best result with 3 measurements against the best results of the methods presented in [14] and [67]. Tab. 5.2 shows the results.

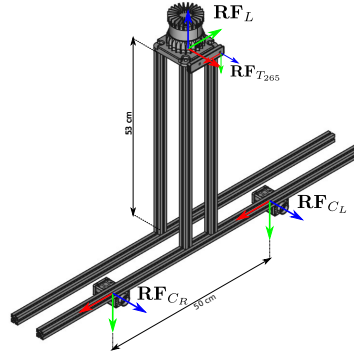
### 5.3.2 Real case

In this section, we describe the experiments conducted on real measurements. We perform a quantitative test on our acquisition system shown in Fig. 5.5 that is equipped with a Ouster

<sup>2</sup>A valid measurement is one for which both LiDAR and RGB sensor are able to detect the target

Method	$e_t$ (cm)	$e_r$ ( $10^{-2}$ rad)
Beltrán <i>et al.</i> [14]	0.82	<b>0.24</b>
Kim <i>et al.</i> [67]	10.2	129.56
<b>Ours</b>	<b>0.11</b>	0.25

**Table 5.2.** Quantitative results on synthetic data achieved through calibration using  $N = 3$  measurements. We choose this measurement count for parity with the methodology proposed in [14]. In [14], a single measurement is deemed sufficient for calibration determination, with 3 measurements considered the optimal scenario. Beyond 3 measurements accuracy does not improve. Both for our study and in alignment with the findings in [67], 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 [14] in rotation, while outperforming all methods on translation, using small commercial tags.



**Figure 5.5.** Acquisition system used for the real case experiments.

OS0-128 LiDAR with a resolution of  $128 \times 1024$ , a RealSense T-265 stereo camera and two Manta-G145 RGB cameras arranged in a wide horizontal stereo configuration.

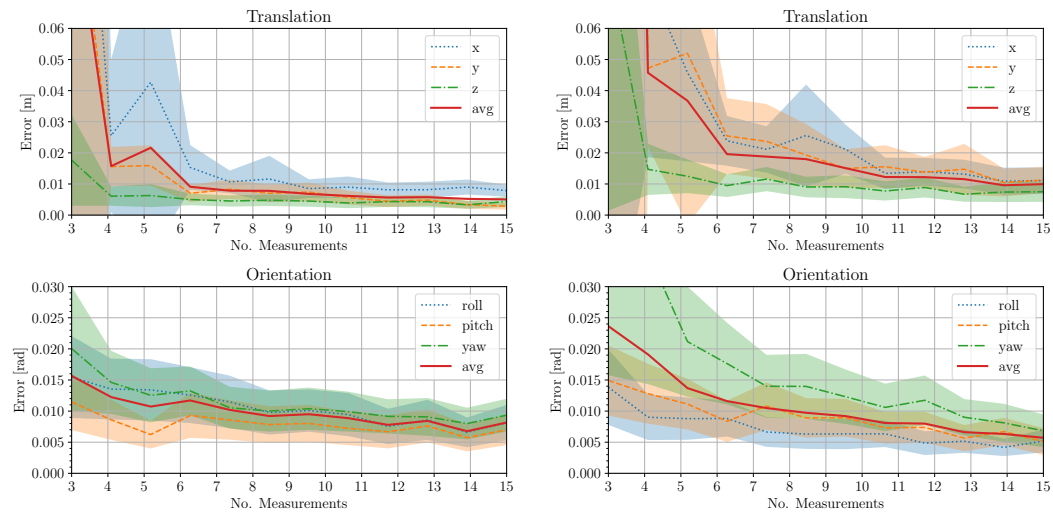
Since no ground truth information is available, we take advantage of the stereo extrinsics to provide an estimate of the calibration error. The offset between multiple camera is measured using optical calibration procedures which typically reach subpixel precision.

In the first experiment, we consider the LiDAR and the Realsense T-265 sensor which provides factory calibrated intrinsic/extrinsic parameters for both cameras. The task of the experiment is to demonstrate the accuracy of the calibrator in real case scenarios and to understand 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 extrinsics for both cameras to obtain an estimate stereo transform. Assuming approximately symmetrical errors in the two cameras, Fig. 5.6a shows the results of this experiment. We were able to obtain 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 acquisition procedure is the same as in the first experiment and Fig. 5.6b shows the experimental result, where we obtain the best solution with 4.6 mm and 0.002





(a) Average camera-wise calibration error in the LiDAR-T265 case. (b) Average camera-wise calibration error in the LiDAR-Manta case.

**Figure 5.6.** Average calibration errors evaluated on separate profiles using real-data. The plots shows how increasing the number of measurements lead to an enhancement in accuracy within our calibration approach.

rads in orientation.

Moreover, Fig. 5.1 and Fig. 5.3 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 occlusions effects, that have been mitigated with an hidden point removal algorithm [64].

Our evaluation indicates that our method is capable of generating extrinsic estimates that are comparable or superior to those obtained using other state-of-the-art approaches. It is important to note that careful consideration is required when selecting a minimal number of measurements. However, our experiments, consistently with the theory, show that the accuracy of these estimates improves as the number of measurements increases.

## 5.4 Conclusion

The experiments in this chapter, 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. 5.1 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 distortion models (Kannala-Brandt [63] for T-265 and Radial-Tangential [17] 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 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

## **605. Ca<sup>2</sup>Lib: Simple and Accurate LiDAR-RGB Calibration using a Small Common Marker**

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 time offsets worsen the calibration accuracy. We remark on the difficulty of synchronizing these two sensors due to their different nature. In particular, the revolution period for typical LiDARs is higher compared to the exposure time of a RGB sensors. We suggest acquiring RGB images when the LiDAR scan overlaps the camera field of view.

One possible addition that would benefit this work is an automatic detection system for calibration targets on LiDAR measurements. This problem may be tackled from a spatial perspective on the raw point cloud or visually by projecting the cloud on a 2D embedding. This feature would either fully or partially replace the current human-aided LiDAR plane detection by providing a good initial guess on the calibration target position.

In summary, this chapter introduces a simple and effective method for accurately estimating extrinsic parameters between LiDARs and RGB sensors. By leveraging the inherent planarity 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. Finally, we also release an open source C++ implementation to benefit the community. This versatile calibration paves the way to systems that operate jointly with measurements from both the sensors, as the one proposed in next chapters: Chapter 6 and Chapter 8.



## Chapter 6

# Photometric LiDAR and RGB-D Bundle Adjustment

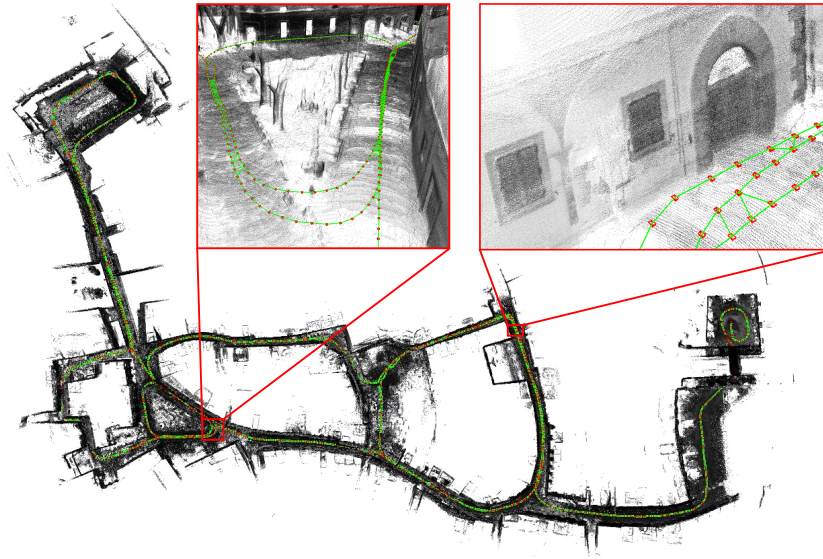
In the introductory parts and mainly in Chapter 4, we discussed the popularity and importance of SLAM in robotics and computer vision. Modern SLAM systems typically comprise two key components: a front-end that estimates sensor odometry in real-time and a back-end that optimizes previous sensor poses and, eventually, the 3D map. The gold standard for the back-end optimization is Bundle Adjustment [124].

In line with our previous works, the main contribution of 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 among the whole set of sensor poses. Our approach implicitly addresses the data association while straightforwardly supporting both RGB-D and LiDAR. We performed a comparative evaluation of benchmark data concerning state-of-the-art sensor-specific refinement strategies and SLAM algorithms. Results show that our simple optimization schema is very effective, performing on par or better than methods specialized for RGB-D and LiDAR data. Given a good calibration between the two sensors, as the one proposed in Chapter 5, 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 available at <https://github.com/rvp-group/ba-mdslam>. Fig. 6.1 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.

### 6.1 Related Work

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 state-of-the-art of both approaches. We finally discuss BA applications in LiDAR SLAM.



**Figure 6.1.** 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 [33] 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.

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. However, they are not operating at sub-pixel resolution, these methods are usually less accurate [108]. State-of-the-art 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 [86, 69].

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 [27]. Goldlücke *et al.* 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 [47]. Slavcheva *et al.* perform pairwise alignment by registering two consecutive signed distance functions (SDFs) generated from the depth images, which is used as global refinement [112]. 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. Hatem Alismail *et al.* consider only pixels with reasonable gradient in the error function to reduce the problem's size [8]. Eriksson *et al.* propose a consensus-based optimization to parallelize BA for large-scale problems [38]. Demmel *et al.* 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 [29].

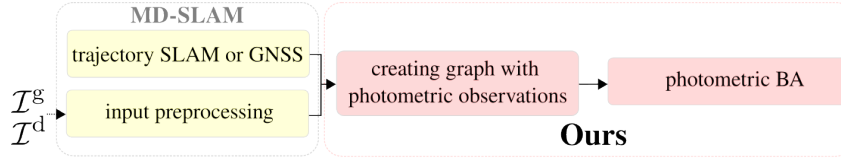
In between the class of direct and indirect methods, we find the work of Forster *et al.* [41], which proposes a hybrid method to estimate the camera's motion. First, an initial guess of the sensor location is computed by minimizing the reprojection error of the world points. Then the estimate is refined 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 [26], 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 [108] 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 [140], subsampled clouds [128] or NDT [118]. Nowadays, LiDAR Odometry and Mapping (LOAM) is perhaps one of the most popular methods for LiDAR odometry [140, 141]. 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 [110], 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 [110, 33]. 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 [139].

In order to remove these inconsistencies, recently Liu and Zhang presented a pose-only global geometrical optimization methodology that considers cloud measurements [76]. 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



**Figure 6.2.** The flow of our approach. The input of our system contains the initial guess of the sensor pose, the intensity image  $\mathcal{I}^g$  and the depth image  $\mathcal{I}^d$ . 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. 6.2.1) and running a global optimization on the set of poses  $\mathbf{X}_{1:N}$ , as detailed in Sec. 6.2.2.

a static data association based on the co-visibility of landmarks; thus, it requires a good initial guess to operate.

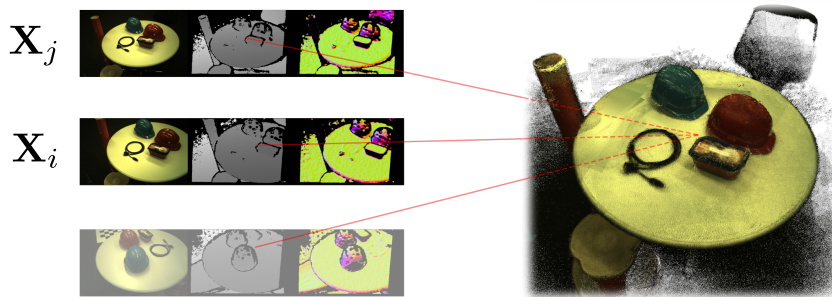
In recent years, the vertical resolution of modern 3D LiDARs increased, and these devices provide scans that resemble more and more dense panoramic images. This allows us to transfer results about direct global refinement from vision to LiDAR. In this work, we present a unified BA strategy that works 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; consistently moving towards the optimum. Our algorithm’s capabilities are demonstrated through quantitative and qualitative analyses, consistently improving the initial estimates provided by any SLAM systems.

## 6.2 Photometric Bundle Adjustment

The goal of our approach is to compute the set of sensor poses  $\mathbf{X}_{1:N}$  whose photometric error between overlapping frames is minimized. The input of our system is a set of triplets  $\langle \mathbf{X}_i, \mathcal{I}^g, \mathcal{I}^d \rangle$ , containing the initial guess of the sensor pose, the grayscale/intensity image  $\mathcal{I}^g$  and the depth image  $\mathcal{I}^d$ . The workflow of our method is illustrated in Fig. 6.2. The first step, equally to our previous work MD-SLAM, is to generate an augmented image pyramid from each pair of images in a triplet Sec. 4.2.1. The second step is determining which pairs of images observe a common structure, which is discussed in Sec. 6.2.1. Finally, these pairs will be used to instantiate a photometric optimization problem presented in Sec. 6.2.2. 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 state-of-the-art approaches.

### 6.2.1 Graph construction

Our system takes as input a collection of triplets  $\langle \mathbf{X}_i, \mathcal{I}^g, \mathcal{I}^d \rangle$ , that consist of the initial estimate of the sensor’s pose and pairs of images for both intensity and depth. To instantiate Eq. (6.3) 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.



**Figure 6.3.** An example of pose-pairs associated, 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. 4.2.1, i.e., images of different resolutions). Here, we show just one level for simplicity. The data used is from ETH3D.

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 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).

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. 6.3.

### 6.2.2 Photometric cost function

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. Note that the formulation is very similar to the one explained in Sec. 4.2.2, however this involves multiple variables rather the one. For clarity, we provide full details of this methodology below. In addition, for completeness, we provide full derivation of the analytic Jacobians.

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, and 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  $\mathcal{I}_i^g(\mathbf{u})$  and the pixel  $\mathcal{I}_j^g(\mathbf{u}')$  of the second image. The evaluation point  $\mathbf{u}'$  is

computed by inverse projecting the pixel  $\mathbf{u}$  from  $\mathcal{I}_i^g$  onto the image plane of  $\mathcal{I}_j^g$ . This accounts for the relative transform  $\mathbf{X}_{j,i} = \mathbf{X}_j^{-1}\mathbf{X}_i$  between the two frames, as follows:

$$\mathbf{u}' = \pi \left( \mathbf{R}_j^T \left( \mathbf{R}_i \pi^{-1}(\mathbf{u}, d) + \mathbf{t}_i - \mathbf{t}_j \right) \right) \quad (6.1)$$

To carry out this operation, the depth at the pixel  $d = \mathcal{I}^d(\mathbf{u})$  needs to be known (see Sec. 2.2). Standard photometric optimization seeks to find the following minimum:

$$\mathbf{X}_{1:N}^* = \operatorname{argmin}_{\mathbf{X}_{1:N}} \sum_{\langle i,j \rangle} \sum_{\mathbf{u}} \|\mathcal{I}_i^g(\mathbf{u}) - \mathcal{I}_j^g(\mathbf{u}')\|^2 \quad (6.2)$$

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. (6.2) 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 [24] these mappings can be encapsulated by the function  $\zeta^c(\mathbf{X}_{j,i}, \mathcal{I}_i^c(\mathbf{u}))$  that calculates the *pixel* value of the  $c^{\text{th}}$  cue after applying the transform  $\mathbf{X}_{j,i}$  to the original channel value  $\mathcal{I}_i^c(\mathbf{u})$ . We use the term *cue* indicated with superscript  $c$ , to refer to a generic channel, since this method holds for normals, depth and grayscale images. We can thus rewrite a more general form of Eq. (6.2) that accounts for all cues and captures this effect as follows:

$$E(\mathbf{X}_{1:N}) = \sum_{i,j} \sum_{\mathbf{u}} L \left\| \sum_c \left( \zeta^c(\mathbf{X}_{j,i}, \mathcal{I}_i^c) - \mathcal{I}_j^c(\mathbf{u}') \right) \right\|_{\Omega^c}^2 \quad (6.3)$$

$$\mathbf{X}_{1:N}^* = \operatorname{argmin}_{\mathbf{X}_{1:N}} E(\mathbf{X}_{1:N}) \quad (6.4)$$

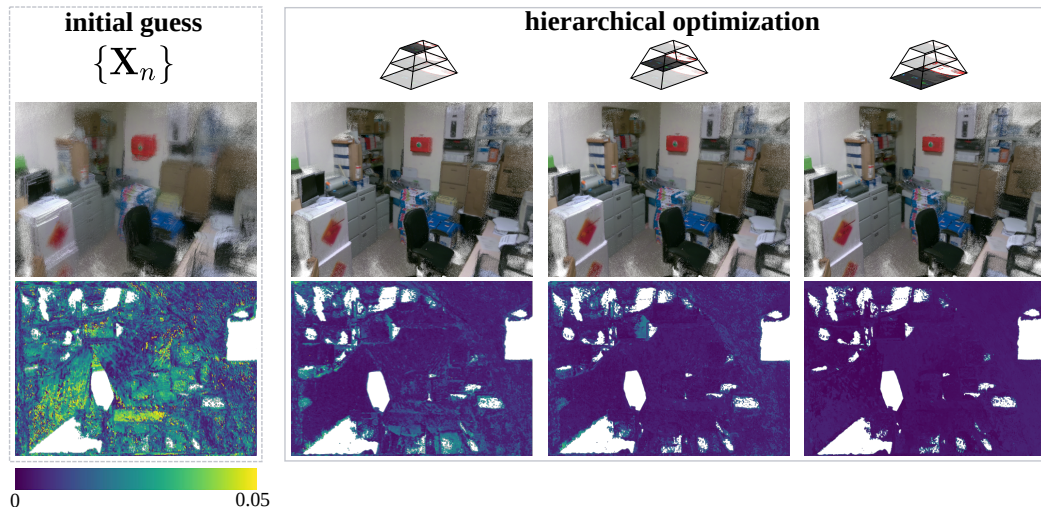
where  $L$  is a Huber robust loss. The squared Mahalanobis distance  $\|\cdot\|_{\Omega^c}^2$  is used to weight the different cues. In our experiment we typically set  $\Omega^g = 0.6$ ,  $\Omega^n = 0.8$  and  $\Omega^d = 1$ , giving more importance to depth and surface normals compared to grayscale channel. For the reasons illustrated in Sec. 2.3.2, to carry on the minimization in Eq. (6.4) we employ the Levenberg-Marquardt algorithm implemented in the `srrg2_solver` [49].

