UNIVERSITY OF TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
DEPARTMENT OF INDUSTRIAL ENGINEERING
FONDAZIONE BRUNO KESSLER

Doctorate Program in Industrial Innovation

Robotic Contextual Awareness for Human-Robot Collaboration and Environmental Understanding

Federico Rollo

Accademic Advisor
Prof. Marco Roveri
Università di Trento

Industrial Advisor
Dr. Arash Ajoudani
Istituto Italiano di Tecnologia

December 2025

To my father,
who is always with me.

Disclaimer

I, Federico Rollo ("the Author"), wrote this doctoral thesis in my personal capacity while working at Leonardo Spa ("the Company"), holding the position of Technical Coordinator of the Perceptive Navigation research area of the Robotics Laboratory in the Innovation Hubs & Intellectual Property UO of the Corporate Division. The views and opinions expressed in this thesis are those of the Author only and do not necessarily reflect the official policy or position of the Company. This thesis has been written using open information sources only, *i.e.*, public domain information gathered from unclassified knowledge sources, being the majority of them available on the internet, and not subject to any disclosure restriction of any kind. This includes some of the yet-unpublished results of various research activities carried out in Leonardo Spa with the thesis author's contribution or under his supervision. All cited sources (texts, patents, data, and images) have been duly referenced, where possible and where required. All cited trademarks and patents have been indicated as the property of their respective owners, where possible and where required. Examples given within this thesis must be considered only as the Author's assumptions and are not reflective of the position of the Company, or of any third party whatsoever. The Author has made reasonable efforts to ensure that all information and material included in this thesis were accurate at the time of writing, but the Author cannot guarantee their accuracy. The Author has not received any remuneration, consideration, or benefit for mentioning, reporting, including, or commenting about any paper, program, product, policy, law, theory, hypothesis, or intellectual property. For any errors or inadequacies that may remain in this work, of course, the responsibility is entirely on my own.

Acknowledgment

I want to express my sincere gratitude to everyone who has supported and guided me throughout my PhD journey. Without them, this path would not have been possible.

My deepest appreciation goes to my former senior seatmate, Andrea Zunino, whose invaluable teaching and insightful reviews were instrumental in initiating my research path. I am profoundly thankful to my supervisors, Prof. Marco Roveri and Dr. Arash Ajoudani, for their exceptional guidance and mentorship. Special thanks are also extended to Navvab Kashiri, my current principal investigator, and all the senior researchers who generously shared their expertise and provided invaluable assistance over the years.

I am immensely grateful to my current and former Robotics Lab teammates at Leonardo Spa, with whom I worked closely during my PhD: Edoardo Del Bianco, Valentina Pericu, Luigi Raiano, Enrico Mingo Hoffman, Fabio Amadio, Gennaro Raiola, Andrea Testa, Saber Mohammadi, and Marco Laghi, among others. Thanks for sharing this PhD journey with me over these years. The shared ideas, experiences, and team spirit have been crucial to my growth, and I cherish the friendships and the vibrant collaborative environment we've built. Thanks also go to my latest Robotics teammates: Simone Giampà, Andrea Monguzzi, Giuseppe Alfonso, and Marco Puliti, with whom I'll continue the journey in robotics after my graduation.

I also need to thank the researchers from the Istituto Italiano di Tecnologia (IIT) for the joyful times spent together. Especially Nikos Tsagarakis, who, together with Arash and their laboratories, Humanoids and Human Centered Mechatronics (HHCM) and Human-Robot Interfaces and Interaction (HRI^2), followed part of my PhD, giving insightful and invaluable suggestions.

A huge thanks goes to Alessandro Garibbo, whose suggestions have been crucial for starting and concluding this path. I would like to thank all the Leonardo Spa staff whom I met during this time for their valuable support, with a special thanks to Alessandro Massa, Salvatore Scervo, and Pierpaolo Gambini for having guided the Leonardo Labs' evolution, and to CEO Roberto Cingolani and Co-General Manager Simone Ungaro, who are pushing and leading the innovation in our company.

Finally, I extend my heartfelt thanks to my family. To my mother, for her unwavering support from the very beginning. To my brother, for his wisdom and protection. To my nephews, who consistently fill my life with joy (and occasional delightful chaos). To my girlfriend, for her boundless love and steady support, especially during the most challenging times. And to my father, who I am certain would have been incredibly proud of this achievement and the path I've taken.

Abstract

The transition of autonomous mobile robots from controlled industrial settings to dynamic, human-centric environments, such as manufacturing, logistics, and healthcare, has made their safe and autonomous operation a critical area of research. These sophisticated machines must be capable of perceiving, understanding, and interacting with their surroundings to navigate freely and perform complex tasks. A significant obstacle to achieving this is the lack of comprehensive contextual awareness, which requires a robot to recognize its spatial environment and identify the objects and actors within it. Without this perceptual knowledge, robots struggle to plan adaptive behaviors or engage in meaningful interaction with humans.

This thesis presents novel solutions to this challenge by exploring two distinct but complementary research directions. The first direction involves human re-identification and tracking to improve HRC. Our developed approach enables a mobile robot to recognize a specific person, facilitating targeted collaboration while ignoring other individuals. The second direction focuses on enhancing the robot's overall perceptual capabilities to understand its environment geometrically and semantically. Geometric information is vital for motion planning and collision avoidance, while semantic knowledge provides the robot with a richer understanding for more advanced interaction. Both solutions are driven by the improvement of the semantical understanding of robots that enhance their knowledge of their surroundings, allowing a smoother and more natural interaction between robots, humans, and the environment. The contributions of this work in human re-identification and environmental understanding represent a significant step toward a future where robots are more contextually aware, enabling safer coexistence and more effective collaboration.

Keywords

Context Awareness, Environmental Understanding, Re-Identification, SLAM, Loop Closure Detection, Semantic Mapping

Contents

List of Tables

List of Figures

List of Acronyms

2D two-dimentional. 21, 41, 47, 63, 69, 74, 76, 77

3D three-dimensional. xiv, 1, 4, 14, 15, 20, 21, 27–29, 31, 33, 41, 47, 49, 61, 69, 71, 74, 76–79, 82, 85, 98

3DGS 3D Gaussian Splatting. 28, 31

AI Artificial Intelligence. 2, 3, 19, 98

AMCL Adaptive Monte Carlo Localization. 9

ATE Absolute Trajectory Error. xi, 99–102, 104, 119

AUC Area Under the Curve. 107, 108

BoW Bag-of-Words. 26

BT Behaviour Tree. 81, 82, 121

CARPE-ID Continuous Adaptation for personalized Re-identification. xiv, xvi, 6, 7, 17, 54–59, 89, 91, 93–97, 118, 119, 142

CL Continual Learning. 17–19

CLIP Contrastive Language-Image Pre-training. 30, 31

CNN Convolutional Neural Network. 14, 16, 26, 29, 37

CP Correspondence Problem. 33, 34

CRF Conditional Random Field. 29, 30

DA Data Association. 33, 34, 37

DDIOR Dynamic Distance–Intensity Outlier Removal. 25

DEMA Damped Exponential Moving Average. xiv, 53, 54, 118

DLO Direct LiDAR Odometry. 21, 62, 101–104

DoF Degrees of Freedom. 27

EMA Exponential Moving Average. xiv, 53, 54, 118

EWC Elastic Weight Consolidation. 18

FFM Feature Funnel Model. 16

FN False Negative. 107

FOV field-of-view. xiv, 15–17, 50, 51, 59, 80, 89, 92, 93, 95

FP False Positive. 107

FSM Finite State Machine. xiv, 47, 49–51

GICP Generalized Iterative Closest Point. 21, 62, 64

GNSS Global Navigation Satellite System. 30

GPU Graphic Processing Unit. 27, 56, 57, 85, 89, 94, 98, 111

GSC Gaussian Scan Context. xi, xv, xvii, 4, 8, 27, 68, 69, 72, 73, 106–110, 120

HOG Histograms of Oriented Gradients. 14

HRC Human-Robot Collaboration. vii, 3, 5, 17, 28, 43, 89, 117–120

HRI Human-Robot Interaction. 2, 5–7, 9, 10, 15–17, 89, 92, 118, 119

ICP Iterative Closest Point. 21, 23, 28, 61, 103, 104

ID Identifier. 6, 35, 52, 54, 92

IEEE Institute of Electrical and Electronics Engineers. xi, xiii–xvii, 44, 46, 48–51, 53, 55, 58, 60, 75, 78, 80, 86, 88, 90, 91, 94–96, 99, 100, 111–114, 142, 143