At each iteration, we suppress the occluded portions of the images before evaluating Eq. (6.3). 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. 6.4 shows the effect of this hierarchical approach applied to a BA problem. Generally, the worse the initial guess, the more the required levels.

### 6.3 Experimental Evaluation

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 state-of-the-art SLAM and BA packages developed specifically for each of these sensor types.





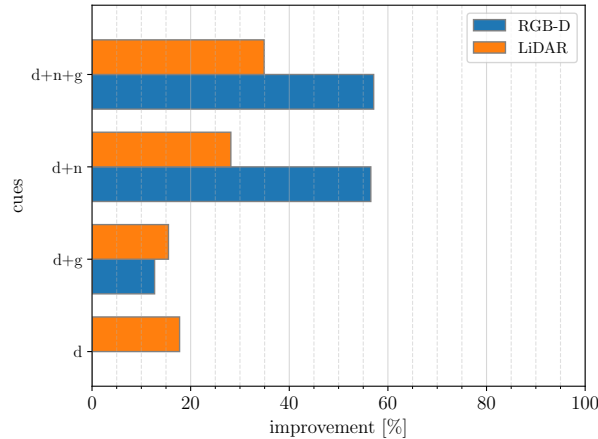
**Figure 6.4.** The effect of hierarchical optimization when performing BA. From left to right, we show how the quality of the estimate improves, starting from a bad initial guess. The heatmap is normalized between 0 and 0.005 [m] for better visualization. The data is self-recorded using an Intel D455.

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 [56], and the timestamps are used to determine the associations. Then, we calculate the RMSE of the translational differences between all matched poses. In Sec. 6.3.1, we discuss the approaches and the datasets used for comparison with RGB-D sensors, while in Sec. 6.3.2 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. 6.6). The image reports timings for each complete optimization iteration at different resolutions.

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. (6.3) 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 merely a few iterations on the finest level.

In Sec. 6.3.3, 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. 6.5, 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.



**Figure 6.5.** 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.

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.

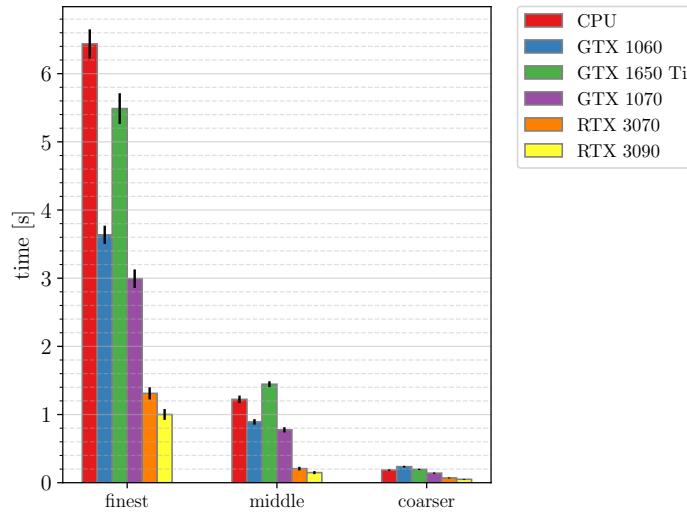
### 6.3.1 RGB-D

As a public benchmark for RGB-D we used several sequences of ETH3D [108]. 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 [33], 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[66], ElasticFusion [133], BundleFusion[26] BAD-SLAM[108], ORB-SLAM2 [86]. DVO SLAM implements a mixed geometry-based and direct registration. Internally the alignment between pairs of keyframes 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 com-





**Figure 6.6.** 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.

binest 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. 6.1, we can see that our photometric refinement performs on par (second after BAD-SLAM) with other state-of-the-art 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. 6.4 and Fig. 6.3 illustrate the effect of the hierarchical optimization on self-recorded data.

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

### 6.3.2 3D LiDAR

To validate our approach on LiDAR measurements we used both our data and public benchmarks. We used the Newer College dataset [142] as a public benchmark. The dataset 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

	ElasticFusion [133]	ORB-SLAM2 [86]	DVO-SLAM [66]	BundleFusion [26]	BAD-SLAM [108]	MD-SLAM [33]	MD-SLAM + Ours
table3	–	0.007	0.008	0.017	<b>0.002</b>	0.016	0.009
table4	0.012	0.008	0.018	–	<b>0.002</b>	0.023	0.008
table7	–	0.010	0.007	0.010	<b>0.003</b>	0.018	0.009
cables1	0.018	0.007	<b>0.004</b>	0.022	0.007	0.021	0.006
plant2	0.017	0.003	0.003	0.004	<b>0.001</b>	0.005	<b>0.001</b>
planar2	0.011	0.005	<b>0.002</b>	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 6.1.** ATE RMSE [m] on ETH3D benchmark [108], recorded with global shutter camera and synchronous streams. ElasticFusion fails in *table3* and *table7*, BundleFusion fails in *table4*.

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 [76], 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 [110], SuMA [13] 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. 6.2 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

	LeGO-LOAM [110]			MD-SLAM [33]			SuMA [13]		
	SLAM	BALM2 [76]	Ours	SLAM	BALM2 [76]	Ours	SLAM	BALM2 [76]	Ours
cloister	0.20	0.25	<b>0.16</b>	0.36	0.37	<b>0.32</b>	3.34	2.67	<b>2.53</b>
quad	<b>0.09</b>	<b>0.09</b>	<b>0.09</b>	0.25	1.98	<b>0.17</b>	1.74	1.72	<b>1.68</b>
stairs	<b>3.20</b>	5.13	3.48	0.23	0.38	<b>0.10</b>	0.67	0.86	<b>0.60</b>

**Table 6.2.** ATE RMSE [m] on Newer College dataset [142], recorded with OS0-128. We always improve the SLAM baseline, a part for *stairs* starting from LeGO-LOAM estimate.

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.

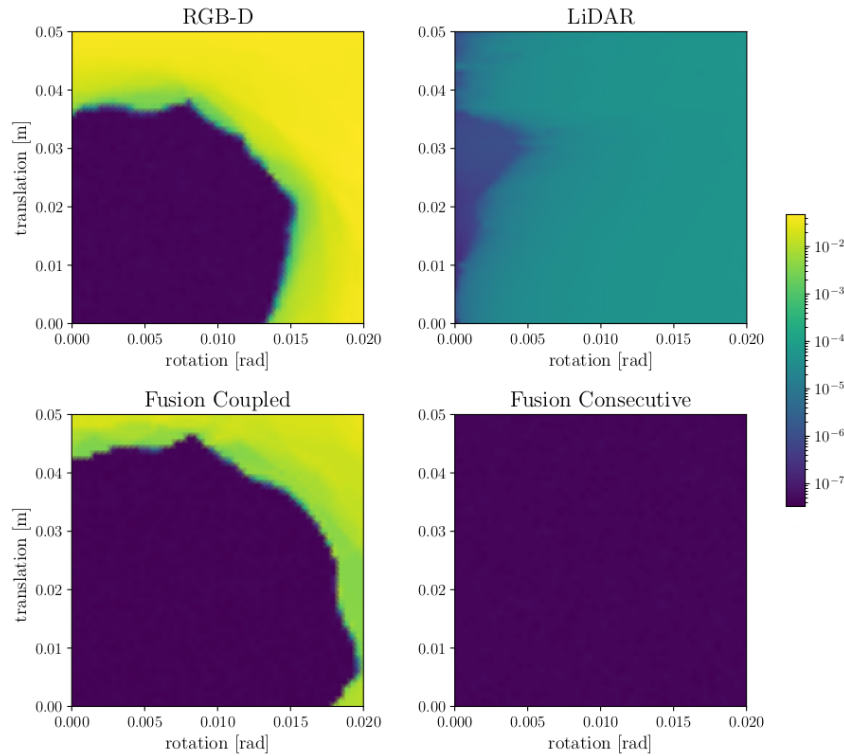
Fig. 6.1 shows our large-scale reconstruction of the historical part of Viterbo (Italy) from self-recorded data using a hand-held Ouster OS0-128 LiDAR. The trajectory is about 2 km long, and the dataset is available from our [repository](#). In addition, we illustrate some additional trajectory plots related to the LiDAR experiments (Fig. 6.8). We did not provide the figures for RGB-D estimate since errors are always close to millimeters.

### 6.3.3 Photometric multimodal: RGB-D + 3D LiDAR

Thanks to the unified nature of our pipeline, it allows the straightforward integration of multiple heterogeneous sensors. In Sec. 6.2.2, we formulated an optimization problem that estimates the sensor’s trajectory  $\mathbf{X}_{1:N}$ .

In a multiple sensors 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. (6.3) to include the constant sensor offsets. Our Appendix (Chapter A) reports this extended formulation. 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 calibrate the offset between the two sensors using a robust point-to-plane alignment [22]. 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. 6.7, 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



**Figure 6.7.** 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.

very large 360 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 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  $E^c(\mathbf{X}_{1:N})$  and LiDAR  $E^l(\mathbf{X}_{1:N})$ , as follows

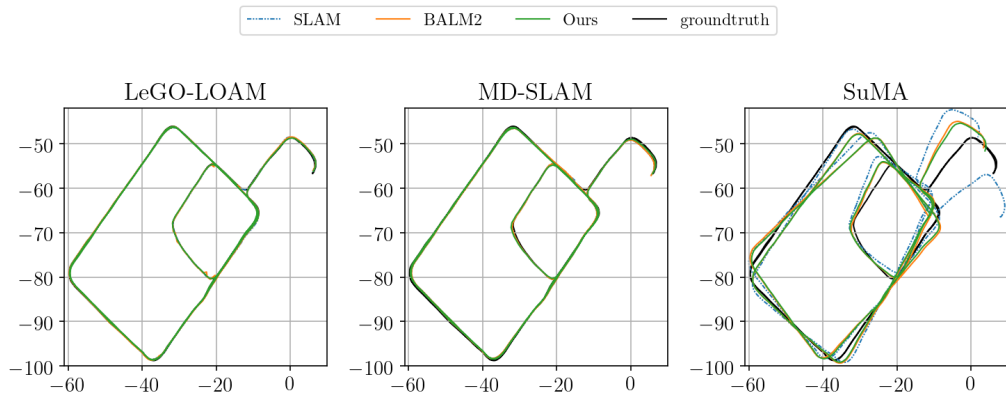
$$\mathbf{X}_{1:N}^{\text{coupled}} = \underset{\mathbf{X}_{1:N}}{\operatorname{argmin}} E^c(\mathbf{X}_{1:N}) + E^l(\mathbf{X}_{1:N}) \quad (6.5)$$

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.

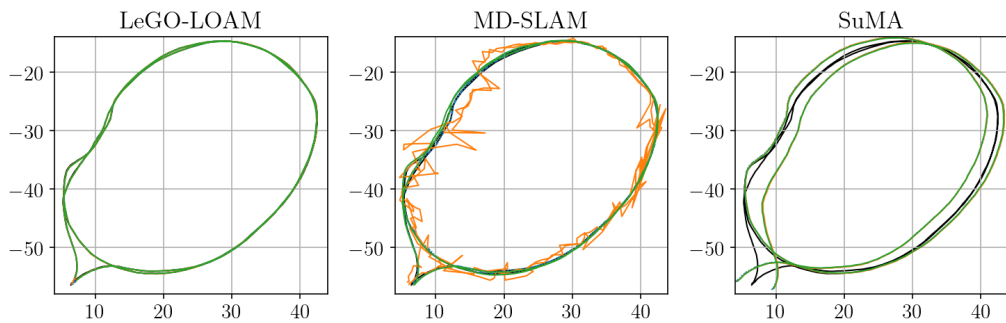
The results suggest that fusing the two sensors always performs better than single sensors (Fig. 6.7). Specifically, *coupled* is generally more accurate compared to RGB-D only, having the converge basin incremented by the LiDAR. Similarly, *consecutive* seems to be the best since it takes full advantage 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.

## 6.4 Conclusion

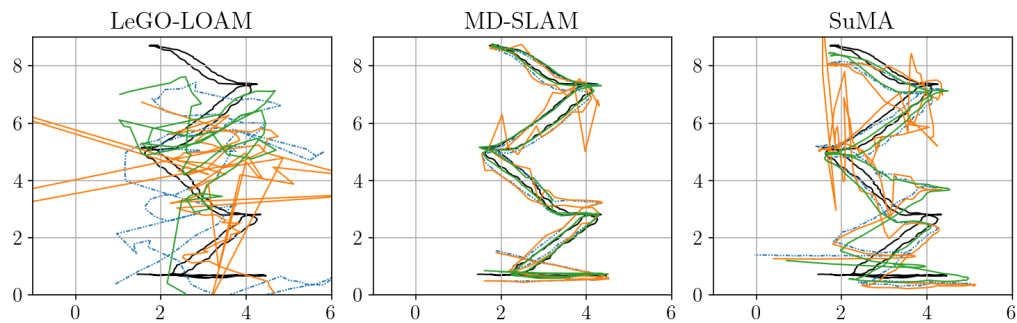
In this chapter, we presented a photometric BA methodology that operates both with RGB-D and LiDAR in a unified manner. To the best of our knowledge, our approach is the only open-source BA system that works independently for depth sensors and specifically exploits the photometric capabilities of the LiDAR image. Comparative experiments show that our simple schema performs on par or better compared to existing sensor-specific state-of-the-art 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. In the upcoming chapter, we introduce a new 3D reconstruction and SLAM benchmark. To enhance the ground truth, we employ our LiDAR BA technique, augmenting the cost function delineated in Sec. 6.4 with supplementary geometric terms, bolstering both convergence and accuracy. In the last chapter, we demonstrate how, by leveraging new data and synchronously utilizing both LiDAR and camera, we can capitalize on the multimodal benefits in terms of both resilience and execution times.



(a) Trajectory estimates of *cloister* sequence. SLAM estimate in blue, BALM2 estimate in orange, Ours in green and ground truth in black.



(b) Trajectory estimates of *quad* sequence. SLAM estimate in blue, BALM2 estimate in orange, Ours in green and ground truth in black. BALM2 suffer of less dense trajectories, since MD-SLAM spawn less keyframe poses to reduce odometry drift.



(c) Trajectory estimates of *stairs* sequence. SLAM estimate in blue, BALM2 estimate in orange, Ours in green and ground truth in black. BALM2 suffer of rotations, as it is observable from MD-SLAM and SuMA initial guess. For completeness we include the LeGO-LOAM plot, however both our and BALM2 fail due to bad initial guess.

**Figure 6.8.** Plots of trajectory estimates in different LiDAR sequences of Newer College [142].

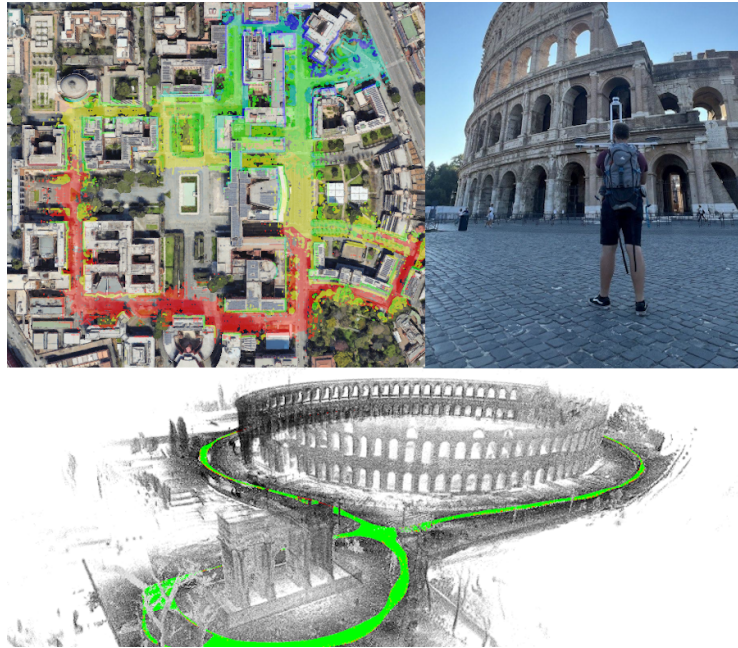
## Chapter 7

# VBR : A Vision Benchmark in Rome

Computer vision communities have relied on standard datasets to enhance their techniques since the early days. When ground truth data was accessible for a specific task, these communities devised 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 [81, 108, 68, 143] have been presented, but no one seemed to have the same impact on the robotics and computer vision community as the original work [44].

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 [44, 68, 136, 18] 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 [81], offering hand-held motion with a more accurate ground truth [98]. Still, to our knowledge, none of the recent datasets seem to address multiple issues. In this chapter, we present a contribution that aims to approach all these aspects simultaneously. At the moment of writing, we propose 6 datasets 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 have 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  $\pm 3$  cm along an indoor/outdoor trajectory of 1.5 km. For each dataset we provide two flavors, similar to KITTI: a training version with ground truth available and a benchmarking version where the ground truth is not publicly provided. The results of the community on the training datasets will be evaluated off-core. The public benchmark with the leading table will be available at our website.



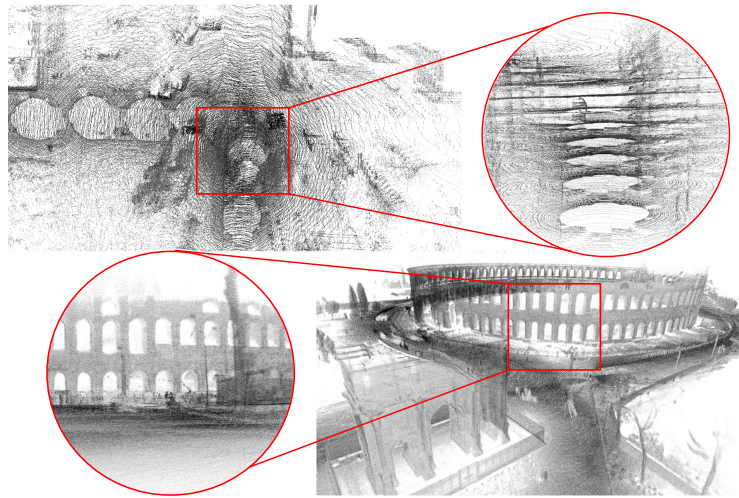
**Figure 7.1.** A summary of our dataset. Data illustrating some of the sequences recorded (top). 3D mapping done with of our ground truth (bottom).

## 7.1 Related Work

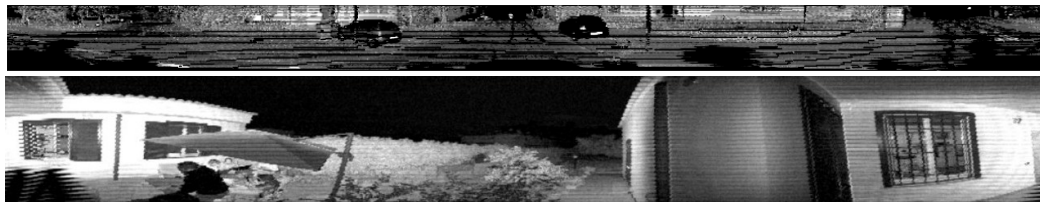
Within the domain of SLAM and 3D reconstruction, several datasets have played pivotal roles in benchmarking and advancing autonomous robotics systems and algorithms. These datasets vary in terms of their operational contexts, sensory configurations, and the accuracy of their associated ground truth data. 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. 7.2, Fig. 7.3). Still, KITTI was pivotal in the development of many popular SLAM methods. Oxford RobotCar is another noteworthy car dataset [81]. 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. Furthermore, approaches like Mulran use the same ground truth generation process, leading to the same issue. However, this dataset is renowned for embracing multimodal sensor data, including LiDAR and radar. While this adds diversity to the sequences, only the front half of the LiDAR field-of-view is included in the data collection [68].

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





**Figure 7.2.** Comparison between LiDAR clouds attached to ground truth trajectories of KITTI (up) and ours (down). The zoom shows the elevation view.



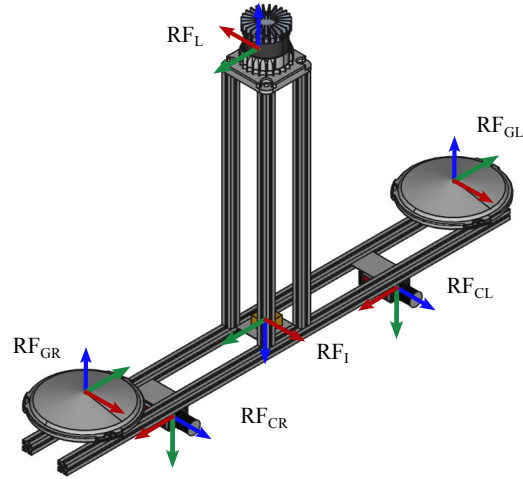
**Figure 7.3.** Projection of the KITTI LiDAR point cloud into an image plane (up), projection of our LiDAR into an image plane (down). The many holes of the up image due to uneven distribution of the LiDAR beams and calibration issues make the KITTI LiDAR image unusable for computer vision tasks.

[19] 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 acquisition 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 [98] and Hilti [143], have achieved exceptional levels of accuracy in ground truth generation through the use of 3D imaging laser scanners. This innovative 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 paper introduces 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-dof ground truth even in large scale scenarios.



**Figure 7.4.** Sensor setup and reference frames. Our ground truth is expressed in the LiDAR reference frame  $RF_L$ . More details can be found in our website and supplementary materials.

	Accurate GT	Indoor	Outdoor	Driving	Large-scale	Benchmark
KITTI [44]			✓	✓	✓	✓
EuRoC [19]	✓	✓				
Oxford RobotCar [81]			✓	✓		
ETH3D [108]	✓	✓				✓
MulRan [68]			✓	✓	✓	
Newer College [98, 142]	✓		✓			
Hilti [143]	✓	✓	✓			✓
Kitti-360 [?]	✓	✓		✓	✓	
<b>Ours</b>	✓	✓	✓	✓	✓	✓

**Table 7.1.** The table summarizes the most important 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	20 Hz
		Stereo configuration Wide baseline	30 Hz
IMU	SBG Ellipse-E	GNSS synchronization 0.05° precision	100 Hz

**Table 7.2.** Summary of the devices in our sensor setup, and the accuracy of the temporal synchronization.

## 7.2 The Datasets

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 section delves into our approaches to address these issues.

### 7.2.1 Sensors setup

Our sensor system is illustrated in Fig. 7.4 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.

The Inertial Measurement Unit (IMU) is an SBG Ellipse-E IMU, 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. 7.1 shows the sensor placement on the car. Tab. 7.2 summarizes the devices used in our system.

### 7.2.2 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 [15]. Using the same target, we estimate the rigid transformation between the right camera ( $R_{FCR}$ ) and the LiDAR. We achieve accurate results by minimizing plane-to-plane error [46]. 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 [35] 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).

### 7.2.3 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.

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.

### 7.2.4 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 ([98], [143]). This is not always possible when moving in a very large environment. In such cases, a GNSS RTK system is usually 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.

In the remainder, we assume a 3D pose  $\mathbf{X} \in \mathbb{SE}(3)$  be represented as a homogeneous  $4 \times 4$  matrix where  $\mathbf{R}$  is the rotation matrix, and  $\mathbf{t}$  is the translation vector. With the operator  $\log(\mathbf{X})$ , we refer to the conversion of a transform in minimal form (e.g. translation vector and unit quaternion for the orientations).

We combine the GPS priors  $\mathbf{Z}_t^g \in \mathbb{SE}(3)$  with a variation of BA formulation proposed in [34]. 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  $\mathbf{Z}_{t,t+1}^{\text{sm}}$  between subsequent frames. To this extent, we use the same ICP algorithm used for temporal synchronization mentioned in Sec. 7.2.3. 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:

$$\begin{aligned} \mathbf{X}_{1:T}^* = \underset{\mathbf{X}_{1:T}}{\operatorname{argmin}} \sum \|\log(\mathbf{Z}_{t,t+1}^{\text{sm}-1} \mathbf{X}_t^{-1} \mathbf{X}_{t+1})\|_{\Omega_{t,t+1}^{\text{sm}}}^2 \\ + \sum \|\log(\mathbf{Z}_t^{g-1} \mathbf{X}_t)\|_{\Omega_t^g}^2 \end{aligned} \quad (7.1)$$

Here  $\|\mathbf{v}\|_{\Omega}^2 = \mathbf{v}^T \Omega \mathbf{v}$  denotes the Omega L2 norm of a vector  $\mathbf{v}$ , with  $\Omega$  representing the information matrix encoding the accuracy of the measurement. Accordingly, in Eq. (7.1)  $\Omega_{t,t+1}^{\text{sm}}$  is the information matrix resulting from scan matching, while  $\Omega_t^g$  encodes the GPS accuracy.

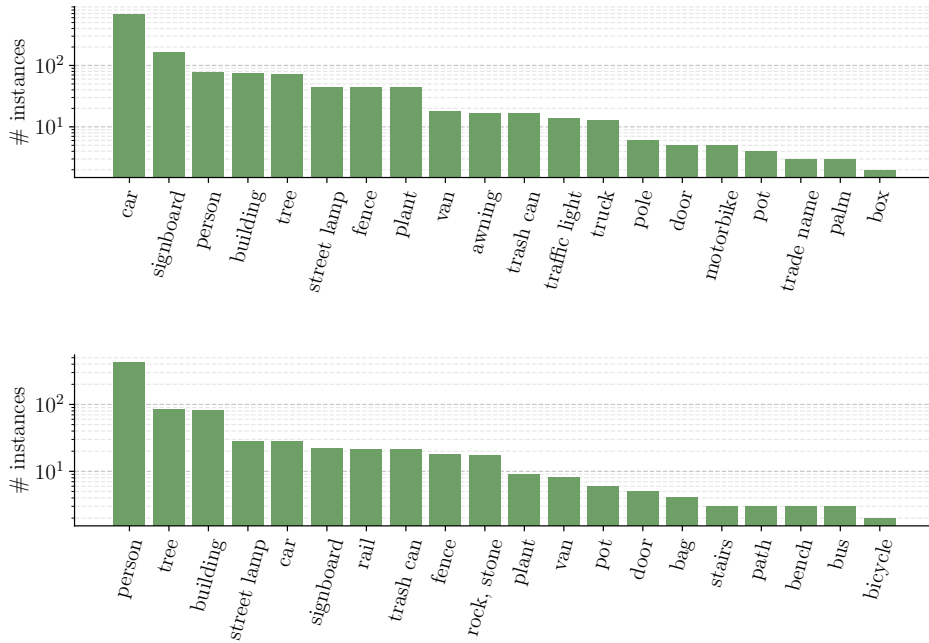
Once this process is completed and we have a reasonable initial guess, we look for the set of poses that is maximally consistent with all LiDAR scans. We solve the following problem using geometric and photometric error terms. The geometric part based on point-to-plane results in a configuration of the rigid motion which is close to the optimum. The second photometric step, increases the accuracy by ensuring subpixel consistency. Let  $\langle i, j \rangle$  be the set of poses,  $\langle k, l \rangle$  geometric associations, and  $u$  the image pixel generated by spherical projecting the LiDAR point cloud into an image (as illustrated in [34]). The total residual can be expressed as follows:

$$E^{\text{ba}} = \sum_{i,j,k,l} \rho_{\text{geo}} \|e_{k,l}^{\text{geo}}\|_{\Omega_{\text{geo}}}^2 + \sum_{i,j,u} \rho_{\text{photo}} \|e_u^{\text{photo}}\|_{\Omega_{\text{photo}}}^2. \quad (7.2)$$

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:

$$e_{k,l}^{\text{geo}}(\mathbf{X}_i, \mathbf{X}_j) = (\mathbf{X}_i \mathbf{p}_{i,j} - \mathbf{X}_j \mathbf{p}_{k,l}) \cdot (\mathbf{R}_i \mathbf{n}_{i,k}) \quad (7.3)$$

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



**Figure 7.5.** Number of top 20 most frequent semantic instance for Ciampino (above) and Colosseum (below) sequences. The instances were counted using OneFormer [?] over a subset of images for each sequence and excluding the most predominant classes: sky, wall, road, grass, sidewalk, ground.

in [34], but relies only on range and intensity images. The overall error function to minimize, taking into account GPS information, will be therefore

$$\mathbf{X}_{1:T}^{gt} = E^{ba} + \sum \|\log(\mathbf{Z}_t^{g-1} \mathbf{X}_t)\|_{\Omega_t^g}^2 \quad (7.4)$$

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 redundantly all degrees of freedom. 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  $\pm 3$  cm accuracy on a trajectory of length of approximately 1.5 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  $\text{RF}_L$ .

### 7.2.5 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  $90^\circ$ , 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. 7.5). Tab. 7.3 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 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.

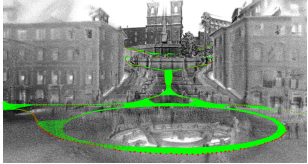
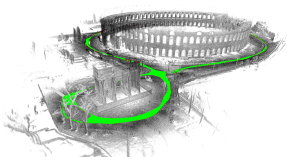
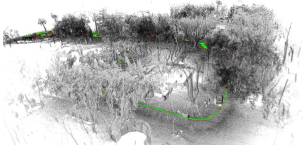
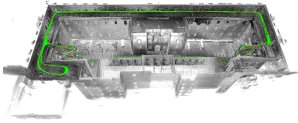
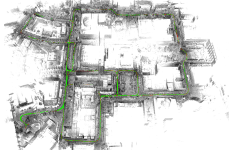
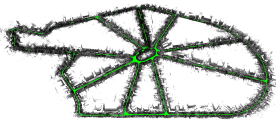
**DIAG** this sequence is a mixed indoor/outdoor dataset. We traveled inside and outside our building, walking inside the corridors, through the courtyard, up to the stairs, and on the roof, which is the only part where the reception of RTK-GPS was available.

**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. 7.7). It is the longest sequence so far: the length of the total trajectory is about 21 km, while being subject to moderate dynamics.

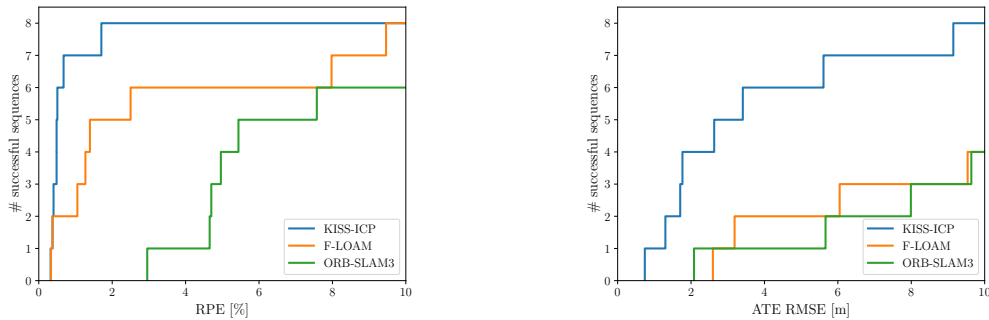
## 7.3 Benchmark

For a detailed assessment of SLAM and odometry estimation, we concentrate on the Absolute Trajectory Error (ATE) and Relative Pose Error (RPE). As in [44], we assess rotation and translation errors independently rather than merging them into one metric.