IMU Inertial Measurement Unit. xv, 20–22, 60, 61, 98, 101, 120

IR Infra Red. 5, 14

iSAM2 Incremental Smoothing and Mapping. 22, 64

ISC Intensity Scan Context. 23, 27, 109

KF Kalman Filter. 21, 47, 49, 50

L-REID Lifelong Re-Identification. 18

LCD Loop Closure Detection. xv, 7, 8, 10, 20–22, 26, 27, 36–38, 41, 59, 60, 63, 64, 68, 69, 72, 74, 98, 108, 110, 120

LED Light-Emitting Diode. 14

LiDAR Light Detection and Ranging. xi, xiii, xv, xvii, 1, 4, 7–10, 20–27, 29–31, 36, 40, 41, 59, 61–63, 65–67, 69–71, 74–78, 98, 99, 101–105, 109–115, 119–121

LIDSOR Low-Intensity Dynamic Statistical Outlier Removal. 25

LIOR Low-Intensity Outlier Removal. 25

LLM Large Language Model. 30

LOAM LiDAR Odometry and Mapping. 20, 22, 26, 27

LVI LiDAR-Visual-Inertial. 21

MAP Maximum A Posteriori. 40

MMT Mutual Mean Teaching. 5, 45

MOT Multi-Object Tracking. xiv, xvi, 6, 15, 17, 51, 52, 54, 89, 90, 92–95, 118

MTT Multi-Target Tracking. 33

NeRF Neural Radiance Field. 28

OE Orientation Error. 101, 102

ORB Oriented FAST and rotated BRIEF. 26, 40

OT Object Tracking. 15

P Precision. 107

PCL Point Cloud Library. 67

PN Progressive Networks. 18

PnP Perspective-n-Point. 29

PR Precision-Recall. 107, 108

R Recall. 107

Re-ID Re-identification. xi, xiii, xiv, xvi, 3–7, 10, 13–19, 34–38, 42–45, 47, 51–55, 59, 85–87, 89, 90, 92–95, 97, 98, 107, 109, 117–120

RGB Red, Green and Blue. xiii, xv, xvii, 9, 16, 21, 26, 29, 30, 44, 47, 48, 52, 74, 75, 112, 114, 115

RGB-D Red, Green, Blue and Depth. xv, xvii, 9, 14, 28–31, 41, 43, 76, 78, 82, 83, 110, 112, 114, 115, 117, 121

RMSE Root Mean Square Error. xi, 99, 101–103

ROS Robot Operating System. 5, 49, 80

RPE Relative Pose Error. xi, 99–102, 104, 119

RViz ROS Visualization Tool. xvi, 9, 80

SC Scan Context. 5, 23, 26, 27, 63, 69, 71

SC++ Scan Context++. xi, xv, xvii, 7, 8, 22, 27, 60, 63, 68–71, 106–110, 119, 120

SDF Signed Distance Function. 41

SfM Structure from Motion. 33

SI Synaptic Intelligence. 18

SIFT Scale-Invariant Feature Transform. 40

SLAM Simultaneous Localization and Mapping. xi, 4, 7, 8, 13, 14, 19–24, 26, 28–31, 33, 34, 36–43, 59–61, 66, 71, 98, 99, 101, 105, 108, 110, 119, 120

SOT Single-Object Tracking. 15, 17, 93

SoTA State-of-the-Art. xi, xvi, 5, 7–10, 13, 17, 20, 30, 43, 59, 68, 90, 92, 94, 95, 98–101, 103, 119, 120

SURF Speeded Up Robust Features. 28

SVM Support Vector Machine. xiv, 16, 48, 49, 86, 87

TE Translation Error. 101, 102

TN True Negative. 107

TP True Positive. 107

TSDF Truncated Signed Distance Function. 29

TVE Translation Vector Error. 101–103

UAV Unmanned Aerial Vehicle. 29

UI User Interface. 80, 81, 121

UKF Unscented Kalman Filter. 16

V-SLAM Visual SLAM. 20, 40

VBR Vision Benchmark in Rome. xi, xvii, 98–100, 106, 109, 120

VIO Visual-Inertial Odometry. 29

VLAD Vector of Locally Aggregated Descriptors. 26

VLM Vision-Language Model. 30, 31

YOLO You Only Look Once. 6, 45

List of Symbols and Notation

This list describes several symbols and the notation that will be later used within the body of the thesis.

$\mathbb{N}$	Set of natural numbers.
$\mathbb{R}$	Set of relative numbers.
x	Scalar.
$x_t \equiv x[t]$	Scalar at time t.
$\lvert x \rvert$	Modulus of scalar x, also known as the absolute value of x.
$\mathbf{x}$	Vector.
$\lvert \mathbf{x} \rvert$	Cardinality of vector $\mathbf{x}$, *i.e.*, number of elements in the vector. The same holds for matrices and sets.
$\mathbf{x}_t \equiv \mathbf{x}[t]$	Vector at time t.
X	Vectors set.
$\mathrm{X}_{1:t}$	Sequence of vectors from $\mathbf{x}_1$ to $\mathbf{x}_t$. *I.e.*, $\{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$.
$\mathbf{X}$	Matrix.
$\mathbf{X}_{i,j}$	Matrix entry on row i and column j. When i or j is replaced by :, it means the entire row or column.
$\mathbb{X}$	Matrices set.
$\mathbb{X}_{1:t}$	Sequence of Matrices from $\mathbf{X}_1$ to $\mathbf{X}_t$. *I.e.*, $\{\mathbf{X}_1, \ldots, \mathbf{X}_t\}$.
$\mathbf{I}_k$	Identity matrix of dimensions $k \times k$.
x^*	Optimal value of x. Valid for scalar, vectors, and matrices.
$\hat{x}$	Estimated value of x. Valid for scalar, vectors, and matrices.
$\in$	Contained in set/sequence. *E.g.*, $\mathbf{x} \in \mathrm{X}$, Vector $\mathbf{x}$ contained in Vector set X.

$\notin$	Not contained in set/sequence. *E.g.*, $\mathbf{x} \notin \mathrm{X}$, Vector $\mathbf{x}$ not contained in Vector set X.
$\equiv$	Defined as. *I.e.*, $x \equiv \ldots$, mean x is defined as or is equivalent to $\ldots$
&	Conditional And.
|	Conditional Or.
$log(x)$	Natural logarithm.
$e^x \equiv exp(x)$	Exponential.
$\mathrm{A} = \{x \| y\}$	Set A is equivalent to the set of all elements x, such that | the condition y is true (or holds).
$\mathbf{P}(\mathbf{a})$	Probability distribution of a.
$\mathbf{P}(\mathbf{a}\|\mathbf{b})$	Conditional probability distribution of a given b.
$\sum_{i=1}^{\mathrm{N}}$	Summation iterating from $i = 1$ to $\mathrm{N} \in \mathbb{N}$.
$\sum_{\mathbf{x} \in \mathrm{X}}$	Summation iterating over X set. The same is valid for matrices.
$\prod_{i=1}^{\mathrm{N}}$	Product iterating from $i = 1$ to $\mathrm{N} \in \mathbb{N}$.
$\prod_{\mathbf{x} \in \mathrm{X}}$	Product iterating over X set. The same is valid for matrices.
μ	Mean value of a Gaussian distribution. Valid for scalar and vector.
σ	standard deviation value of a Gaussian distribution. Valid for scalar and vector.
$\mathbf{\Sigma}$	Covariance matrix of a Gaussian distribution.
$\mathrm{G} \equiv \mathcal{N}(\mu, \sigma)$	Univariate Gaussian distribution.
$\boldsymbol{\mathcal{G}} \equiv \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$	Multivariate Gaussian distribution.
$\mathbf{p} \in \mathbb{R}^3$	Three-dimensional point in the Cartesian space.
$\mathbf{p} \in \mathbb{R}^4$	Three-dimensional point in the Cartesian space plus intensity value.
$p^x,\ p^y,\ p^z,\ p^\psi$	x, y, z and intensity values of point $\mathbf{p}$.
${}^a\mathbf{p} \ \equiv \ {}^a\mathbf{t}$	Point $\mathbf{p}$ or vector $\mathbf{t}$ expressed in frame a.
${}^a\mathbf{t}_b \equiv \{{}^a\mathbf{x}_b,\ {}^a\mathbf{y}_b,\ {}^a\mathbf{z}_b\}$	Translation vector $(x,\ y,\ z)$ from frame b to frame a.