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

**Table 7.3.** Summary of our sequences.





**Figure 7.6.** Our benchmark. Cumulative RPE [%] and ATE RMSE [m] across all training sequences in our dataset for KISS-ICP, F-LOAM and ORB-SLAM3.

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 [56]. 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), over a set of subsequences of different lengths, as proposed by [44]. 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. 7.6. 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.

Additional information, supplementary materials, and the leading table can be accessed on our website.

### 7.3.1 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 [129], F-LOAM [132] and ORB-SLAM3 [20]. The evaluation outcomes are reported in Fig. 7.6, 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.



**Figure 7.7.** The image shows the overlay of the 3D model obtained from our ground truth system and a view from Google Maps.

## 7.4 Conclusion

In this chapter, we present a new 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 leading table 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.).

## Acknowledgement

We thank Juan D. Tardos for supporting us with ORB-SLAM. We are grateful to Roberto Mauroni and Francesco Spognardi for their invaluable support. Special thanks to Vincenzo Suriani for providing supplementary hardware. Last but not least, we appreciate Luca Calocchia's contribution in developing the website.

## Chapter 8

# Multimodal, fast and uniform 3D reconstruction

Leveraging the combined strengths of both LiDAR and camera sensors can lead to enhanced 3D reconstruction capabilities. Throughout this thesis, we delved into the unique advantages each modality offers. LiDAR, with its ability to capture accurate depth information even in challenging light conditions, is instrumental in crafting reliable 3D maps. On the contrary, cameras provide us with detailed visual content, enriching the depth data with texture and color, attributes LiDAR cannot discern. By integrating the detailed imagery from cameras with the sparse yet precise measurements of LiDAR, we aim for richer, denser, faster, and more precise 3D reconstructions than with a single modality. Moreover, using conventional cameras to estimate depths and achieve a 3D reconstruction is challenging. Furthermore, after investing time in the reconstruction process, great accuracy is not guaranteed. Differently, with precise measurements from LiDAR, we can simplify these complexities, reducing or even eliminating the need for intensive structure optimization. The confluence of these sensors promises enhanced resilience in complex situations — be it the lack of visual features, noisy depth measurements, or limited data availability. This fusion also suggests well for efficiency, curtailing computational demands and offering a path to real-time applications. However, many studies that utilize both of these sensors tend to design intricate systems that separate the use of each sensor, thereby not fully capitalizing on their inherent similarities and synergies. In this concluding chapter, we show how our BA strategy based on photometric and geometric information coupled with an efficient technique to generate dense depth images from the overall LiDAR model can enhance 3D reconstructions. It is important to note that this chapter does not venture into detailed numerical evaluations or exhaustive experiments. With no direct counterpart in the academic landscape, our primary intent here is to provide a glimpse into the potential advantages of combining modalities, both in terms of robustness and computational times, particularly within our cohesive framework.

### 8.1 Related Work

Chapter 4 and Chapter 6 comprehensively review the literature on camera and LiDAR techniques crafted specifically for each sensor, related to SLAM and BA. In this section, our focus narrows to algorithms that simultaneously employ both LiDAR and cameras.

As we discussed in Chapter 2, for perception tasks, especially SLAM and 3D reconstruc-

tion, various sensors are at our disposal. Cameras and LiDARs stand out as the predominant choices. Nevertheless, even with the vast research dedicated to them, they are not without limitations. For instance, cameras can face challenges like scale factor drift in single-camera setups, delayed depth determination, limited stereo-vision range, and occasional sparseness in the mapped reconstructions. Notably, utilizing RGB-D outdoors can be problematic. Contrastingly, 3D LiDAR-centric techniques lean towards scan registration and pose-graph methods. Some of these prioritize detecting landmarks, but often, the resultant point clouds are not dense enough for peak performance. Yet, the main advantage of LiDAR is its good range accuracy, leading to detailed mappings. Combining visual and LiDAR techniques currently seems like a promising frontier in SLAM and mapping, since sensors can complete each other.

A popular tool for sensor fusion, owing to its straightforward application, is the Kalman Filter. As demonstrated in [135], an RGB-D camera combined with LiDAR EKF-SLAM was introduced to overcome visual tracking setbacks. In cases where visual tracking is unsuccessful, the LiDAR pose assists in localizing the RGB-D camera's point cloud data, facilitating 3D map creation. However, this method does not truly integrate the two modalities but rather toggles between them based on situational needs.

In many visual-LiDAR SLAM scenarios, LiDAR is mainly used to estimate motion via scan-matching, while cameras detect features for loop closures [74]. However, as we already seen in Chapter 3, the benefit of accomplishing loop closures through LiDAR is the independence from external light source (i.e. intensity measured will be the same during day and night).

Looking from a different perspective, the impressive results achieved by visual-SLAM algorithms have engaged in employing sensor fusion to derive optimal solutions within these frameworks. As an illustration, Graeter *et al.* employed LiDAR for depth extraction. After projecting LiDAR point cloud onto the RGB frame, motion estimation and mapping proceeded through a visual keyframe-driven schema [48]. Differently, very targeted for application, Scherer *et al.* employed a drone-based hybrid system to map a river's path and adjacent vegetation. Here, visual odometry combined with inertial readings estimated the drone's position, while LiDAR identified obstacles and marked the river's edges [104]. Sauerbeck *et al.* incorporated 3D LiDAR depth into ORB-SLAM3 by building on the RGB-D mode [103]. Zhang and Singh drew from their previous work LOAM [140] to craft VLOAM [141]. This method, blending high-frequency visual odometry with its LiDAR counterpart, aims to polish motion estimates and counteract drifts. Still, two separate and independent methodologies are required.

It is evident that such strategies are more of loose collaborations than fully integrated solutions, often overlooking feature detection within their measurement scopes and focusing more on the technical enhancements for SLAM algorithms. In addition, to our knowledge, no current research aims to exclusively blend the two sensors specifically for mapping purposes.

In this chapter, we delve into the fusion of LiDAR and camera within our consolidated approach, capitalizing on the inherent similarities between the two sensors. Building on our previous works presented respectively in Chapter 5, for an accurate calibration using printable targets and, in Chapter 7 for some new modern, synchronized and challenging data; our qualitative experiments show that our approach can be used effectively for robust and fast 3D reconstructions.

## 8.2 Multimodal Bundle Adjustment

Expanding upon the insights from previous chapters, in Sec. 8.2.2 we delve into how we address multimodal BA within our cohesive framework, utilizing both geometric and photometric errors from LiDAR and cameras. Given the necessity for depth when using cameras and, as previously discussed, the absence of datasets with dense depths in large-scale scenarios or in tandem with LiDAR, we detail our approach to efficiently render dense depth in Sec. 8.2.1. This is achieved using an adapted version of Voxelhashing [89] in combination with the globally refined LiDAR model. While our method is designed to function with either RGB-D or LiDAR independently, as well as in conjunction, the results from previous chapters have underscored the efficacy of our unified approach for individual sensor modalities. However, for simplicity in this discussion, we will focus primarily on the simultaneous optimization of both sensors for 3D reconstruction.

### 8.2.1 Voxelhashing for camera dense depth generation

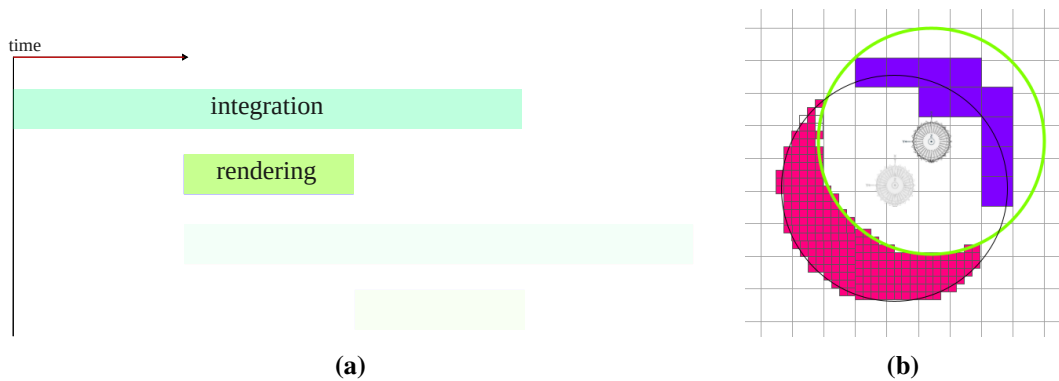
Chapter 7 highlighted our creation of a LiDAR model of the environment using Bundle Adjustment, fusing GNSS and LiDAR odometry to generate a precise ground truth for our benchmark. Specifically, we derive the set of poses that maximize both the geometrical and photometrical consistencies. With the numerous LiDAR measurements acquired, the resultant model is both dense and precise, making the the approach suitable for dense depth view rendering. Yet, considering the model’s density (bilions of points), without an appropriate data structure, reprojection of the model onto each camera view becomes extremely time-consuming.

To address this issue and speed up depth generation, we introduce a GPU-based hash-table. Drawing from the foundational work by Niessner *et al.* called *VoxelHashing* [89], we have developed our own version. Besides compatibility with current hardware, it can handle varied projection types, encompassing both traditional cameras and LiDAR sensors. VoxelHashing, in the realm of computer vision and 3D reconstruction, is pivotal for efficiently generating extensive 3D models from depth cameras promptly. This data structure employs a straightforward spatial hashing technique to condense space, facilitating real-time access and modifications to implicit surface data represented as Truncated Signed Distance Function (TSDF) <sup>1</sup>. Instead of requiring a standard or hierarchical grid data structure, it only densely stores surface data in areas with observed measurements. GPU memory and performance metrics arise when preserving surface data beyond the view frustum in the hash table. To navigate this and support expansive maps, given the high host-GPU bandwidth, a bidirectional streaming approach is adopted. Much like its predecessor, our adaptable data structure is apt for this role, as in-and-out streaming of voxel blocks do not require any reorganization of the hash table. More details about VoxelHasing can be found in the original paper [89], here we briefly describe the main differences with the original implementation and how we employ this data structure for efficient dense depth rendering in our scenario.

Mainly our processing is divided into three steps, *integration*, *rendering* and *streaming*. Initially, we integrate using streaming RAM-GPU all LiDAR range images within up to a maximum consecutive number of frames. After, we render each camera depth in the

---

<sup>1</sup>TSDF is a simplified version of a Signed Distance Function (SDF), which is a mathematical function that describes the distance between any point in space and the nearest point on a given surface. The distance is positive if the point is outside the surface, negative if it is inside, and zero if it is on the surface.



**Figure 8.1.** Streaming Scenario: On the left, Fig. 8.1a illustrates the rendering process over the integration area. The entire upper block is integrated, available in RAM for access, and ready for streaming to the GPU during rendering. The rendering block is processed in the middle of the integrated one, where model is denser. This procedure is repeated until all data has been processed and camera depths for all frames are rendered. On the right, as depicted in Fig. 8.1b, we demonstrate the RAM-GPU streaming process. In this situation, our LiDAR is moving from left-bottom to right-up. The magenta voxels (small squares) are streamed out to RAM since they fall outside the LiDAR’s maximum radius, while the purple world chunks (big squares) are about to be streamed into the GPU because they are entering the LiDAR’s maximum radius. While this happens in 3D space, we draw it in 2D for clearness.

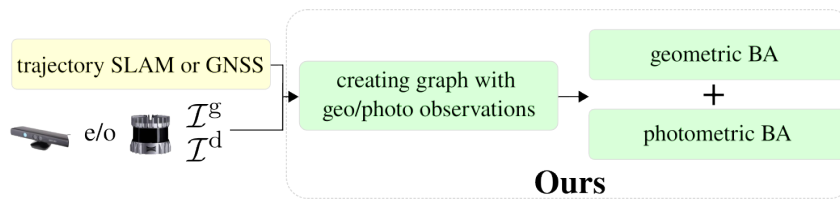
denser area of our integrated model. We repeatedly this process until all corresponding depth images have been rendered. We details each of these steps below.

### Integration

Before adding new TSDFs, we need to allocate voxel blocks that not only fall within each input LiDAR range sample’s footprint but are also inside the truncation zone of the surface reading. We handle range samples concurrently, inserting hash entries and setting up voxel blocks around the noted surface within this truncation area. For every depth input, we generate a ray restricted to the truncation zone. As in the original implementation, since the voxel resolution and block size is known, we employ DDA [9] to identify all voxel blocks intersecting with the ray. We then add a new voxel block entry to the hash table for each suitable candidate. Once allocated, we refresh all the designated voxel blocks present within the LiDAR’s view frustum. Yet, a sizable portion of the hash table remains empty, meaning they do not link to any voxel blocks. Additionally, many voxel blocks lie outside the viewing frustum. Given these factors, TSDF integration becomes highly efficient by exclusively targeting the blocks situated inside the current LiDAR view. We only integrate points that are within 30 meters distance relative to LiDAR for better accuracy and density.

### Rendering

Once we have integrated our model using LiDAR range and intensity data, we are ready to render our camera view with depth information from the model. For this task, we employ the extrinsic between camera and LiDAR (Chapter 5) and the intrinsics calibration parameters of the camera (Chapter 2) to ray-cast the part of the model within the camera view frustum. To maintain a good and sharp quality of depth rendering we assume that each valid TSDF is in



**Figure 8.2.** The flow of our multimodal approach. The system’s input comprises the initial estimate of the sensor pose, the intensity/grayscale image  $\mathcal{I}^g$ , and the depth image  $\mathcal{I}^d$ . The essence of our refinement strategy, highlighted in green in this sketch, involves creating a graph with geometric and photometric observations (Sec. 6.2.1) and executing a global optimization on the set of poses  $\mathbf{X}_{1:N}$ , as elucidated in Sec. 8.2.2. It is worth noting that the input can originate from either RGB-D, LiDAR, or both sensors. In multimodal optimization, our findings suggest using LiDAR for geometric minimization and the camera for photometric error reduction.

the order of millimeters. In addition, given the density of the model, we remove occlusions using simple z-buffer<sup>2</sup>.

### Streaming

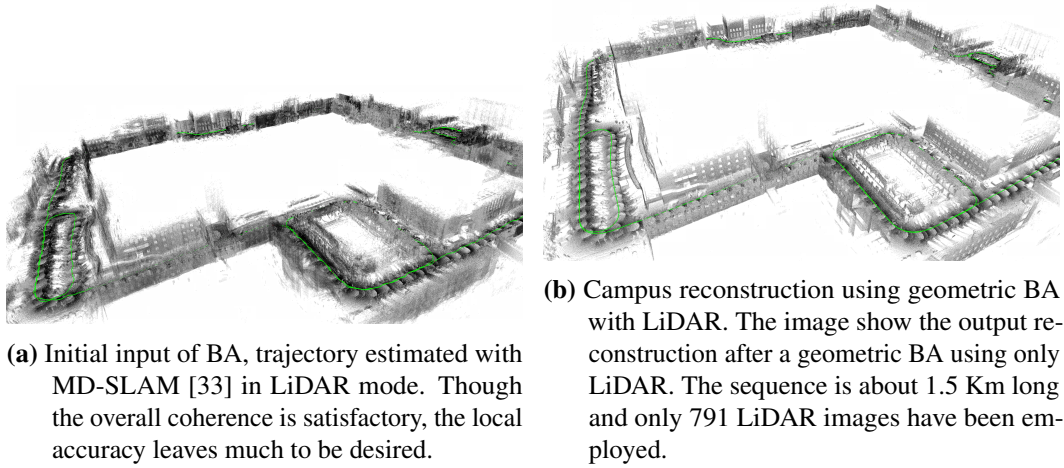
Given that our models encompass billions of points and, faced with constrained memory resources, we stream data from disk to RAM and between RAM and GPU. Because our data is temporally ordered, we process by integrating a fixed number of LiDAR frames each time, settling on 1000 frames for the sake of simplicity. After integrating the requisite number of frames and with all data housed in RAM, we only render camera depths within the core segment of this vast chunk to boost density (refer to Fig. 8.1a for clarity).

Moreover, to take full advantage of the GPU, given the high host-GPU bandwidth, we stream bidirectionally based on active regions and the sensor’s frustum during both the integration and querying stages. To achieve this, we establish an active region described as a sphere that encloses the current LiDAR view frustum, with an additional safety perimeter around it. For our LiDAR, we assume a range up to 50 meters. We position the sphere’s center 25 meters from the LiDAR location, with a radius of 50 meters as shown in Fig. 8.1b. On the host side, voxel data is not structured into a hash table anymore. Instead, akin to the original design [89], we logically partition the world space uniformly into *chunks*, with our current setup designating each chunk as  $10 \text{ m}^3$ . The bidirectional streaming of voxel blocks occurs every frame.

### 8.2.2 Cost function

Expanding on our prior photometric studies outlined in Sec. 6.2.2 and Sec. 6.2.1, we have integrated a geometrical element, similar to the one depicted in Sec. ???. Our objective is to determine the set of sensor positions  $\mathbf{X}_{1:N}$ , ensuring that both geometric and photometric discrepancies between overlapping images are minimized consistently. The system’s input comprises triplets  $\langle \mathbf{X}_i, \mathcal{I}^g, \mathcal{I}^d \rangle$ , which include an initial estimate of the sensor position, the grayscale/intensity image  $\mathcal{I}^g$ , and the depth image  $\mathcal{I}^d$ . Our method’s progression is showcased in Fig. 8.2. The initial phase involves identifying image pairs that capture a shared structure, as elaborated in Sec. 6.2.1. Subsequently, these pairs serve as the foundation

<sup>2</sup>Within a pixel the sample closer to the camera along to the z-axis (forward from the camera) is stored.



**Figure 8.3.** Before and after global geometric optimization using LiDAR.

for a combined photometric and geometric optimization problem, as detailed in Sec. 8.2.2. This approach is always versatile, suitable for LiDAR and RGB-D either individually or in tandem. When employed together, considering the specificities of the sensors – the accurate geometry gauged by LiDAR and the dense data procured by cameras – we predominantly resort to LiDAR for geometric minimization and RGB-D for photometric optimization.

Unlike our purely photometric approach in Sec. 6.2, we avoid the use of pyramids and hierarchical optimization to broaden the convergence basin. Instead, similar to Sec. ??, we utilize geometrical point-to-plane error with explicit data association. This explicit incorporation of geometry enhances the method’s robustness compared to relying solely on photometric error in hierarchical optimization, though it does introduce added complexity to the implementation, particularly in terms of data structure. The overall cost function we seek to minimize is a sum of geometric and photometric terms:

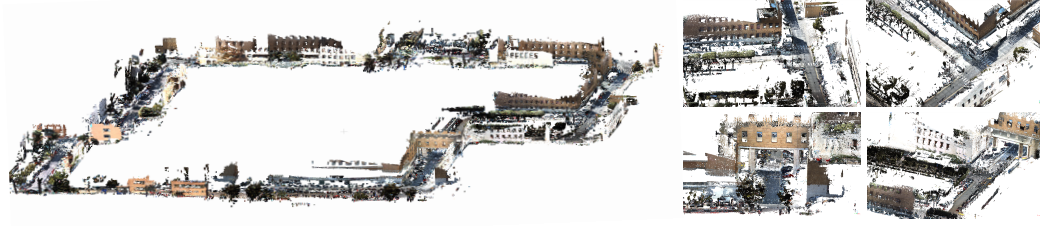
$$E^{\text{ba}} = \sum_{i,j,k,l} L_{\text{geo}} \|e_{k,l}^{\text{geo}}\|_{\Omega_{\text{geo}}}^2 + \sum_{i,j,\mathbf{u}} L_{\text{photo}} \|e_{\mathbf{u}}^{\text{photo}}\|_{\Omega_{\text{photo}}}^2 \quad (8.1)$$

with  $L$  our Huber robust loss,  $k, l$  point to plane association and  $\mathbf{u}$  the pixel value. For clarity, in the following, we wrap-up and detail both geometric and photometric residual, highlighting the differences with the one already presented respectively in Sec. ?? and in Sec. 6.2.2.

### Geometric error

Point-to-plane data associations are commonly used for RGB-D and LiDAR [145] and are known to be effective in ICP [22]. As discussed in Sec. ??, we employ an efficient k-d tree data association based on Principal Component Analysis (PCA) splitting criteria. A k-d tree (short for "k-dimensional tree") is a space-partitioning data structure that is useful for organizing points in k-dimensional space. It is commonly used for range searches and nearest neighbor searches in multi-dimensional data. When combined with PCA, the splitting and search criteria are based on the directions in which the data varies the most (principal components). For instance, the first principal component is the direction of maximum variance, while the second principal component (orthogonal to the first) indicates





**Figure 8.4.** Qualitative evaluation of multimodal BA in one of our *Campus* sequences. The reconstruction has been done using LiDAR within geometric optimization (see Fig. 8.3b) and cameras for photometric, generating dense depth images from LiDAR model through voxelhashing. The sequence is about 1.5 Km long and only 791 RGB images have been used to render the reconstruction.

the direction of the second greatest variance, and so forth. Throughout our optimization, to refine data association at each iteration, adjusting the splitting direction using the estimate  $\mathbf{X}_{j,i}$ . This ensures a streamlined implementation, as well as enhanced final accuracy. As previously introduced in Eq. (??), the geometric term can be compactly expressed as

$$\mathbf{e}_{k,l}^{\text{geo}}(\mathbf{X}_i, \mathbf{X}_j) = (\mathbf{X}_{j,i} \pi^{-1}(\mathbf{u}_{i,k}, d) - \pi^{-1}(\mathbf{u}_{j,l}, d)) \cdot (\mathbf{R}_{j,i} \mathbf{n}_{i,k}) \quad (8.2)$$

with  $\mathbf{u}_{i,k}$  and  $\mathbf{u}_{j,l}$  denoting corresponding pixel (that get inverse projected in 3D) between the poses  $\mathbf{X}_i$  and  $\mathbf{X}_j$ , and  $\mathbf{n}_{i,k}$  be the corresponding normal. Our current implementation, from data association to the construction of the quadratic form (Eq. (2.30), Eq. (2.31)) is parallelized in CPU.

### Photometric error

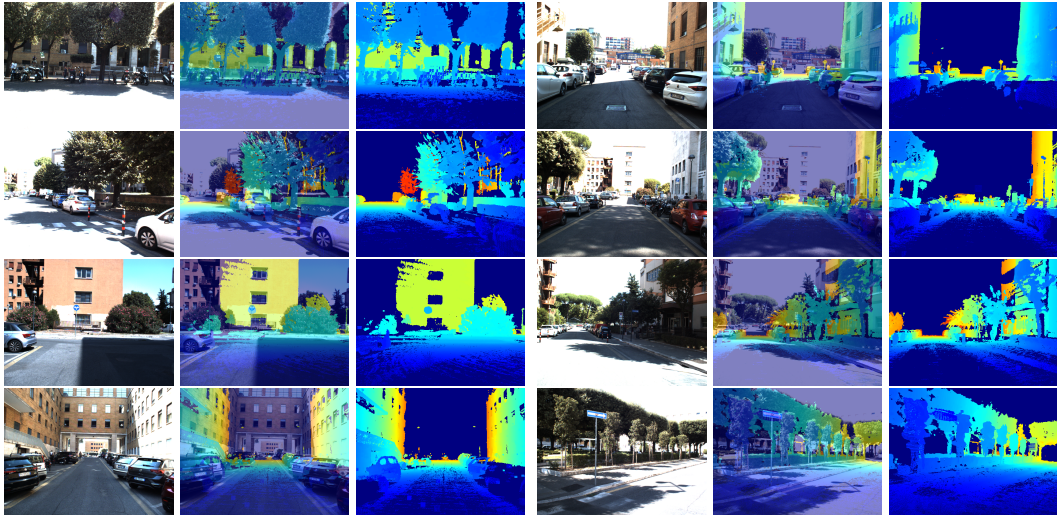
Photometric error is very important to refine optimization at sub-pixel level, given good initial guess provided by the geometrical optimization. The photometric error at image coordinates  $\mathbf{u}$  in the matching pair is the difference between  $\mathcal{I}_i^g(\mathbf{u})$  and the pixel  $\mathcal{I}_j^g(\mathbf{u}')$  of the second image:

$$\mathbf{e}_{\mathbf{u}}^{\text{photo}}(\mathbf{X}_i, \mathbf{X}_j) = \sum_c \left( \zeta^c(\mathbf{X}_{j,i}, \mathcal{I}_i^c) - \mathcal{I}_j^c(\mathbf{u}') \right) \quad (8.3)$$

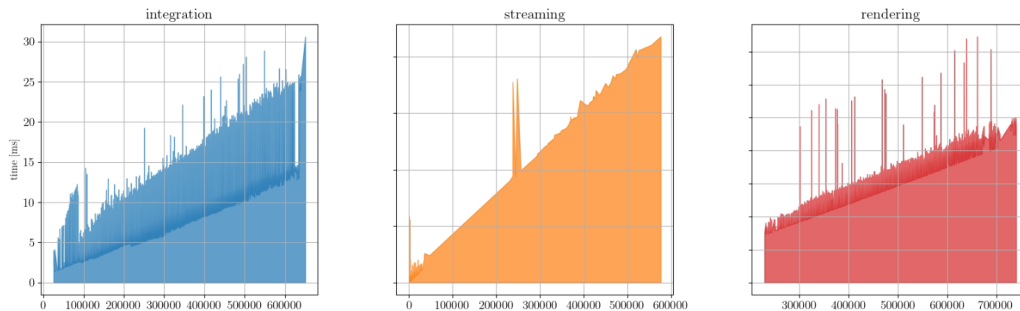
where  $\mathbf{u}' = \pi \left( \mathbf{R}_j^T (\mathbf{R}_i \pi^{-1}(\mathbf{u}, d) + \mathbf{t}_i - \mathbf{t}_j) \right)$  and  $\zeta^c(\cdot)$  is the mapping function (see Sec. 6.2.2 for more details). At each iteration, we remove the occluded portions of the images before evaluating Eq. (8.3). The optimization proceeds by seeking the optimum of all poses. The major difference compared to the previously presented approach illustrated in Sec. 6.2.2 is that this one does not employ hierarchical optimization and uses only depth and grayscale/intensity channels, not surface normals ( $c = (g, d)$ ). Hierarchical optimization and surface normals can be omitted in the photometric optimization, thanks to the geometric contribution, which increases the convergence basin and accuracy of the initial guess.

## 8.3 Experimental Evaluation

In this section, we offer a qualitative look at the outcomes derived from our multimodal approach. Our focus is primarily on the sequential optimization of LiDAR using geometric



**Figure 8.5.** Qualitative samples of depth renderings. The samples have been captured from *Campus* sequence (motion by car) and rendered using technique discussed in Sec. 8.2.1. The depth is thresholded at 50 meters.



**Figure 8.6.** Voxelhashing runtimes. From left to right respectively integration, streaming and rendering runtimes. The  $y$ -axis reports time in ms, the  $x$ -axis the number of hash buckets involved in the process.

residuals and the camera with photometric residuals, as this combination emerged as a notably effective fusion strategy. We also provide insights into the runtime of our algorithms, highlighting the impact in terms of time of using combined data. While this section is not designed to provide a comprehensive set of experiments or quantitative evaluations, it does aim to demonstrate the broad benefits of multimodality, particularly in large-scale settings. Given the absence of a direct counterpart for comparison, we benchmark against COLMAP [106, 107], a widely recognized tool in 3D reconstruction. It is worth noting that COLMAP addresses a more intricate challenge, operating solely on a sequence of images captured by cameras. Differently, we employ images generated by LiDAR as well as images obtained from cameras. Our intention with these results is to offer readers a perspective on the scale of runtimes and robustness, comparing multimodal 3D reconstruction against more traditional vision methodologies. Our experiments were conducted on a PC equipped with an Intel Core i9-13900KF CPU @ 5.80GHz, having 128GB of RAM and a Geforce GTX 4090 X 24G graphics card. Our photometric BA scheme is implemented in CUDA 11. We perform

ours		COLMAP [106, 107]		
	791 images		791 images	6807 images
MD-SLAM [33]	5.523	Feature extraction	0.114	2.798
Depth rendering	20.712	Feature matching	0.194	1.717
Bundle Adjustment	0.641 + 0.161	Bundle Adjustment	30.720	188.666
Total	<b>27.037</b>	Total	31.028	193.181

**Table 8.1.** Multimodal 3D reconstruction runtimes compared to COLMAP (in minutes). On the left are our runtimes, where Bundle Adjustment refers to the combined times of geometric (first) and photometric (second) minimization, as detailed in Sec. 8.2.2. On the right, we have the runtimes for COLMAP. It is worth noting that 3D reconstruction relying solely on a camera is inherently more complex and time-consuming (i.e. structure optimization). The purpose here is not a direct comparison but rather to provide a general sense of how our multimodal approach stacks up against traditional vision techniques. COLMAP optimization with 791 images fails and results are reported in Fig. 8.7a. COLMAP optimization with 6807 images fails again and results are reported in Fig. 8.7b

some experiments on a *Campus* sequences, from our VBR benchmark (Chapter 7). This sequence contains 6807 synchronized rgb images and LiDAR point clouds. This particular sequence was chosen without specific criteria; it is simply one of the available options. It does not pose extreme challenges for traditional methods, as it captures car movement at 50 Km/h within an urban setting and is about 1.5 Km long.

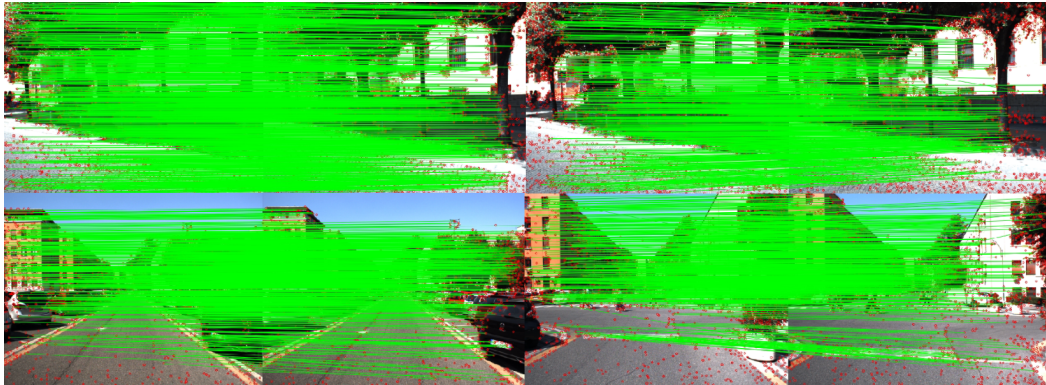
For our 3D reconstruction optimization, we employed the `srrg2_solver` [49] with the LM method as iterative strategy. The optimization ceases when the magnitude difference of the error between iterations falls below  $10^{-4}$ , for both LiDAR geometric and RGB-D photometric.

### 8.3.1 Qualitative results

In this section we show some qualitative results of our approach, namely: geometric optimization, photometric and dense depth renderings. In addition, we show some COLMAP qualitative results when running in sequential mode (not exhaustive matching between frames, since data is ordered) to create sparse, feature-based 3D reconstruction model. We solve exactly the same problem but in different ways, since the input data is different: COLMAP uses only images captured by camera, we employ both LiDAR and camera measurements together. Our results are depicted in Fig. 8.3 and Fig. 8.4, illustrating the initial state - before global optimization, mid-process - after LiDAR geometric optimization, and final outcome - after dense photometric camera refinement. The comprehensive reconstruction utilized 791 LiDAR scans and 791 RGB images, we downsample the original data for a more immediate solution. The optimization process was approximately completed in 10 iterations for both the geometric and photometric minimization. This reconstruction employs approximately 3 Gb of RAM and 16 Gb of DRAM. Using the original 3D model, to employ photometric optimization on cameras, we rendered dense depths using the method outlined in Sec. 8.2.1. This process utilizes 20 Gb of DRAM and 20 Gb of RAM, taking approximately 20 minutes to process the entire dataset. Qualitative examples of the generated depths are showcased in Fig. 8.5, which displays the RGB, the rendered dense depth, and the overlaid image. The runtimes of voxelhashing are plotted in Fig. 8.6. Our implementation of



(a) Result of COLMAP with 791 images (fail). (b) Result of COLMAP with 6807 images (fail).



(c) SIFT [78] matches from COLMAP. Despite the good amount of matches the 3D reconstruction pipeline seems to fail.

**Figure 8.7.** Qualitative COLMAP results in one of the *Campus* sequence. It consistently fails, regardless of the number of images or the quality of matches.

voxelhashing has runtimes that grows linearly with the number of hash buckets allocated in memory, still with 600k buckets allocated, around 153M voxels integrated, the runtimes are below 30 ms for integration and bidirectional streaming. For depth rendering with similar quantities we are below 20 ms. The total time required for the full process is of about 30 minutes (Tab. 8.1). We tried COLMAP with two configurations, a subsampled one using the same number of frames we used for our reconstruction and another one using all frames synchronized with the LiDAR at 20Hz. Given the higher number of matches and overlap between frames, depicted in Fig. 8.7c, COLMAP fails after some time in both configurations. More details runtimes are shown in Tab. 8.1.