${}^{a}\mathbf{R}_{b}$	Rotation matrix from frame b to frame a.
${}^{a}\mathbf{H}_{b}$	Homogeneous transformation from frame b to frame a.
${}^{a}\mathbf{T}_{b}$	Homogeneous transformation from frame b to frame a containing only translational vector. *I.e.*, ${}^{a}\mathbf{R}_{b} = \mathbf{I}_{3}$.
$\mathcal{L}$	Loss function.
$s \equiv \mathcal{P}$	Point cloud scan.
$\mathcal{S} \equiv \sum \mathcal{P}$	Submap defined as sum of point clouds.
$\mathbb{S} \equiv \sum \mathcal{S}$	Submap sum.
$\lVert \cdot \rVert_0$	L0-Norm.
$\lVert \cdot \rVert_2$	L2-Norm also known as Euclidean norm.
$\lVert \cdot \rVert_F$	Frobenius norm.
$\langle \mathbf{a}, \mathbf{b} \rangle$	Dot product between vectors $\mathbf{a}$ and $\mathbf{b}$.
$\lceil a \rceil$	Ceiling function of a scalar a. Uprounds the scalar a up to the nearest integer.

Chapter 1

Introduction

In recent years, the primary focus of robotics has shifted toward enhancing the autonomy and motion capabilities of robots to operate within our complex, human-centric environments. This evolution is driven by the growing demand for mechatronic systems that assist with repetitive, dangerous, or physically demanding tasks. While the goal of a fully autonomous robot capable of adapting to any situation is still a significant research challenge, substantial progress has been made. A core requirement for achieving this level of autonomy is the robot's ability to effectively perceive, understand, and interact with its surroundings. Without a comprehensive grasp of its environment, a robot cannot act responsively or perform meaningful, real-world tasks.

For decades, the majority of industrial robots, particularly in sectors like the automotive and production industry, have operated as *blind* machines. These robots are programmed to repeat the same high-accuracy motions within a highly structured environment. This approach, rooted in the Ford concept of the assembly line, works efficiently as long as the environment remains static. However, as soon as a variable changes, these robots become unable to complete their tasks without manual recalibration, leading to costly downtime. Modern solutions have begun to address this limitation by incorporating sensing capabilities, such as cameras and time-of-flight sensors, to adapt a robot's final pose to the object it is manipulating. This allows for smoother task execution and increases the robot's awareness of its immediate surroundings.

This early integration of sensors marked a crucial step forward, but it largely focused on a purely geometric understanding of the world. Advanced sensing modalities, such as LiDAR and camera sensors, provide rich 3D data that enables robots to construct detailed maps for path planning and obstacle avoidance. While these geometric representations are excellent for describing the location and size of objects, they offer

little insight into their semantic meaning, *e.g.*, what they represent or why they are significant. This limitation is a significant barrier to operating in dynamic, unstructured environments.

1.1 Problem statement

Humans and many animals possess a far more profound perceptual capability. We not only perceive the geometric layout of our environment but also assign high-level semantic meaning to objects, places, and other entities. For instance, while geometric cues are essential for reactive behaviors, like catching a falling object, semantic understanding is indispensable for all high-level reasoning and decision-making. We instinctively know to grasp a knife by its handle, not its blade, and we avoid walking through mud, especially with new shoes. This ability to fuse geometric and semantic information is what enables us to make nuanced decisions and exhibit highly adaptive behaviors.

The progression from a purely geometric understanding to a more semantically informed perception is paramount for mobile robots to operate intelligently in complex, human-inhabited spaces. This interdisciplinary field, which merges Artificial Intelligence (AI) with spatial and geometrical concepts, is now known as Spatial AI. While some tasks still benefit from standard computer vision and geometric algorithms, more complex problems, such as object detection, require data-driven solutions powered by neural networks.

This thesis revolves around the concept of robotic contextual awareness, which is the ability of a robot to perceive and comprehend its surroundings. The concept is wide and covers many aspects. The problems faced in this thesis are deeply discussed in Chapter 3 where the data association and the mapping problems are presented respectively in Section 3.1 and Section 3.2.

1.2 Presented solutions

This thesis argues that to truly achieve advanced robotic autonomy and intuitive Human-Robot Interaction (HRI), robots must develop a comprehensive contextual awareness. This means moving beyond simply knowing where things are to understanding what they are, who is present, and how these elements relate to each other within a dynamic scene. Only by integrating this deeper understanding can robots become truly robust, adaptable, and trustworthy collaborators.

The research is moving toward developing robots that can understand their context and act as an active part of the environment, rather than an outsider object. This includes enhanced capabilities for environmental understanding (Spatial AI) and improved motion capabilities that can adapt to the environment. Current research on adaptive motion capabilities, often leveraging reinforcement learning and foundation models, shows great promise for specific situations due to their robustness. However, their generalization capabilities remain poor, as even small environmental changes can cause these controllers to fail because they have not encountered that exact situation during training. By combining deep contextual understanding with advanced motion capabilities, we can develop robots that are not just blind manipulators, but active, intelligent actors in our evolving world.

This PhD thesis addresses the critical challenge of achieving comprehensive contextual awareness for mobile robots operating in complex, semi-structured, and unstructured environments. For robots to become truly intelligent and autonomous, they must move beyond a purely geometric understanding of their surroundings and acquire a holistic situational awareness. This includes not only an understanding of the static environment but also the dynamic entities within it, particularly humans, and the intricate relationships between them.

A deep, contextual understanding is the cornerstone of effective robotic autonomy in human-centric spaces. Without it, a robot is limited to reactive, often brittle behaviors. For a robot to engage in socially compliant navigation, intuitive Human-Robot Collaboration (HRC), and truly adaptive task execution, it must be able to understand what objects are, who is present, and how all these elements relate to each other. This thesis posits that true robotic context awareness emerges from the synergistic integration of two fundamental domains: environmental perception and dynamic entity understanding.

To achieve this comprehensive contextual awareness, this research is structured around two interconnected pillars: reliable object detection and Re-ID and robust environmental mapping and semantic understanding.

The first pillar of this research delves into the persistent perception of dynamic entities, primarily focusing on humans. Through the work on Person Re-ID, including contributions like FollowMe [111], continuous adaptation [112], and personalized Re-ID through unsupervised continual learning [110], the crucial need for robots to continuously track and recognize individuals across different viewpoints and over time is addressed. This capability is essential for enabling personalized interactions and human-aware navigation, allowing a robot to maintain a consistent understanding of a person's

identity despite occlusions or changes in position.

The second pillar of this work focuses on endowing robots with the ability to build accurate and semantically rich representations of their operational space. This is explored through the work in SLAM and Semantic SLAM. This research, including contributions such as LEO-SLAM [107], Artifacts Mapping [109], Ground-Aware LiDAR Filtering, and GSC, enables a robot to not only localize itself and map its environment precisely but also to identify and localize salient objects and features. This capability is particularly critical in challenging scenarios, such as environments with significant reflections, where traditional methods often fail.

A central argument of this thesis is that neither of these pillars is sufficient on its own. While advanced Semantic SLAM provides the foundational environmental context, the Re-ID research enriches this understanding with dynamic human presence and identity. Conversely, the robust environmental understanding gained through the advanced mapping techniques offers vital contextual cues that enhance the reliability and adaptability of person Re-ID in real-world settings.

The combined approach forms the foundation for developing more sophisticated and trustworthy autonomous systems. By integrating environmental mapping with dynamic object Re-ID, the robots can move beyond simple, reactive behaviors to exhibit proactive, socially compliant, and intelligent actions. This research thus paves the way for a new generation of robots that are not just machines in our world, but active and understanding participants in human-centric environments.

1.3 Innovative aspects

In the following, the contributions are quickly introduced.

This thesis presents several key contributions in the robotic context awareness realm; the most significant are:

- A robust Re-ID method (see Section 4.1) that accurately recognizes specific people or objects despite partial or total occlusions over extended periods. It enhances robustness to appearance changes through a statistical continual adaptation and a continual learning technique. This technique employs a parallel twin neural network for online training, which allows the Re-ID application to operate without interruption.

- A novel 3D LiDAR SLAM approach (see Section 4.2.1) that uses a submap-based keyframe strategy. This approach significantly enhances odometry and

loop closure for more accurate and reliable mapping.

- A statistical version of a SoTA loop closure algorithm known as Scan Context (SC), where, by employing Gaussian probabilities, place recognition capabilities are improved (see Section 4.2.3).
- The introduction of multi-modal semantic mapping to substantially improve a robot's environmental understanding capabilities (see Section 4.3).

The following sections provide a brief overview of these contributions.

1.3.1 Visual re-identification for human-robot collaboration

The first contribution on context awareness is human Re-ID for HRI. The first introduced approach is FollowMe [111], a robust framework for person-following based on visual human Re-ID and hand gesture detection. The proposed architecture consists of three integrated modules: Perception, Decision Making, and Navigation. These modules receive sensor data and issue command references to the robot.

FollowMe is adaptable to various floating-base systems because it is built upon established robotics tools like Robot Operating System (ROS) [100], and SoTA machine learning algorithms such as Yolact++ [12], Mutual Mean Teaching (MMT) pretraining [36], and Mediapipe [157]. A key feature is that the user can move the robot without manual teleoperation. This approach uses visual Re-ID, which is light, robust, and personalized, meaning the target person doesn't need to wear special patterns or intrusive devices like beacon or Infra Red (IR) emitters. A hand gesture recognition module is also introduced, allowing the user to send intuitive commands, such as "stop", to the robot.

This work highlights that human detection and Re-ID using visual data are powerful tools for personalized HRC. The system is user-friendly because the robot can autonomously collect and use calibration data to re-identify and follow the target person. While much research focuses on isolated topics like multi-object tracking, this work addresses the more complex, real-world challenge of integrating mobile robotics into a practical application environment.

The potential applications for this framework are broad, including carrying heavy items, assisting with manipulation tasks at different stations [61], and providing support for people with care needs [32]. The Re-ID and hand gesture detection modules are qualitatively evaluated using dedicated experimental setups, demonstrating impressive accuracy. Additionally, the entire FollowMe framework is qualitatively evaluated in a

simulated working area, where the robot successfully follows the target, responds to hand gestures, and navigates around static and dynamic obstacles.

To further minimize human intervention in robot recognition and Re-ID, a key limitation of the original FollowMe [111] framework is addressed: its inability to track a person who changes their appearance, such as by changing their outfit, after the initial calibration. To overcome this, a novel Re-ID module that employs a deep learning approach based on feature extraction and continual adaptation is proposed. As the robot tracks the person, it continuously acquires new images, using this new appearance information to update an ideal target representation model. This allows the system to re-identify the person even if tracking is momentarily lost.

This new module, called CARPE-ID, integrates a MOT algorithm with an adaptive Re-ID layer. The MOT algorithm, known as *yolo_tracking*, is built upon the popular You Only Look Once (YOLO) framework [105] and the StrongSORT tracker [30]. The additional Re-ID layer adapts to changes in the target's appearance, effectively managing the "Identifier (ID) jumps" that can occur due to occlusions or appearance differences.

In this work, key contributions are:

- The development of CARPE-ID, a framework that achieves personalized HRI tasks with a specific target through continual adaptation.
- A quantitative evaluation of the system through real-world tests in a Human-Robot collaborative scenario.
- An in-depth analysis of the system's limitations and a discussion of potential solutions.

While CARPE-ID was a significant step toward a comprehensive tracking framework, it suffered from a major limitation: catastrophic forgetting. This means it could not re-identify previous appearances of the target person over time, but just for a confined amount of it.

An enhanced version of the CARPE-ID framework to solve the catastrophic forgetting issue [110] is proposed. This approach uses the output from CARPE-ID to train a parallel twin feature extractor network in an unsupervised continual learning manner. This method allows the network to autonomously embed and learn the appearances of the specific target, creating a highly personalized and robust Re-ID system. A key innovation of this final version is the development of a smart image pool acquisition method specifically designed to overcome the catastrophic forgetting problem.

The final contributions of this Re-ID work are:

- A novel unsupervised continual learning approach that trains a parallel feature extractor network online to create a personalized Re-ID framework. This unique combination has not been utilized in other Re-ID or tracking approaches in the literature.
- The development of an autonomous smart image pool acquisition method to directly address the issue of catastrophic forgetting.

In its final iteration, the Re-ID framework addresses the major challenges of personalized person tracking. While the original FollowMe framework was robust for simple HRI tasks, the introduction of CARPE-ID, along with its subsequent enhancements through continual learning and smart image pooling, has addressed its limitations. This demonstrates a robust Re-ID behavior and a superior level of human-robot integration, which are essential for a complete robotic context awareness.

1.3.2 Geometrical robotic mapping and place recognition

Geometrical mapping approaches are predominantly based on SLAM techniques. This is essential because perfect localization is rarely known in most environments, and a robot must localize itself to accurately create a map.

For this reason, LEO-SLAM [107], a LiDAR-based SLAM algorithm designed to generate an environmental map that accurately represents real-world surroundings and supports navigation and exploration tasks, is developed. This approach is built upon recent theoretical and algorithmic advancements. Using scan-matching algorithms, odometry is refined using LiDAR scans, and SC++ [54] is used for robust LCD. The key contributions lie in the unique alignment and loop closure strategies. Specifically:

- A multi-level alignment strategy that not only compares the current scan with the previous one (scan-to-scan, s2s) and with the last N_s submaps (scan-to-submaps, s2$\mathbb{S}$), similar to SoTA methods but, additionally introduce a submap-to-submaps alignment step ($\mathcal{S}$2$\mathbb{S}$), which is performed once a submap keyframe is finalized.
- A submap-based SC++ LCD method that uses an adaptive search area. This provides richer features for comparison compared to analyzing a single, sparse scan.

This efficient submap representation produces denser and more distinctive point clouds, which facilitates robust scan matching and significantly improves LCD.

During real-world robot experiments, the generated map was not completely accurate because LiDAR sensors are susceptible to reflections from transparent or reflective surfaces. These reflected points often have low intensity and can be filtered out by setting a simple threshold. However, this method can mistakenly remove ground points, as a low incidence angle between the LiDAR ray and a reflective ground surface can also result in a low-intensity return.

To address these issues, an efficient ground-aware intensity filter is introduced into LEO-SLAM. This filter enhances LiDAR-based systems by preserving crucial ground information while effectively filtering out erroneous reflections. This solution specifically tackles two key challenges: consistent erroneous reflections and the loss of valid low-intensity ground points.

The main contributions for this part of the work are:

- A simple yet effective intensity-based filter that accounts for the robot's height to retain low-reflectance ground points.
- A comprehensive evaluation and comparison of the filter's effectiveness using SoTA odometry systems. This solution is tested on a quadruped robot using a custom indoor dataset that includes numerous glass doors, demonstrating its robust performance in challenging real-world environments.

Another challenge was the accuracy of the place recognition algorithm. The conventional approach of setting a loop closure threshold often leads to a trade-off between low accuracy and an increased risk of false positives.

To address this, an extension of SC++ [54] called GSC is proposed. As its name implies, the method integrates a statistical analysis of the input point cloud to generate a more robust environmental representation within the context matrix. This statistical formulation allows the algorithm to capture the overall point distribution, thereby enhancing its resilience to the noise and outliers that are common in real-world scenarios.

This work represents a significant step forward in enhancing the robustness of SLAM algorithms for real-world robotic applications. The key contributions are:

- A more robust and computationally efficient LCD method achieved by integrating a statistical analysis of the point cloud.
- An in-depth study and comparative evaluation of multiple statistical distance metrics, including a discussion of the factors that influence their relative performance.