## 8.4 Conclusion

Throughout this chapter, we have explored the combined power of LiDAR and camera sensors in enhancing 3D reconstruction. We have seen how LiDAR’s strength in capturing depth, complements the camera’s ability to provide rich visual details. When these two are fused, the result is a more detailed 3D representation than what can be achieved using only one of them. This integrated approach also simplifies the typically complex process of vision-based 3D reconstructions, as the precision of LiDAR measurements reduces the need for extensive structure optimization. While many studies have approached these sensors separately, we have proposed a unified approach. Through our BA strategy, we have highlighted the potential improvements in 3D reconstructions by utilizing both photometric and geometric data, generating efficiently dense depths. However, it is important to note that

---

this chapter does not dive deep into detailed evaluations or extensive experiments. Our main goal here was to showcase the benefits of using both modalities together and the potential of our integrated approach. Looking ahead, it is clear that combining different sensors in a cohesive framework holds great promise for the future of 3D reconstructions, offering improved robustness, accuracy and efficiency in a compact and cohesive implementation.



## Chapter 9

# Conclusion

### 9.1 Summary

In this thesis, we presented several contributions towards uniform SLAM and 3D reconstructions by harnessing the similarities between LiDAR and cameras. These advancements employ each sensor both separately and in conjunction. Additionally, we introduced calibration techniques between these two sensors using commercial tags and a new challenging public benchmark, laying the groundwork for multimodal 3D geometric reconstructions and more robust and accurate SLAM algorithms.

Chapter 3 evaluates VPR techniques adapted for 3D LiDAR intensity data converted into images. Using various robotic/vision datasets, we found that these methods reliably detect loops, suggesting potential for broader LiDAR-only SLAM applications. Although they might slightly lag behind RGB image methods, they maintain consistent SLAM performance under different lighting conditions, eliminating the need for intricate RGB algorithms.

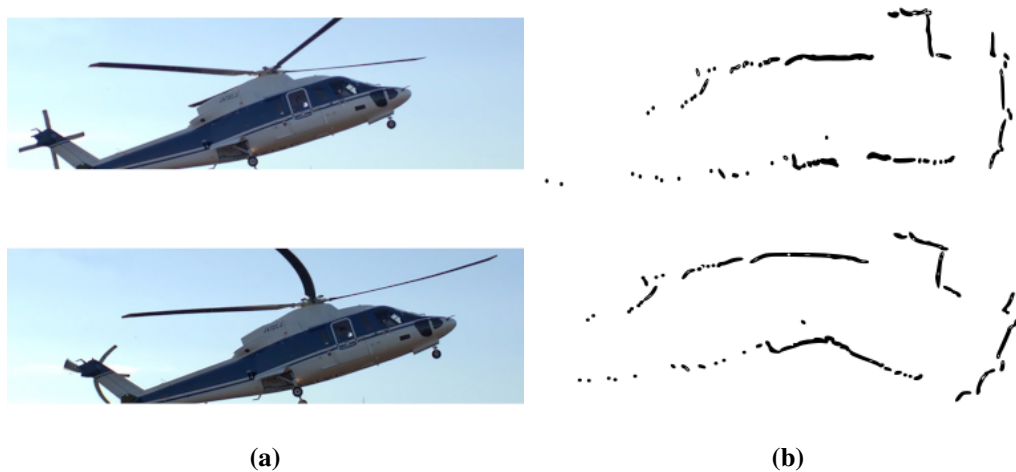
In Chapter 4, we present a direct SLAM system for both LiDAR and RGB-D, challenging the convention of separate systems for distinct sensors. Merging position tracking and appearance-based relocalization, our approach addresses extensive loop closures while also offering a more generalized solution for depth sensors. Complementing this in Chapter 6, we introduce a unified photometric BA method, refining SLAM-derived trajectories for maximized global photometric consistency. Both implementations were released as open source.

In Chapter 5, the main contribution is an extrinsic calibration technique for aligning LiDAR and camera. This is fundamental for systems that employ sensors together. While traditional robotics methods use large, expensive tags to address LiDAR sparseness, we utilize compact markers like A3 chessboards. The implementation was released as open-source.

In Chapter 6, using a precise calibration of both sensors, we demonstrate that our unified photometric BA strategy, which integrates both LiDAR and RGB-D, enhances performance. This joint approach leverages the advantages of each: LiDAR for a broader convergence basin and camera for dense sub-pixel resolution.

In Chapter 7, we offer a comprehensive robotics perception dataset from Rome, including RGB, dense depth, 3D LiDAR point clouds, IMU, and GPS data. Taking into account the shortcomings of current datasets and acknowledging the precision achieved by modern SLAM and 3D reconstruction algorithms on current benchmarks, our new challenging data





**Figure 9.1.** Distorted images and maps. On the left, the impact of an image taken with a rolling shutter camera (at the bottom) is contrasted with one taken using a global shutter. To the right, the distinction between a “skewed” point cloud and its rectified counterpart is displayed, presented in 2D for clarity. Image acknowledgments go to [1] and [102], respectively.

aims to enhance algorithmic performance and prevent overfitting. We prioritize precise calibration and synchronization while capturing diverse environments such as multi-floor buildings, indoor-outdoor, gardens, and highways using cutting-edge tools. Gathered both handheld and via vehicles, our dataset suits various robotic applications. We provide an accurate ground truth, built upon findings illustrated in previous chapters, globally optimizing RTK-GPS readings and LiDAR point clouds. The dataset, divided into training and validation sets, is available at [www.rvp-group.net/slam-dataset](http://www.rvp-group.net/slam-dataset).

In the closing Chapter 8, we have highlighted the synergy of LiDAR and camera sensors for enhanced 3D reconstruction. Merging LiDAR’s depth precision with the camera’s visual richness streamlines the reconstruction process. Our unified BA strategy demonstrates the advantages of using both modalities. Rather than diving into detailed evaluations, our aim was to showcase this integrated approach’s potential in terms of robustness and runtimes. Two sensors is better than one: the fusion of LiDAR and camera sensors presents a promising future for efficient, robust and accurate 3D reconstructions algorithms.

## 9.2 Outlook

Although the author hopes that some aspects of this thesis may retain relevance in the future, the vision of an optimal 3D reconstruction system, as introduced earlier, appears elusive for now. This section endeavors to highlight potential avenues of future research that might pave the way towards realizing that vision.

**Tightly-coupled multimodal SLAM.** In this thesis, we have explored how cameras and LiDARs can be treated similarly by leveraging the commonalities in their measurements. At present, our approach relies on having dense depth data for the camera. Instead of rendering this offline and then utilizing it as proposed in Chapter 8, a more integrated solution would benefit from the incremental LiDAR model during sensor tracking or SLAM. This can potentially be applied in small dense patches for photometric refinement in the



passive image space. By adopting this method, even during ego-motion estimation, one could harness the strengths of both sensors: the extensive support from LiDAR to widen the convergence basin using geometry and the greater photometric accuracy from one or multiple cameras.

**Rolling shutter and “skewing” effects.** As delineated in the background section (Sec. 2.2.7), cameras predominantly employ either global shutters, which consistently capture entire scenes at once, or rolling shutters, sequentially exposing pixels and potentially inducing distortions. These distortions are particularly pronounced in hand-held devices due to unavoidable micro-movements, complicating 3D reconstruction. Notably, even minor rotational shifts intensify the rolling shutter effect. Fig. 9.1a shows the differences between images captured by the two different camera types. Mechanical LiDARs manifest a skewing phenomenon reminiscent of the rolling shutter observed in cameras. They rapidly record vertical data, but horizontal data acquisition is contingent on the encoder’s rotational speed. The intensity of this skewing correlates directly with the sensor’s motion velocity. The parallels between the behaviors of the two sensors are evident. Fig. 9.1b shows the differences between a distorted map and one compensated through some geometric techniques [102]. For precise and rectified mapping, addressing these distortions is fundamental. Given the shared characteristics between both sensors, a symmetrical rectification approach might be viable. A prospective avenue worth exploring could involve refining the sensor intrinsics individually for each image row or column.

**Include semantic information.** Incorporating semantic understanding into 3D reconstruction offers the potential for substantial advancements. In many real-world scenarios, certain elements of a scene may not provide strong signals for direct reconstruction due to their homogeneous textures, lack of distinct features or noise. However, by leveraging semantic segmentation, it is possible to classify and identify these elements, such as walls or objects. This enhanced understanding would empower reconstruction algorithms to interpret even weak cues accurately, drawing upon prior knowledge about the typical structures of certain elements. Thus, integrating semantics becomes pivotal for achieving comprehensive and accurate reconstructions, especially in environments with passive sensing.

**City-Scale SLAM.** Chapter 4 delves into a uniform SLAM system tailored for RGB-D cameras and LiDAR, optimized for substantial (few Km) yet not full city-scale trajectories (scale of 50 Km). To transition from this large-scale foundation to a comprehensive city-scale landscape, several enhancements are in order. For starters, sensor tracking reliability is paramount. The likelihood of system failures needs diminishing. While integrating IMU measurements and incorporating multiple sensor views is a conventional strategy, it is crucial in this scenario. Notably, IMU data can bolster resilience against moving objects. Moreover, post-failure re-localization is essential. While this is an extensively studied domain, unique challenges like discrepancies between visual and inertial data (such as when using the lift) necessitate innovative solutions. Furthermore, transitioning to city-scale demands scalability in the SLAM system. Standard methodologies, such as windowed bundle adjustment (like the one employed by [86]), can offer partial solutions. However, challenges like ensuring trajectory integrity against incorrect loop closure detections might still be unresolved in this vast landscape and would be fundamental to reach full autonomy.

**Deformable world and dynamics.** In the context of this thesis, the foundational assumption has been that of a static environment, with any moving entities deemed as outlying observations. Such an approach, while practical for certain scenarios, inherently limits the system’s understanding of dynamic environments. A more encompassing system

would proactively recognize and incorporate these moving elements, facilitating a richer and more realistic reconstruction. This becomes particularly crucial when we consider not just rigid motions, like cars moving on a road, but also deformable objects, such as humans in motion. Their inherent complexities in movement patterns and deformations add layers of intricacy to the reconstruction process. Recent research, such as the work by [71, 88], has begun to explore these challenges, marking early steps towards a comprehensive solution. However, bridging the gap between these initial explorations and a fully matured, robust system for dynamic 3D reconstruction still demands significant research efforts.

**Implicit rendering.** In the evolving landscape of 3D reconstruction, Neural Radiance Fields (NeRFs) have emerged as a pivotal methodology. NeRFs utilize deep neural networks to directly infer a continuous volumetric scene from a sparse set of 2D images. They represent the radiance as a function of the 3D location and the viewing direction. This representation has proven adept at delivering high-quality novel views of diverse scenes, often capturing intricate details with remarkable accuracy. A notable strength of NeRFs lies in their ability to preserve these details due to their continuous volumetric representation and implicit rendering. Furthermore, they offer a compact representation where the entire scene's information is efficiently encoded into the neural network's weights. Parallel to this, Gaussian Splatting has also been explored in the realm of 3D reconstruction. It is a technique that emphasizes the influence of specific samples in volumetric spaces. By using Gaussian weights, this method ensures that the contribution of a sample decreases with its distance from a query point. Gaussian splatting can be particularly beneficial for handling and merging data from multiple sensors or viewpoints, ensuring smooth and seamless reconstructions, faster compared to NeRFs. While these techniques are both at the forefront of current research, their full potential and employment in the areas of fast and large 3D reconstruction remains a domain of active exploration.

# Appendices



## Appendix A

# Appendix

### A.1 Jacobian derivation of Photometric Bundle Adjustment

In this section, we discuss the photometric error term from Eq. (6.3) in more detail and provide analytic expressions for the Jacobian matrices. This is an additional section that we write for completeness.

We represent as  $\Delta \mathbf{x}$  the Lie algebra  $\mathfrak{se}(3)$  associated with the group  $\mathbb{SE}(3)$ , parameterized as  $\Delta \mathbf{x} = [\Delta \mathbf{t}, \Delta \mathbf{q}]^T$ .  $\Delta \mathbf{t} \in \mathbb{R}^3$  is the translation, and  $\Delta \mathbf{q} \in \mathbb{R}^3$  is the imaginary part of a unit quaternion. The rotation matrix can be calculated from the perturbation vector using the *exponential map* at the identity  $\Delta \mathbf{R} = \exp(\Delta \mathbf{q})$ . We extend the notation of the exponential map to refer to the transformation encoded in  $\Delta \mathbf{x}$ . Let  $\Delta \mathbf{X} = \exp(\Delta \mathbf{x})$ , be this transformation whose rotation is  $\Delta \mathbf{R}$  and translation  $\Delta \mathbf{t}$ . We use the  $\boxplus$  operator to denote applying a perturbation to a transform  $\mathbf{X} \exp(\Delta \mathbf{x}) := \mathbf{X} \boxplus \Delta \mathbf{x}$ .

In the reminder, the error term differs slightly from the one presented in Eq. (6.3). Here we insert the offset mapping RGB-D or LiDAR with respect to the reference frame of a multi-device sensor platform. This is fundamental to fuse the two sensors as illustrated in Sec. 6.3.3. The constant rigid transformation comprises rotation  $\mathbf{R}_o$  and translation  $\mathbf{t}_o$ .

For compactness, we define two quantities  $\mathbf{p}_u$  as the point transformed by this offset and  $\bar{\mathbf{p}}_u$  as the point  $\mathbf{p}_u$  transformed by the estimate and the inverse of the offset as follows:

$$\mathbf{p}_u = \mathbf{R}_o \pi^{-1}(\mathbf{u}, d) + \mathbf{t}_o, \quad (\text{A.1})$$

$$\bar{\mathbf{p}}_u = \mathbf{R}_o^T \left( \mathbf{R}_j^T (\mathbf{R}_i \mathbf{p}_u + \mathbf{t}_i - \mathbf{t}_j) - \mathbf{t}_o \right). \quad (\text{A.2})$$

Thus, we can rewrite our error in the following way:

$$\mathbf{e}_u^c = \zeta^c(\mathbf{X}_{j,i}, \mathcal{I}_i^c(\mathbf{u})) - \mathcal{I}_j^c(\mathbf{u}') = \zeta^c(\bar{\mathbf{p}}_u) - \mathcal{I}_j^c(\pi(\bar{\mathbf{p}}_u)). \quad (\text{A.3})$$

Applying the perturbations  $\Delta \mathbf{x}_i$  and  $\Delta \mathbf{x}_j$  on the right hand side in Eq. (A.3) leads to:

$$\mathbf{e}_u^c(\mathbf{X}_i \boxplus \Delta \mathbf{x}_i) = \zeta^c(\bar{\mathbf{p}}_{u|i}) - \mathcal{I}_j^c(\pi(\bar{\mathbf{p}}_{u|i})), \quad (\text{A.4})$$

$$\mathbf{e}_u^c(\mathbf{X}_j \boxplus \Delta \mathbf{x}_j) = \zeta^c(\bar{\mathbf{p}}_{u|j}) - \mathcal{I}_j^c(\pi(\bar{\mathbf{p}}_{u|j})). \quad (\text{A.5})$$

with  $\bar{\mathbf{p}}_{u|i}$  and  $\bar{\mathbf{p}}_{u|j}$  respectively defined as:

$$\bar{\mathbf{p}}_{u|i} = \mathbf{R}_o^T \left( \mathbf{R}_j^T \left( \mathbf{R}_i \left( \mathbf{R}(\Delta \mathbf{q}) \mathbf{p}_u + \Delta \mathbf{t} \right) + \mathbf{t}_i - \mathbf{t}_j \right) - \mathbf{t}_o \right), \quad (\text{A.6})$$

$$\bar{\mathbf{p}}_{u|j} = \mathbf{R}_o^T \left( \mathbf{R}(-\Delta \mathbf{q}) \mathbf{R}_j^T \left( \mathbf{R}_i \mathbf{p}_u + \mathbf{t}_i - \mathbf{t}_j \right) - \Delta \mathbf{t} - \mathbf{t}_o \right). \quad (\text{A.7})$$

Deriving these two quantities with respect to the perturbations leads to the following Jacobians:

$$\frac{\partial \bar{\mathbf{p}}_{u|i}}{\partial \Delta \mathbf{x}_i} = \mathbf{R}_o^T \mathbf{R}_j^T \mathbf{R}_i \left[ \mathbf{I}_{3 \times 3} \quad 2[\mathbf{p}_u]_{\times} \right], \quad (\text{A.8})$$

$$\frac{\partial \bar{\mathbf{p}}_{u|j}}{\partial \Delta \mathbf{x}_j} = -\mathbf{R}_o^T \left[ \mathbf{I}_{3 \times 3} \quad 2[\mathbf{R}_j^T (\mathbf{R}_i \mathbf{p}_u + \mathbf{t}_i - \mathbf{t}_j)]_{\times} \right]. \quad (\text{A.9})$$

Projective Jacobians depends on the projection model, differing RGB-D and LiDAR, these are calculated respectively deriving Eq. (2.4) and Eq. (2.10) with respect to the transformed point  $\bar{\mathbf{p}}_u$ .