1.3.3 Introducing semantics for higher-level scene understanding and interaction

As previously discussed, while geometrical mapping is crucial for robotic navigation and exploration, performing higher-level autonomous tasks requires semantic understanding. Many SoTA robotic approaches, reported in Section 2.3, are based on a single camera, which limits mapping capabilities to small indoor environments. Conversely, solutions in autonomous driving often use both LiDAR and RGB cameras but neglect camera depth estimation, which is not essential in their context.

To bridge the strengths of both robotics and autonomous driving, a modular, multi-modal (RGB-D camera-LiDAR) online semantic mapping framework is proposed. This architecture can fuse sensor information in real-time, adapting its approach based on object distance and sensor accuracy. Semantic information from images is used to enrich the filtered and stabilized positions of objects, leading to precise object localization. The objects' dimensions are simplified as spherical shapes. This framework relies on external geometric navigation systems, such as SLAM or other localization algorithms like Adaptive Monte Carlo Localization (AMCL) [149].

The proposed application demonstrates good accuracy for both near and distant objects, thanks to this novel camera-LiDAR depth fusion technique. To the best of the author's knowledge, this specific fusion approach has not been explored in other robotic or autonomous driving semantic mapping works. This application also operates online on low-resource embedded systems, further highlighting its practical contributions. A custom RViz[1] plugin is introduced to improve the user experience for visualizing and interacting with the semantic map.

To demonstrate the utility of these semantic maps, an easy-to-use framework for a mobile robotic collaborator to complete a "bring me" service is developed. This work leverages the knowledge from the semantic mapping framework to accomplish the task, representing a step towards a more comprehensive and adaptable application that can dynamically respond to workers' needs.

1.4 Thesis structure

The thesis will follow a structured approach that moves from a more human-centric view to an environmental one. Specifically, the context awareness of the mobile robot will be first considered in a HRI scenario, to a robot-environment interaction, concluding

[1]rviz: `http://wiki.ros.org/rviz`

with a conceptual union between the two that brings the robot to a semantic environmental context-awareness and understanding. This is done because those two topics complement each other to ensure human-robot coexistence and robot autonomy. For these reasons, the following sections are ordered from human Re-ID and robot interaction to robot environmental understanding and mapping. Specifically, in Chapter 2 the SoTA is analyzed and compared. The human Re-ID and tracking techniques for HRI are analyzed in Section 2.1, while the robotic environmental mapping is analyzed in Section 2.2, concluding with an in-depth analysis of the semantic SLAM world in Section 2.3.

Chapter 3 will highlight the problems present in those fields and will provide hints on how they can be faced and handled. In Section 3.1, the correspondence problem is analyzed. As can be seen in the next sessions, this problem arises many times in robotics, as, also for humans, data association is one of the primary tasks performed to maintain consistency between the world perceived with our senses and the cognition and representation of those perceptions. Section 3.2 instead will focus on the environmental mapping problem. The problem is analyzed from both geometrical and semantic perspectives, and their dependencies and differences are highlighted. The mapping problem is also dependent on the previous one, as data association and correspondences are a crucial part of the mapping process. The methodologies and algorithms developed to solve these problems are presented in Chapter 4. Here, the theoretical information about the proposed solutions is described along with an explanation of the ideas behind them. Specifically, in Section 4.1, the Re-ID application for HRI is handled along with some subsequent improvements for robustness and accuracy. Section 4.2 contains the geometrical contribution proposed to improve the 3D SLAM problem in terms of new approaches to the LiDAR odometry, place recognition, and LCD. Finally, in Section 4.3, the semantic-aided mapping methods are presented. These methods make use of the expertise acquired with the previous two modules and put them together for an increased contextual awareness of the environment. The presented methods are then thoroughly analyzed and proved in Chapter 5 where each component is validated using different techniques and using benchmarks and real robot experiments. In Section 5.1, the Re-ID capabilities of the robot are evaluated, and a specific HRI task has been considered to prove the usability of those capabilities in a real robotic scenario. Section 5.2 demonstrates the mapping capabilities in terms of accuracy and performance of the proposed SLAM and loop closure methods, and Section 5.3 presents the results of the semantic contextual awareness and an example of how it can be used for improving robotic capabilities. As a conclusion, Chapter 6 reports the conclusive remarks and

the take-home messages of this document, along with some of the wrong choices made during this journey and the limitations that need to be handled for future works.

Chapter 2

Related works

Robotic systems operating in close proximity to humans demand a high degree of contextual awareness to ensure both safety and operational efficiency. This critical capability is built upon the robot's fundamental perceptive skills: reliably identifying collaborators, accurately mapping the workspace geometry, and semantically understanding the objects and areas within it. This chapter provides a comprehensive review of the current and historical SoTA research that directly addresses the core perceptive challenges investigated in this thesis.

The discussion begins with the Re-ID in Section 2.1, focusing on the essential task of maintaining a persistent identity for collaborators across various viewpoints and occlusions. Successively, the broader challenge of environmental perception is addressed, divided into two key areas: geometrical perception as reviewed through SLAM and related works in Section 2.2, and semantic understanding, the assignment of meaning to raw sensor data, in Section 2.3. Collectively, the works presented herein have significantly contributed to enhancing the robot's perceptive capabilities, forming the necessary foundation for the novel contributions presented in this thesis.

2.1 Visual re-identification for human-robot collaboration

Visual tracking and Re-ID are critical capabilities for robots intended to operate alongside humans and dynamic obstacles in unstructured environments. They are essential not only for safe coexistence but also for recognizing specific individuals with whom the robot must collaborate in settings like manufacturing. A primary requirement for mobile collaborative robots is the ability to accompany a person between different work-

stations to execute specific tasks. While conventional methods involve human-operated controls (*e.g.*, joysticks), a more autonomous and user-friendly solution involves a robot with visual recognition executing a person-following action without continuous human intervention.

2.1.1 Person following

Existing applications in both industry (*e.g.*, Piaggio's Gita and Kilo transport robots[1]) and research employ a variety of sensors to achieve person-following. Many of these traditional methods rely on extrinsic or intrusive localization devices. For instance, some use dedicated signal emitters, such as Wi-Fi transmitters [37], IR Light-Emitting Diodes (LEDs) [27, 2], or Bluetooth connections [97]. Other approaches utilize simple range-based sensors, such as sonar rings and rangefinders [91]. A drawback of these frameworks is the dependency on the target wearing or carrying specific, easily recognizable devices. Furthermore, many of these systems lack integration with robust obstacle avoidance algorithms, although [2] does use an ultrasonic sensor for this purpose.

More robust detection has been achieved using vision-based sensors, particularly RGB-D cameras. Object detection is often used for many tasks that span from object grasping [28] to semantic mapping [109]. Early work involved detecting specific objects, such as a red T-shirt, and using a range sensor for distance estimation and obstacle avoidance [95]. Other approaches use anthropometric features, such as shoulder length and head height, for distinction and Re-ID [126]. Fusion techniques are also common, such as combining camera-based person detection with laser range finders [129]. Feature extraction has evolved from using basic color and contour information [119] to more sophisticated holistic features like Histograms of Oriented Gradients (HOG) [26].

More recently, sophisticated tracking and detection methods have been employed. Eisenbach et al. [32] presented a care robot that detects legs via a laser range finder and the upper body using an orientation-based decision tree of HOGs, improving the tracking with color and clothing texture matching. Other solutions have fused SLAM with vision, as seen in Weber et al. [144], which uses LSD-3D and a Convolutional Neural Network (CNN) for head detection. The emergence of affordable aerial robotics has also seen the development of person-following drones that use holistic information (skeleton points) and simple gestures for control, as in Naseer et al. [83]. Collaborative robotics has leveraged tools like OpenPose [16] to track a person's skeleton in 3D using stereo cameras for tasks like pick and place, often integrating haptic feedback [61].

[1]Piaggio Gita and Kilo: `Gita-and-Kilo`

Crucially, most of these works lack the necessary robustness and personalization for long-term collaboration, as they primarily rely on transient or broadly shared features (*e.g.*, location, pose, simple colors) that are hardly distinguishable across individuals or robust to changes.

2.1.2 Visual object tracking

To overcome the limitations of the works presented in the previous Section 2.1.1, one of the approaches that can be used to enhance collaborative tasks is Object Tracking (OT). The field of visual tracking is broadly divided into Single-Object Tracking (SOT) and MOT. SOT algorithms track one selected object based on an initial bounding box, regardless of class, while MOT algorithms detect and track multiple objects simultaneously.

Recent surveys classify SOT techniques based on feature analysis, segmentation, estimation, and learning [128]. Specifically, correlation and deep learning algorithms have received detailed attention [161, 50], particularly those employing Siamese networks and discriminative correlation filters. Conversely, other reviews focus on the recent developments in MOT algorithms [74].

While many robust MOT systems exist, a performance trade-off often emerges. For example, TMOT [130] performs well in cluttered and occluded environments but with unsatisfactory time performance for real-time robotic applications. Faster MOT algorithms, such as those based on the LMOT framework [81], solve the timing issue, but these trackers typically cannot re-identify objects that leave and then re-enter the camera FOV. This limitation represents a key area of improvement and one of the strengths of the proposed works [111, 112, 110].

2.1.3 Other proposals

As established, general SOT and MOT algorithms often fail to meet the demands of personalized HRI due to their lack of robustness in re-identifying targets after partial or total occlusions. Consequently, the literature on robotics applications offers more specialized Re-ID methods.

The use of soft biometrics, such as skeleton points and face features, is widely employed for person tracking and Re-ID in robotic assistance. For instance, Patruno et al. [89] proposed creating a person descriptor using soft biometric features extracted from depth and color information. They utilized AlphaPose [33] to extract 3D skeleton points, normalizing the target person to a standard posture. A grid applied to this

normalized posture, with the mean color of each cell, formed the Re-ID descriptor. However, this method is limited by its reliance on clothing color, making it prone to errors when people wear similar outfits. Similarly, Ye et al. [154] developed a person-following system that uses a predefined model and skeleton heuristics to achieve robustness against partial occlusion. Crucially, this system assumes the person always remains within the camera's FOV, which is not guaranteed in dynamic real-world environments.

Face recognition provides another specialized approach. Liu et al. [70] proposed a method that trains a metric model offline and then uses online face information to match the target and update the model. They introduced the Feature Funnel Model (FFM) to merge appearance and skeleton information. Likewise, Wang et al. [143] improved HRI by using an unsupervised face Re-ID approach with a pre-trained CNN feature extractor. A major practical limitation of both face-based methods is the assumption that the human collaborator consistently faces the robot, an assumption often invalidated in real-world scenarios.

Other explored works use different sensor modalities and learning strategies. Koide et al. [59] employed a RGB monocular approach for tracking, extracting the skeleton using OpenPose [17], and tracking the target with an Unscented Kalman Filter (UKF). For Re-ID, they combined convolutional channel features with boosting techniques. However, their method requires an initial deep feature appearance calibration, which is used statically for subsequent Re-ID, lacking dynamic adaptation.

In a different direction, Cocsar et al. [22] utilized a thermal camera for tracking and Re-ID. They trained a network using a custom dataset. They sampled data through an entropy process to create discriminative descriptors. These descriptors were then used to train a SVM classifier. Unfortunately, this approach necessitates extensive training for thermal integration and, critically, cannot adjust to changes in the target's appearance, making it unsuitable for long-term personalized HRI.

The examination above reveals that existing methods suffer from significant limitations, particularly the inability to robustly handle target appearance shifts (*e.g.*, clothing changes) and a reliance on intrusive assumptions (*e.g.*, constant visibility, frontal face-to-robot orientation). These restrictions do not satisfy the established specifications for a reliable, personalized tracking system.

This thesis addresses these gaps by proposing a solution that tackles the problem of person Re-ID robustly, even in the presence of target appearance shifts and occlusions. A system that generalizes the person-following task using deep neural network features for Re-ID and visual data, similar to how a human would perceive, with additional depth data for localization, is presented. The key contribution lies in the development

and seamless integration of the entire visual pipeline: visual person detection, Re-ID, localization modules, and hand gesture detection for commanding. This comprehensive pipeline, or its individual components, can be readily adapted for other HRI tasks. The specific methods used for this integrated approach are presented in detail in Section 4.1.

2.1.4 Continual learning

To further enhance the performance of the personalized tracking system, an unsupervised, online training technique that employs Continual Learning (CL) and real-time dataset creation to improve Re-ID is developed [110]. Given the novelty of concurrently addressing both advanced Re-ID and CL within a robotic framework, this section first concludes by summarizing the related Re-ID literature presented above before detailing the SoTA in CL.

As highlighted in the previous section and in the prior work CARPE-ID [112], most SOT and MOT algorithms do not meet the strict requirements of personalized robotics, particularly their limitations in re-identifying targets after prolonged occlusions. Alternative methods that utilize skeleton point extraction for color-based Re-ID [89] or structure-matching [154] have been explored. However, these techniques are susceptible to errors when clothing is similar and often rely on the restrictive assumption that the person remains within the camera's FOV. Similarly, face-based recognition [70, 143] is compromised by the impractical assumption that the target consistently faces the robot, an issue confirmed in the experiments.

In the FollowMe framework [111], a learning-based Re-ID approach that used a calibration phase to adapt to the target's initial appearance is presented. While this module significantly improved HRC, it was inherently sensitive to subsequent target appearance changes.

These accumulated limitations across the SoTA make most existing approaches unsuitable for robust, personalized human tracking. The approach here presented builds directly upon the preliminary work, the CARPE-ID framework [112], which successfully addressed challenges related to occlusion and appearance changes. However, as noted in the Introduction, CARPE-ID is not immune to catastrophic forgetting, meaning it struggles to re-identify a person reverting to an older appearance. This latest work specifically proposes a novel solution, the use of CL, to this critical challenge, which is not sufficiently studied in the HRI tracking field.

Continual Learning, also known as Lifelong Learning, is a critical paradigm focused on training neural networks to adapt to new incoming data streams sequentially without

suffering from catastrophic forgetting, the abrupt and severe loss of previously acquired knowledge. This field attempts to solve the fundamental stability-plasticity dilemma, ensuring the model retains old knowledge (stability) while being flexible enough to integrate new information (plasticity) [87, 140].

CL methodologies can be broadly categorized into three main approaches:

- *Rehearsal/Memory-Based Methods*: These techniques maintain a limited memory buffer of representative data samples (exemplars) from previous tasks. By interleaving or "rehearsing" these old samples alongside the new data, the model is constantly reminded of its past knowledge. An early example of this concept was presented by Robins et al. [106]. More recent exemplar-based approaches, such as iCARL [104] and GDumb [96], focus on sophisticated selection and management of this memory buffer to maximize information retention. The work proposed in this thesis draws inspiration from this concept to develop the smart image pool acquisition method, reinforcing the online unsupervised continual learning in the framework.

- *Regularization-Based Methods*: This category mitigates forgetting by imposing constraints on the network's parameters during the learning of a new task. The core idea is to identify the weights that are critical to the performance of previous tasks and penalize any drastic changes to them. Landmark techniques in this area include Elastic Weight Consolidation (EWC) [57] and Synaptic Intelligence (SI) [156], which compute importance based on network sensitivity to weight changes. While computationally efficient due to low memory overhead, these methods can suffer from a poor ability to generalize to highly dissimilar new tasks, a critical trade-off addressed in recent theoretical analyses [63].

- *Dynamic Architectural Methods*: These approaches address the stability-plasticity dilemma by either expanding or dynamically modifying the network architecture as new tasks arrive. Methods like Progressive Networks (PN) dynamically allocate new resources (*e.g.*, adding new network branches) for each task while keeping parameters for old tasks frozen [116]. Other methods utilize task-specific and task-shared parameters or dynamically expand the network only when capacity is insufficient.

In the context of Person Re-ID, this concept is often termed Lifelong Re-Identification (L-REID), which is particularly challenging as it is an open-set, fine-grained problem where new identities constantly appear across various camera domains. Recent L-REID

studies have tackled issues like re-indexing free learning, where the model must remain compatible with features calculated by older models [24]. Others have proposed systems like Adaptive Knowledge Accumulation [98] to continually accumulate knowledge and improve generalization across unseen domains, often drawing inspiration from human cognitive models. Furthermore, work on Continual Meta Metric Learning aims to incrementally learn a feature space that generalizes well to new tasks [3].