**Pinhole model:**

$$\frac{\partial \pi_p(\bar{\mathbf{p}}_u)}{\partial \bar{\mathbf{p}}_u} = \frac{\partial \phi(\mathbf{K} \bar{\mathbf{p}}_u)}{\partial \bar{\mathbf{p}}_u} = \frac{1}{v_z^2} \begin{bmatrix} v_z & 0 & -v_x \\ 0 & v_z & -v_y \end{bmatrix} \Big|_{[v_x, v_y, v_z] = \mathbf{K} \bar{\mathbf{p}}_u} \mathbf{K}. \quad (\text{A.10})$$

**Spherical model:**

$$\frac{\partial \pi_s(\bar{\mathbf{p}}_u)}{\partial \bar{\mathbf{p}}_u} = \frac{\partial \mathbf{K}_{[1,2]} \psi(\bar{\mathbf{p}}_u)}{\partial \bar{\mathbf{p}}_u} = \mathbf{K}_{[1,2]} \left[ \begin{array}{c} \frac{1}{v_x^2 + v_y^2} \begin{bmatrix} -v_y & v_x & 0 \end{bmatrix} \\ \frac{1}{v_x^2 + v_y^2 + v_z^2} \begin{bmatrix} -\frac{v_x v_z}{\sqrt{v_x^2 + v_y^2}} & -\frac{v_y v_z}{\sqrt{v_x^2 + v_y^2}} & \sqrt{v_x^2 + v_y^2} \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \end{array} \right] \Big|_{[v_x, v_y, v_z] = \bar{\mathbf{p}}_u}. \quad (\text{A.11})$$

Image jacobians are numerically computed for each channel  $c$  with pixel-wise derivation:

$$\begin{aligned} \frac{\partial \mathcal{I}_{r,c}^c}{\partial r} &= \frac{1}{2} \left( \mathcal{I}_{r+1,c}^c - \mathcal{I}_{r-1,c}^c \right) \\ \frac{\partial \mathcal{I}_{r,c}^c}{\partial c} &= \frac{1}{2} \left( \mathcal{I}_{r,c+1}^c - \mathcal{I}_{r,c-1}^c \right) \end{aligned} \quad (\text{A.12})$$

Jacobians on the mapping function  $\zeta^c$  for each channel grayscale/intensity, range/depth and normals, respectively  $\{g, d, n\}$  are computed also with respect to perturbations, since the estimates appears also to the left of the error function Eq. (A.3). In the reminder we specifically write mapping function and Jacobians of each cue.

**Intensity**

The mapping function does not affect the intensity/grayscale channel, therefore we have:

$$\zeta^g(\bar{\mathbf{p}}_u) = \mathcal{I}^g(\bar{\mathbf{p}}_u), \quad \frac{\partial \zeta^g(\bar{\mathbf{p}}_{u|i})}{\partial \Delta \mathbf{x}_i} = 0, \quad \frac{\partial \zeta^g(\bar{\mathbf{p}}_{u|j})}{\partial \Delta \mathbf{x}_j} = 0. \quad (\text{A.13})$$

We need to differ between the two depth sensor models, since RGB-D provides depth while LiDAR range measurements. Hence, leading to different Jacobians.

### Depth

$$\zeta^d(\bar{\mathbf{p}}_u) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{p}}_u, \quad \frac{\partial \zeta^d(\bar{\mathbf{p}}_{u|i})}{\Delta \mathbf{x}_i} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \frac{\partial \bar{\mathbf{p}}_{u|i}}{\partial \Delta \mathbf{x}_i}, \quad \frac{\partial \zeta^d(\bar{\mathbf{p}}_{u|j})}{\Delta \mathbf{x}_j} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \frac{\partial \bar{\mathbf{p}}_{u|j}}{\partial \Delta \mathbf{x}_j}. \quad (\text{A.14})$$

### Range

$$\zeta^d(\bar{\mathbf{p}}_u) = \|\bar{\mathbf{p}}_u\|, \quad \frac{\partial \zeta^d(\bar{\mathbf{p}}_{u|i})}{\Delta \mathbf{x}_i} = \frac{\bar{\mathbf{p}}_u^T}{\|\bar{\mathbf{p}}_u\|} \frac{\partial \bar{\mathbf{p}}_{u|i}}{\partial \Delta \mathbf{x}_i}, \quad \frac{\partial \zeta^d(\bar{\mathbf{p}}_{u|j})}{\Delta \mathbf{x}_j} = \frac{\bar{\mathbf{p}}_u^T}{\|\bar{\mathbf{p}}_u\|} \frac{\partial \bar{\mathbf{p}}_{u|j}}{\partial \Delta \mathbf{x}_j}. \quad (\text{A.15})$$

$$(\text{A.16})$$

### Normals

Normals, differently, affect only rotation, hence is convenient to rewrite our error function expressed in Eq. (A.3). This reduces to:

$$\mathbf{e}_u^n = \zeta^n \left( \mathbf{R}_o^T \mathbf{R}_j^T \mathbf{R}_i \mathbf{R}_o \mathbf{n}_u \right) - \mathcal{I}^n \left( \pi \left( \bar{\mathbf{p}}_u \right) \right). \quad (\text{A.17})$$

It is trivial to derive Eq. (A.3) with respect to the angular parts of the perturbations. Note that  $\mathbf{n}_u$  is the normal prior to any transformation, since everything is enrolled in Eq. (A.17).

$$\frac{\partial \zeta^n \left( \mathbf{R}_o^T \mathbf{R}_j^T \mathbf{R}_i \mathbf{R}_o \mathbf{n}_u \right)}{\Delta \mathbf{q}_i} = 2 \mathbf{R}_o^T \mathbf{R}_j \mathbf{R}_i \mathbf{R}_o [\mathbf{n}_u]_{\times}, \quad (\text{A.18})$$

$$\frac{\partial \zeta^n \left( \mathbf{R}_o^T \mathbf{R}(-\Delta \mathbf{q}_j) \mathbf{R}_j^T \mathbf{R}_i \mathbf{R}_o \mathbf{n}_u \right)}{\Delta \mathbf{q}_j} = -2 \mathbf{R}_o^T [\mathbf{R}_j^T \mathbf{R}_i \mathbf{R}_o \mathbf{n}_u]_{\times}. \quad (\text{A.19})$$

Finally, we can reconstruct the full jacobians:

$$\mathbf{J}_i = \frac{\partial \zeta^c(\bar{\mathbf{p}}_{u|i})}{\Delta \mathbf{x}_i} - \frac{\partial \mathcal{I}_{r,c}^c}{\partial r, c} \frac{\partial \pi(\bar{\mathbf{p}}_u)}{\partial \bar{\mathbf{p}}_u} \frac{\partial \bar{\mathbf{p}}_{u|i}}{\partial \Delta \mathbf{x}_i}, \quad (\text{A.20})$$

$$\mathbf{J}_j = \frac{\partial \zeta^c(\bar{\mathbf{p}}_{u|j})}{\Delta \mathbf{x}_j} - \frac{\partial \mathcal{I}_{r,c}^c}{\partial r, c} \frac{\partial \pi(\bar{\mathbf{p}}_u)}{\partial \bar{\mathbf{p}}_u} \frac{\partial \bar{\mathbf{p}}_{u|j}}{\partial \Delta \mathbf{x}_j}. \quad (\text{A.21})$$





# Bibliography

- [1] Differences between rolling and global shutter cameras. <https://www.premiumbeat.com/blog/know-the-basics-of-global-shutter-vs-rolling-shutter/>. Accessed: 30-10-2023. xvii, 100
- [2] Intel realsense lidar camera l515. <https://www.intelrealsense.com/lidar-camera-l515>. Accessed: 18-10-2023. xi, 10, 12
- [3] Intel realsense d405. <https://www.intelrealsense.com/depth-camera-d405>. Accessed: 18-10-2023. xi, 10, 12
- [4] Livox mid-70. <https://www.livoxtech.com/mid-70>. Accessed: 18-10-2023. xii, 13
- [5] Microsoft kinect. <https://azure.microsoft.com/it-it/products/kinect-dk>. Accessed: 18-10-2023. xi, 10, 12
- [6] Ouster os1-128. <https://ouster.com/products/scanning-lidar/os1-sensor>. Accessed: 18-10-2023. xii, 13
- [7] Velodyne puck. <https://velodynelidar.com/products/puck>. Accessed: 18-10-2023. xii, 13
- [8] 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). 63
- [9] AMANATIDES, J. AND WOO, A. A fast voxel traversal algorithm for ray tracing. In *Eurographics*, vol. 87, pp. 3–10. Citeseer (1987). 90
- [10] ARANDJELOVICAND, R., GRONAT, P., TORII, A., PAJDLA, T., AND SIVIC, J. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 5297–5307 (2016). 31
- [11] BAY, H., ESS, A., TUYTELAARS, T., AND GOOL, L. V. Speeded-up robust features (SURF). *Journal of Computer Vision and Image Understanding (CVIU)*, **110** (2008), 346. 31
- [12] BAY, H., TUYTELAARS, T., AND GOOL, L. V. Surf: Speeded up robust features. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pp. 404–417 (2006). 33, 41

- [13] 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)* (2018). 29, 41, 46, 49, 70, 71
- [14] BELTRÁN, J., GUINDEL, C., DE LA ESCALERA, A., AND GARCÍA, F. Automatic extrinsic calibration method for lidar and camera sensor setups. *IEEE Trans. on Intelligent Transportation Systems (ITS)*, **23** (2022), 17677. doi:10.1109/TITS.2022.3155228. xix, 51, 53, 57, 58
- [15] BRADSKI, G. AND KAEHLER, A. Opencv. *Dr. Dobb's journal of software tools*, **3** (2000). 33, 80
- [16] BRIZI, L., GIACOMINI, E., DI GIAMMARINO, L., FERRARI, S., SALEM, O., DE REBOTTI, L., AND GRISSETTI, G. Vbr: A vision benchmark in rome. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*. IEEE (2024). 5
- [17] BROWN, D. Decentering distortion of lenses. *Photogrammetric Engineering*, **32** (1996), 444. 59
- [18] BURNETT, K., ET AL. Boreas: A multi-season autonomous driving dataset. *Intl. Journal of Robotics Research (IJRR)*, **42** (2023), 33. Available from: <https://doi.org/10.1177/02783649231160195>, doi:10.1177/02783649231160195. 75
- [19] BURRI, M., NIKOLIC, J., GOHL, P., SCHNEIDER, T., REHDER, J., OMARI, S., ACHELNIK, M. W., AND SIEGWART, R. The euroc micro aerial vehicle datasets. *Intl. Journal of Robotics Research (IJRR)*, **35** (2016), 1157. 77, 78
- [20] CAMPOS, C., ELVIRA, R., RODRÍGUEZ, J. J. G., 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. 85
- [21] CHEN, X., LÄBE, T., MILIOTO, A., RÖHLING, T., VYSOTSKA, O., HAAG, A., BEHLEY, J., AND STACHNISS, C. OverlapNet: Loop Closing for LiDAR-based SLAM. In *Proc. of Robotics: Science and Systems (RSS)* (2020). Available from: <https://www.ipb.uni-bonn.de/wp-content/papercite-data/pdf/chen2020rss.pdf>. 31
- [22] CHEN, Y. AND MEDIONI, G. Object modelling by registration of multiple range images. *Image and Vision Computing*, **10** (1992), 145. 71, 92
- [23] COP, K. P., BORGES, P. V., AND DUBÉ, R. Delight: An efficient descriptor for global localisation using lidar intensities. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 3653–3660 (2018). 31, 41
- [24] CORTE, B. D., BOGOSLAVSKYI, I., STACHNISS, C., AND GRISSETTI, 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, 40, 41, 66
- [25] CUMMINS, M. AND NEWMAN, P. Highly scalable appearance-only SLAM - FAB-MAP 2.0. In *Proc. of Robotics: Science and Systems (RSS)* (2009). 31

- [26] DAI, A., NIESSNER, M., ZOLLHÖFER, M., IZADI, S., AND THEOBALT, C. Bundle-fusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans. on Graphics*, **36** (2017), 1. 63, 68, 70
- [27] 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). 62
- [28] DELLENBACH, P., DESCHAUD, J., 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. IEEE (2022). 20
- [29] 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). 63
- [30] DESCHAUD, J. Imls-slam: scan-to-model matching based on 3d data. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 2480–2485 (2018). 29
- [31] DETONE, D., MALISIEWICZ, T., AND RABINOVICH, A. Superpoint: Self-supervised interest point detection and description. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPR)*, pp. 224–236 (2018). 31, 33
- [32] DI GIAMMARINO, L., ALOISE, I., STACHNISS, C., AND GRISETTI, G. Visual place recognition using lidar intensity information. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4382–4389 (2021). 5, 41, 45
- [33] 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). xiv, 5, 62, 63, 68, 70, 71, 92, 95
- [34] DI GIAMMARINO, L., GIACOMINI, E., BRIZI, L., SALEM, O., AND GRISETTI, G. Photometric lidar and rgb-d bundle adjustment. *IEEE Robotics and Automation Letters (RA-L)*, (2023). 5, 81, 82
- [35] DORNAIKA, F. AND HORAUD, R. Simultaneous robot-world and hand-eye calibration. *IEEE Trans. on Robotics and Automation*, **14** (1998), 617. 80
- [36] DUBÉ, R., DUGAS, D., STUMM, E., NIETO, J., SIEGWART, R., AND CADENA, C. Segmatch: Segment based place recognition in 3d point clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 5266–5272 (2017). 31
- [37] ENGEL, J., SCHÖPS, T., AND CREMERS, D. Lsd-slam: Large-scale direct monocular slam. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pp. 834–849 (2014). 41
- [38] 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). 63

- [39] 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. doi:10.1109/ACCESS.2023.3247195. 53
- [40] 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. 53
- [41] FORSTER, C., ZHANG, Z., GASSNER, M., WERLBERGER, M., AND SCARAMUZZA, D. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Trans. on Robotics (TRO)*, **33** (2016), 249. 63
- [42] GÁLVEZ-LÓPEZ, D. AND TARDOS, J. D. Bags of binary words for fast place recognition in image sequences. *IEEE Trans. on Robotics (TRO)*, **28** (2012), 1188. 31, 32, 34, 41, 47
- [43] GARRIDO-JURADO, S., MUÑOZ-SALINAS, R., MADRID-CUEVAS, F., AND MARÍN-JIMÉNEZ, M. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, **47** (2014), 2280. Available from: <https://www.sciencedirect.com/science/article/pii/S0031320314000235>, doi:<https://doi.org/10.1016/j.patcog.2014.01.005>. 51
- [44] 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). 35, 36, 37, 75, 78, 83, 85
- [45] GEORGE, A. AND LIU, J. W. The evolution of the minimum degree ordering algorithm. *Siam review*, **31** (1989), 1. 27
- [46] GIACOMINI, E., BRIZI, L., DI GIAMMARINO, L., SALEM, O., PERUGINI, P., AND GRISETTI, G. Ca<sup>2</sup>lib: Simple and accurate lidar-rgb calibration using small common markers. *Sensors*, **24** (2024). doi:10.3390/s24030956. 5, 80
- [47] GOLDLÜCKE, B., AUBRY, M., KOLEV, K., AND CREMERS, D. A super-resolution framework for high-accuracy multiview reconstruction. *Intl. Journal of Computer Vision (IJCV)*, **106** (2014), 172. 62
- [48] GRAETER, J., WILCZYNSKI, A., AND LAUER, M. Limo: Lidar-monocular visual odometry. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 7872–7879. IEEE (2018). 88
- [49] 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. xiii, 21, 25, 40, 66, 95
- [50] GRISETTI, G., GUADAGNINO, T., ALOISE, I., COLOSI, M., DELLA CORTE, B., AND SCHLEGEL, D. Least squares optimization: From theory to practice. **9** (2020). Available from: <https://www.mdpi.com/2218-6581/9/3/51>, doi:10.3390/robotics9030051. 56