In contrast, earlier works focusing on Re-ID using deep metric learning, such as those employing multi-stage training with incremental triplet margins [160] or an Easy Positive strategy [151], are typically restricted to static image datasets. As highlighted by Lesort et al. [62], while the concept of CL strongly aligns with the goals of autonomous agents and embodied AI, the integration of these sophisticated CL techniques into a real-time tracking and personalized robotic application remains a complex and sparsely explored area. The Re-ID works proposed in this thesis provide a key contribution to this novel intersection.

2.2 Environmental geometric mapping

When everyone introduces geometric mapping, the focus falls on SLAM. SLAM is a cornerstone of autonomous robotics, defined as the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it. A core challenge in SLAM is the inherent "chicken-and-egg" problem: a map is needed to accurately determine the robot's location (localization), but an accurate location is required to correctly build the map (mapping) [31, 8]. Without a consistent solution, small errors in either process quickly accumulate, leading to map inconsistencies and complete loss of localization, a phenomenon known as drift.

The field of SLAM addresses this circular dependency through probabilistic estimation techniques, which formulate the problem as a single joint probability function over the robot's trajectory $\mathrm{X}_{1:t}$ and the map's features $\mathbf{M}$, effectively exploiting the correlation between localization and mapping estimates to achieve convergence:

$$P(\mathrm{X}_{1:t}, \mathbf{M} \mid \mathrm{Z}_{1:t}, \mathrm{U}_{1:t}) \tag{2.1}$$

where $\mathrm{X}_{1:t} = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ is the sequence of robot poses (or states) up to time t, $\mathbf{M}$ is the environment map, $\mathrm{U}_{1:t} = \{\mathbf{u}_1, \ldots, \mathbf{u}_t\}$ is the sequence of control inputs (odometry or motion commands) up to time t and $\mathrm{Z}_{1:t} = \{\mathbf{z}_1, \ldots, \mathbf{z}_t\}$ is the sequence of observations (sensor readings) up to time t. This joint probability can be broken down

using the Bayes filter and the Markov assumption as explicitly expressed in Section 3.2.

While SLAM can be implemented using various sensors, including cameras (Visual SLAM (V-SLAM)) and 2D laser scanners, 3D LiDAR has become the gold standard for high-precision, large-scale, and real-time mapping in complex environments, particularly in autonomous driving and advanced robotics.

3D LiDAR sensors emit laser pulses to measure distances, generating a dense and geometrically accurate point cloud of the environment. Unlike V-SLAM, LiDAR-based systems are robust to variations in ambient light and texture, offering superior performance in poor lighting or feature-sparse scenes.

Modern 3D LiDAR systems are typically structured into a front-end and a back-end.

The front-end takes care of the odometry and mapping. This module handles the high-frequency sensor data, primarily focusing on scan-to-scan or scan-to-map registration to estimate the robot's pose (position and orientation) between consecutive frames. Early but influential methods, such as LiDAR Odometry and Mapping (LOAM) [159], achieved low drift and high efficiency by exploiting feature extraction from the point cloud data. Subsequent tightly coupled approaches, like Fast-LIO [150], fuse LiDAR measurements directly with data from an IMU at an early stage, significantly improving robustness and accuracy by compensating for sensor motion distortion.

The back-end instead focuses on the global optimization of the problem formulated in the front-end. This module periodically refines the overall consistency of the map and trajectory. It typically uses factor graphs, a graph-based structure where robot poses and landmark positions are nodes, and measurements are edges. A specialized factor graph structure composed only of pose nodes is called a pose graph. A critical function here is LCD, which recognizes previously visited areas and adds constraints to the factor graph. This step is essential for correcting the cumulative drift that is unavoidable in the odometry stage, often leveraging non-linear optimization libraries like GTSAM[2] or g2o[3].

The SoTA in 3D LiDAR is now moving towards integrating these traditional geometric methods with deep learning to handle challenges such as sensor data degradation in high-dynamic or featureless environments, and to manage the compatibility of features in Lifelong SLAM scenarios.

[2]GTSAM: `https://gtsam.org/`
[3]G2O: `https://g2o.com/`

2.2.1 3D LiDAR SLAM

LiDAR-based SLAM has gained significant attention due to its robustness and suitability for mobile robotics, especially in large-scale and challenging environments. Several surveys have documented advancements in this field, including a recent study by Li et al. [66], which categorizes modern SLAM systems into three main approaches: LiDAR-based, multi-sensor, and deep learning-based SLAM. Their analysis examines key components, including front-end data association, LCD, back-end optimization, and map construction.

While vision-based SLAM [158] offers advantages in feature extraction and LCD due to the rich textures captured by RGB cameras, it is highly susceptible to variations in illumination, occlusions, and viewpoint changes. As a result, LiDAR SLAM is often preferred in scenarios where these environmental factors pose significant challenges.

Alternatively, multi-sensor solutions leverage complementary modalities to enhance SLAM performance. For example, the LiDAR-Visual-Inertial (LVI) SLAM system by Shan et al. [125] tightly fuses multiple sensor streams, while other approaches [19, 109] integrate semantic information to enrich map representations. Despite these advancements, the two-dimentional (2D) LiDAR SLAM system developed by Macenski et al. [75] remains one of the most widely adopted solutions, stemming largely from its seamless integration with the ROS2 Navigation Stack for real-world deployment.

3D LiDAR SLAM odometry primarily relies on point cloud matching using least-squares optimization techniques. This typically involves variants of the Iterative Closest Point (ICP) algorithm [11], such as Generalized Iterative Closest Point (GICP) [121], to estimate the robot's motion between consecutive scans.

Methods like KISS-ICP [136] focus on lightweight and efficient LiDAR odometry, relying solely on geometric information and direct scan matching. Similarly, Direct LiDAR Odometry (DLO) [18] proposes a fast and robust LiDAR-based localization method that utilizes a custom ICP solver and combines both scan-to-scan and scan-to-map matching to maintain accuracy in challenging conditions. To counteract the effects of aggressive motions, approaches like POINT-LIO [44], which is an improvement over FAST-LIO2 [150], integrate IMU data to perform high-frequency, point-wise LiDAR-inertial odometry, employing an iterated extended Kalman Filter (KF) for real-time state estimation. However, a well-known limitation of all LiDAR odometry systems is pose drift, *i.e.*, the accumulation of small errors that results in map distortions, making them unreliable for long-term global consistency.

Several 3D LiDAR SLAM solutions take a feature-based approach, where raw point

clouds are processed into distinct, robust features for odometry estimation. A prominent example is LOAM [159], which extracts edge and plane features and sequentially matches them with two separate feature maps. This approach inspired time-efficient adaptations such as LeGO-LOAM [123], which incorporates ground plane constraints into a two-step optimization process, and F-LOAM [138], which improves accuracy by introducing surface smoothness into feature computation while employing both scan-to-scan and scan-to-map strategies. A further refinement, LIO-SAM [124], tightly integrates IMU data to enhance accuracy and performance compared to other LOAM-based approaches. While feature-based approaches offer improved computational efficiency for low-resource devices, their accuracy is inherently limited because feature matching can provide less comprehensive information than direct geometric point cloud alignment. Furthermore, IMU-based methods introduce additional challenges, such as sensor calibration and synchronization, which can cause systematic errors if not handled properly.

To correct the cumulative drift from the odometry front-end, a complete SLAM framework requires a back-end for global map optimization. For instance, ART-SLAM [34] combines scan-to-scan LiDAR odometry, LCD, and pose graph optimization. However, relying solely on scan-to-scan matching can lead to significant odometry drift, which pose-graph optimization alone may struggle to correct in complex environments.

To address these limitations, an approach that employs a multi-level scan matching strategy that avoids using the full map, instead relying on recent local submaps to maintain consistency between consecutive submaps, is proposed in Section 4.2. Additionally, submap-based SC++ [54] LCD is used to identify revisited locations and correct the drift introduced by odometry errors. A pose graph is constructed during mapping and globally optimized using Incremental Smoothing and Mapping (iSAM2) [51] when a loop closure is successfully detected.