- [51] GRISETTI, G., KÜMMERLE, R., STRASDAT, H., AND KONOLIGE, K. g2o: A general framework for (hyper) graph optimization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 9–13 (2011). 41
- [52] GROSSBERG, M. D. AND NAYAR, S. K. A general imaging model and a method for finding its parameters. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, vol. 2, pp. 108–115. IEEE (2001). 9
- [53] GUO, J., BORGES, P. V., PARK, C., AND GAWEL, A. Local descriptor for robust place recognition using lidar intensity. *IEEE Robotics and Automation Letters (RA-L)*, **4** (2019), 1470. 31, 41
- [54] HARRIS, M. ET AL. Optimizing parallel reduction in cuda. *Nvidia developer technology*, **2** (2007), 70. 28
- [55] HIGHAM, N. J. *Accuracy and Stability of Numerical Algorithms*. SIAM (2002). 28
- [56] HORN, B., HILDEN, H., AND NEGAHDARIPOUR, S. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, **5** (1988), 1127. 46, 67, 85
- [57] HUBER, D. F. AND HEBERT, M. *Automatic three-dimensional modeling from reality*. Ph.D. thesis (2002). 30
- [58] IZADI, R. A. N. S., HILLIGES, O., MOLYNEAUX, D., KIM, D., DAVISON, A. J., KOHI, P., SHOTTON, J., HODGES, S., AND FITZGIBBON, A. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc. of the Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 127–136 (2011). 40
- [59] J. SERAFIN, E. O. AND GRISETTI, G. Fast and robust 3d feature extraction from sparse point clouds. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4105–4112 (2016). 41
- [60] JOHNSON, A. AND BROWN, B. A hybrid pipeline for 3d reconstruction using lidar and rgb data. In *3* (2020). 51
- [61] JOHNSON, A. E. Spin-images: a representation for 3-d surface matching. (1997). 30
- [62] KAESS, M., RANGANATHAN, A., AND DELLAERT, F. isam: Incremental smoothing and mapping. *IEEE Trans. on Robotics (TRO)*, **24** (2008), 1365. 26, 41
- [63] KANNALA, J. AND BRANDT, S. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, **28** (2006), 1335. doi:10.1109/TPAMI.2006.153. 59
- [64] KATZ, S., TAL, A., AND BASRI, R. Direct visibility of point sets. *ACM Trans. Graph.*, **26** (2007), 24–es. Available from: <https://doi.org/10.1145/1276377.1276407>, doi:10.1145/1276377.1276407. 59
- [65] KELLER, M., LEFLOCH, D., LAMBERS, M., AND IZADI, S. Realtime 3d reconstruction in dynamic scenes using pointbased fusion. In *Proc. of the International Conference on 3D Vision (3DV)*, pp. 1–8 (2013). 41

- [66] 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). 41, 46, 48, 68, 70
- [67] KIM, E.-S. AND PARK, S.-Y. Extrinsic calibration between camera and lidar sensors by matching multiple 3d planes. *IEEE Sensors Journal*, **20** (2020). Available from: <https://www.mdpi.com/1424-8220/20/1/52>, doi:10.3390/s20010052. xix, 54, 57, 58
- [68] 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. IEEE (2020). 75, 76, 78
- [69] 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). 62
- [70] 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). 57
- [71] LAMARCA, J., PARASHAR, S., BARTOLI, A., AND MONTIEL, J. Defslam: Tracking and mapping of deforming scenes from monocular sequences. *IEEE Trans. on Robotics (TRO)*, **37** (2020), 291. 102
- [72] LEUTENEGGER, S., CHLI, M., AND SIEGWART, R. Y. Brisk: Binary robust invariant scalable keypoints. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, pp. 2548–2555. IEEE (2011). 31, 33
- [73] 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. *IEEE Sensors Journal*, **22** (2022), 16472. doi:10.1109/JSEN.2022.3189041. 53
- [74] LIANG, X., CHEN, H., LI, Y., AND LIU, Y. Visual laser-slam in large-scale indoor environments. In *International Conference on Robotics and Biomimetics (ROBIO)*, pp. 19–24. IEEE (2016). 88
- [75] LINEGAR, C., CHURCHILL, W., AND NEWMAN, P. Work Smart, Not Hard: Recalling Relevant Experiences for Vast-Scale but Time-Constrained Localisation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)* (2015). 31
- [76] LIU, Z. AND ZHANG, F. Balm: Bundle adjustment for lidar mapping. *IEEE Robotics and Automation Letters (RA-L)*, **6** (2021), 3184. 63, 70, 71
- [77] LOWE, D. G. Distinctive Image Features from Scale-Invariant Keypoints. *Intl. Journal of Computer Vision (IJCV)*, **60** (2004), 91. 31
- [78] LOWE, D. G. Distinctive image features from scale-invariant keypoints. *Intl. Journal of Computer Vision (IJCV)*, **60** (2004), 91. 96

- [79] LOWRY, S., SÜNDEHAUF, N., NEWMAN, P., LEONARD, J. J., COX, D., CORKE, P., AND MILFORD, M. J. Visual place recognition: A survey. *IEEE Trans. on Robotics (TRO)*, **32** (2015), 1. 29, 32
- [80] LV, X., WANG, B., DOU, Z., YE, D., AND WANG, S. Lccnet: Lidar and camera self-calibration using cost volume network. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2894–2901 (2021). 52
- [81] MADDERN, W., PASCOE, G., LINEGAR, C., AND NEWMAN, P. 1 year, 1000 km: The oxford robotcar dataset. *Intl. Journal of Robotics Research (IJRR)*, **36** (2017), 3. 75, 76, 78
- [82] MAGNUSSON, M., ANDREASSON, H., NÜCHTER, A., AND LILIENTHAL, A. J. Automatic appearance-based loop detection from three-dimensional laser data using the normal distributions transform. *Journal of Field Robotics (JFR)*, **26** (2009), 892. 30, 41
- [83] MILFORD, M. Vision-based place recognition: how low can you go? *Intl. Journal of Robotics Research (IJRR)*, **32** (2013), 766. 31
- [84] MILFORD, M. AND WYETH, G. SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)* (2012). 31
- [85] MIRZAEI, F. M., KOTTAS, D. G., AND ROUMELIOTIS, S. I. 3d lidar–camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization. *Intl. Journal of Robotics Research (IJRR)*, **31** (2012), 452. Available from: <https://doi.org/10.1177/0278364911435689>, arXiv:<https://doi.org/10.1177/0278364911435689>, doi:10.1177/0278364911435689. 53
- [86] MUR-ARTAL, R. AND TARDÓS, J. D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans. on Robotics (TRO)*, **33** (2017), 1255. 41, 46, 48, 62, 68, 70, 101
- [87] NASEER, T., BURGARD, W., AND STACHNISS, C. Robust Visual Localization Across Seasons. *IEEE Trans. on Robotics (TRO)*, (2018). doi:10.1109/tro.2017.2788045. 29, 31
- [88] NEWCOMBE, R. A., FOX, D., AND SEITZ, S. M. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 343–352 (2015). 102
- [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. 89, 91
- [90] NÜCHTER, A., LINGEMANN, K., HERTZBERG, J., AND SURMANN, H. Heuristic-based laser scan matching for outdoor 6d slam. In *Annual Conf. on Artificial Intelligence*, pp. 304–319 (2005). 40



- [91] PANDEY, G., MCBRIDE, J. R., AND EUSTICE, R. M. Ford campus vision and lidar data set. *Intl. Journal of Robotics Research (IJRR)*, **30** (2011), 1543. 35, 36, 37
- [92] PANDEY, G., MCBRIDE, J. R., SAVARESE, S., AND EUSTICE, R. M. Automatic extrinsic calibration of vision and lidar by maximizing mutual information. **32** (2015), 696. 52
- [93] PARK, Y., YUN, S., WON, C. S., CHO, K., UM, K., AND SIM, S. Calibration between color camera and 3d lidar instruments with a polygonal planar board. *IEEE Sensors Journal*, **14** (2014), 5333. Available from: <https://www.mdpi.com/1424-8220/14/3/5333>. 53
- [94] 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. 53
- [95] PUSZTAI, Z. AND HAJDER, L. Accurate calibration of lidar-camera systems using ordinary boxes. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 394–402 (2017). doi:10.1109/ICCVW.2017.53. 53
- [96] QI, C. R., SU, H., MO, K., AND GUIBAS, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 652–660 (2017). 31
- [97] QI, C. R., YI, L., SU, H., AND GUIBAS, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pp. 5099–5108 (2017). 31
- [98] 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. xiii, xiv, xix, 30, 35, 36, 37, 38, 47, 48, 49, 75, 77, 78, 81
- [99] RASSHOFFER, H. R., SPIES, M., AND SPIES, H. Influences of weather phenomena on automotive laser radar systems. *Advances in Radio Science*, **9** (2011), 49. 13
- [100] RÖHLING, T., MACK, J., AND SCHULZ, D. A fast histogram-based similarity measure for detecting loop closures in 3-d lidar data. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 736–741 (2015). 31, 41
- [101] RUBLEE, E., RABAU, V., KONOLIGE, K., AND BRADSKI, G. Orb: An efficient alternative to sift or surf. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, pp. 2564–2571 (2011). xix, 31, 33, 37, 40, 41, 69
- [102] SALEM, O., GIACOMINI, E., BRIZI, L., GIAMMARINO, L. D., AND GRISETTI, G. Low frequency spinning lidar de-skewing. *arXiv preprint arXiv:2303.07312*, (2023). xvii, 20, 100, 101
- [103] SAUERBECK, F., OBERMEIER, B., RUDOLPH, M., AND BETZ, J. Rgb-l: Enhancing indirect visual slam using lidar-based dense depth maps. In *International Conference on Computer, Control and Robotics (ICCCR)*, pp. 95–100. IEEE (2023). 88



- [104] SCHERER, S., REHDER, J., ACHAR, S., COVER, H., CHAMBERS, A., NUSKE, S., AND SINGH, S. River mapping from a flying robot: state estimation, river detection, and obstacle mapping. *Autonomous Robots*, **33** (2012), 189–88
- [105] SCHLEGEL, D. AND GRISETTI, G. Hbst: A hamming distance embedding binary search tree for feature-based visual place recognition. *IEEE Robotics and Automation Letters (RA-L)*, **3** (2018), 3741. xiii, xix, 30, 31, 32, 34, 37, 40, 45
- [106] SCHÖNBERGER, J. L. AND FRAHM, J.-M. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2016). 94, 95
- [107] SCHÖNBERGER, J. L., ZHENG, E., POLLEFEYS, M., AND FRAHM, J.-M. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)* (2016). 94, 95
- [108] 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). xix, 39, 41, 47, 48, 62, 63, 68, 70, 75, 78
- [109] SEGAL, A., HAEHNEL, D., AND THRUN, S. Generalized-icp. In *Proc of Robotics: Science and Systems* (2009). 41
- [110] SHAN, T. AND ENGLLOT, 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 (2018). 41, 46, 49, 63, 70, 71
- [111] SINGANDHUPE, A., LA, H. M., AND HA, Q. P. Single frame lidar-camera calibration using registration of 3d planes. In *2022 Sixth IEEE International Conference on Robotic Computing (IRC)*, pp. 395–402 (2022). doi:10.1109/IRC55401.2022.00076. 53
- [112] SLAVCHEVA, M., KEHL, W., NAVAB, N., AND ILIC, S. Sdf-2-sdf: Highly accurate 3d object reconstruction. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pp. 680–696. Springer (2016). 62
- [113] SMITH, J. AND DOE, J. Fusion of lidar and rgb data for high-resolution 3d reconstruction. *Journal of Computer Vision*, **112** (2019), 345. 51
- [114] STEDER, B., GRISETTI, G., AND BURGARD, W. Robust place recognition for 3d range data based on point features. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 1400–1405. IEEE (2010). 30
- [115] STEDER, B., GRISETTI, G., LOOCK, M. V., AND BURGARD, W. Robust on-line model-based object detection from range images. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4739–4744. IEEE (2009). 30
- [116] STEDER, B., RUHNKE, M., GRZONKA, S., AND BURGARD, W. Place recognition in 3d scans using a combination of bag of words and point feature based relative pose estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 1249–1255. IEEE (2011). 30

- [117] STEDER, B., RUSU, R. B., KONOLIGE, K., AND BURGARD, W. Point feature extraction on 3d range scans taking into account object boundaries. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pp. 2601–2608. IEEE (2011). 30
- [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. *Intl. Journal of Robotics Research (IJRR)*, **31** (2012), 1377. 41, 63
- [119] STRÖM, D. P., BOGOSLAVSKIY, I., AND STACHNISS, C. Robust exploration and homing for autonomous robots. *Journal on Robotics and Autonomous Systems (RAS)*, (2016). 41
- [120] STUMM, E., MEI, C., LACROIX, S., AND CHLI, M. Location Graphs for Visual Place Recognition. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)* (2015). 31
- [121] STURM, J., ENGELHARD, N., ENDRES, F., BURGARD, W., AND CREMERS, D. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 573–580 (2012). xix, 47, 48
- [122] SUN, C., WEI, Z., HUANG, W., LIU, Q., AND WANG, B. Automatic targetless calibration for lidar and camera based on instance segmentation. *IEEE Robotics and Automation Letters (RA-L)*, **8** (2022), 981. 52
- [123] TRAHANIAS, P., BURGARD, W., ARGYROS, A., HÄHNEL, D., BALTZAKIS, H., PFAFF, P., AND STACHNISS, C. TOURBOT and WebFAIR: Web-operated mobile robots for tele-presence in populated exhibitions. *IEEE Robotics and Automation Magazine (RAM)*, **12** (2005), 77. 41
- [124] 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). 61
- [125] TÓTH, 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 (2020). [doi:10.1109/ICRA40945.2020.9197316](https://doi.org/10.1109/ICRA40945.2020.9197316). 53
- [126] UY, M. A. AND LEE, G. H. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 4470–4479 (2018). 31
- [127] VALGREN, C. AND LILIENTHAL, A. SIFT, SURF & Seasons: Appearance-Based Long-Term Localization in Outdoor Environments. *Journal on Robotics and Autonomous Systems (RAS)*, **85** (2010), 149. 31
- [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, 63

- [129] 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. *IEEE Robotics and Automation Letters (RA-L)*, **8** (2023), 1029–85
- [130] VYSOTSKA, O. AND STACHNISS, C. Lazy Data Association For Image Sequences Matching Under Substantial Appearance Changes. *IEEE Robotics and Automation Letters (RA-L)*, **1** (2016), 213. Available from: <http://www.ipb.uni-bonn.de/pdfs/vysotska16ral-icra.pdf>. 31
- [131] VYSOTSKA, O. AND STACHNISS, C. Effective Visual Place Recognition Using Multi-Sequence Maps. *IEEE Robotics and Automation Letters (RA-L)*, **4** (2019), 1730. Available from: <http://www.ipb.uni-bonn.de/pdfs/vysotska2019ral.pdf>. 29, 31
- [132] 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). 85
- [133] WHELAN, T., LEUTENEGGER, S., SALAS-MORENO, R., GLOCKER, B., AND DAVISON, A. Elasticfusion: Dense slam without a pose-graph. In *Proc. of Robotics: Science and Systems (RSS)* (2015). 46, 48, 68, 70
- [134] WILLIAMS, B., I., G. K., AND REID. Real-time slam relocalisation. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, pp. 1–8 (2007). 40
- [135] XU, Y., OU, Y., AND XU, T. Slam of robot based on the fusion of vision and lidar. In *Proc. of the International Conference on Cyborg and Bionic Systems (CBS)*, pp. 121–126. IEEE (2018). 88
- [136] YAN, Z., SUN, L., KRAJNIK, T., AND RUCHEK, 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). 75
- [137] 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 (2021). 52
- [138] ZAGANIDIS, A., ZERNTEV, A., DUCKETT, T., AND CIELNIAK, G. Semantically assisted loop closure in slam using ndt histograms. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4562–4568 (2019). 31
- [139] 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). 63
- [140] ZHANG, J. AND SINGH, S. Loam: Lidar odometry and mapping in real-time. In *Proc. of Robotics: Science and Systems (RSS)* (2014). 29, 41, 63, 88
- [141] 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). 29, 41, 63, 88

- [142] ZHANG, L., CAMURRI, M., AND FALLON, M. Multi-camera lidar inertial extension to the newer college dataset. *arXiv*, 2112.08854. (2021). [arXiv:2112.08854](https://arxiv.org/abs/2112.08854). xiv, xv, xix, 40, 48, 49, 69, 71, 74, 78
- [143] 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. *IEEE Robotics and Automation Letters (RA-L)*, **8** (2022), 408. 75, 77, 78, 81
- [144] 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 (2018). [doi: 10.1109/IROS.2018.8593660](https://doi.org/10.1109/IROS.2018.8593660). 53
- [145] ZOLLHÖFER, M., STOTKO, P., GÖRLITZ, A., THEOBALT, C., NIESSNER, M., KLEIN, R., AND KOLB, A. State of the art on 3d reconstruction with rgb-d cameras. In *Computer graphics forum*, vol. 37, pp. 625–652. Wiley Online Library (2018). 92