Point clouds wrong-reflections filtering through point intensity

LiDAR intensity data, which represents the returned energy level of the emitted laser beams, has emerged as a valuable resource for enhancing LiDAR odometry and SLAM systems. This measurement provides additional information about the reflectance properties of objects' materials [52]. For example, highly reflective metallic surfaces typically yield high-intensity returns, while materials like concrete exhibit medium- to low-intensity returns. By complementing geometric features, these intensity measurements add a form of semantic information to the point cloud, leading to more robust data interpretation.

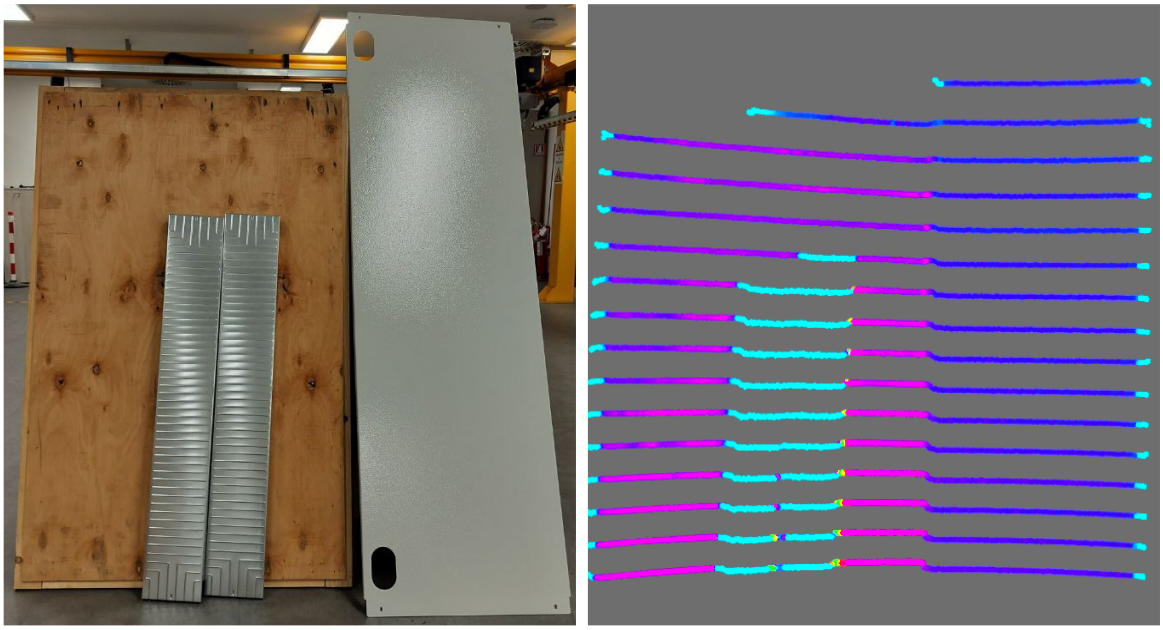
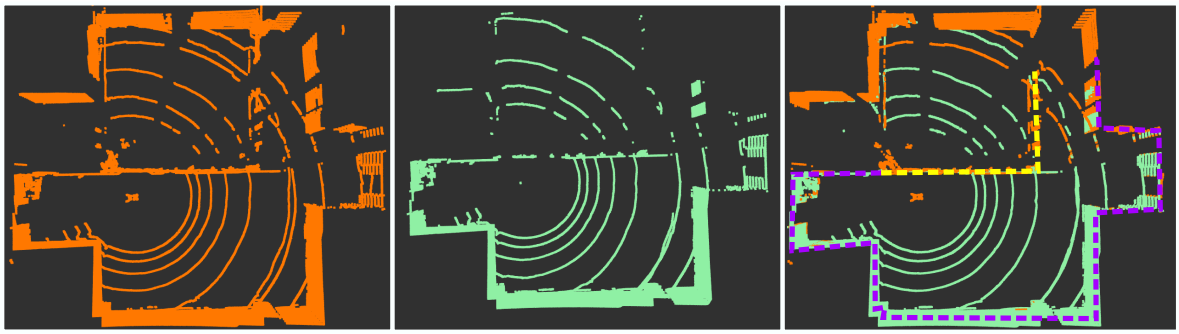


Figure 2.1: Comparison of LiDAR-perceived intensities across diverse materials. The intensity ranges from the lower values of shiny metallic surfaces, represented in light blue, through the medium values of opaque metallic surfaces in violet, to the higher values of the wooden panel in pink.

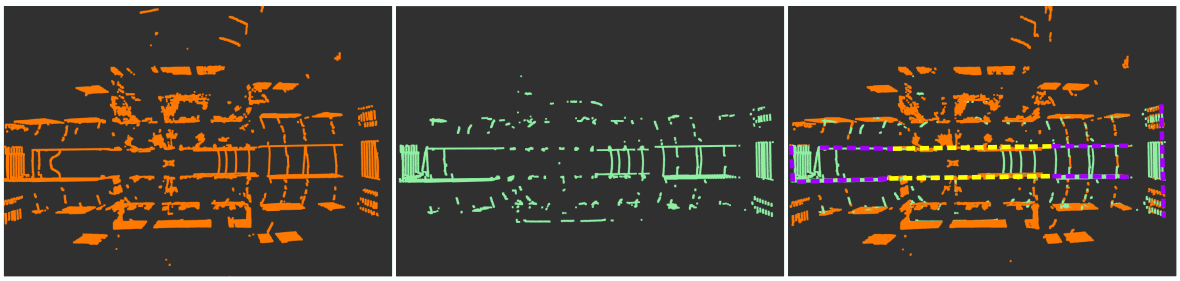
Several studies have utilized intensity data to improve SLAM components. Wang et al. [137] proposed the Intensity Scan Context (ISC), extending the original SC approach [55] by integrating intensity information to enhance loop closure and place recognition capabilities. He et al. [46] incorporated intensity into the ICP algorithm [11], leading to significant performance improvements in LiDAR odometry systems. Other works by Guadagnino et al. [41] for sparse LiDAR odometry and by Wang et al. [139] for large-scale environment mapping also rely heavily on intensity. However, a common practice in these approaches is to filter out or threshold low-intensity points as outliers. This method can inadvertently remove genuine points of interest, such as those on the ground, which naturally exhibit low returns due to their low incidence angle θ (see the left image in Fig 4.11 compared to the right one where the ground is maintained).

The measured LiDAR intensity is a complex function of the target surface's reflectance and roughness, as well as external factors like distance, incidence angle, and the sensor's transmitted energy [52]. For instance, a metallic surface may exhibit lower intensity values than a wooden panel (see Fig. 2.1), primarily due to variations in reflection, surface texture, and the surface's orientation relative to the LiDAR scanner.

In indoor environments, highly reflective surfaces (*e.g.*, mirrors) and transparent



(a) First filtering example with windows on both sides, in a narrow corridor.



(b) Second filtering example with windows only on one side, in a wide room.

Figure 2.2: Examples of the filtering effect are shown in two scenarios that show numerous erroneous reflections caused by the presence of windows. The original point cloud, collected from the robotic platform, is shown in orange, while the point cloud obtained using the ground-aware intensity filter appears in light green. The purple lines in the right images represent the structure of the environment/scenario, and the yellow lines indicate the positions of the windows. The overlap in the right image demonstrates that the proposed filter successfully removes almost all incorrect reflections while preserving ground points.

surfaces (*e.g.*, windows) are common and often lead to erroneous point detections in the LiDAR cloud. These artifacts pose significant challenges for odometry and SLAM systems.

In indoor settings, reflections from objects like windows create significant complications. For example, in Fig. 2.2, windows cause spurious point detections (orange cloud) due to reflections along the yellow-marked boundary. The filtered point cloud (light green) is compared against the ground truth (purple line).

This challenge is especially pronounced in scenarios like navigating in a room with windows on one side that create a mirroring effect, projecting reflections onto both sides of the robot (see Fig. 2.2a) or in a narrow corridor with windows on both sides (see Fig. 2.2b). In these complex scenarios, intensity information is beneficial not only for semantic data enrichment but also as a crucial tool for filtering erroneous measurements caused by reflections and refractions.

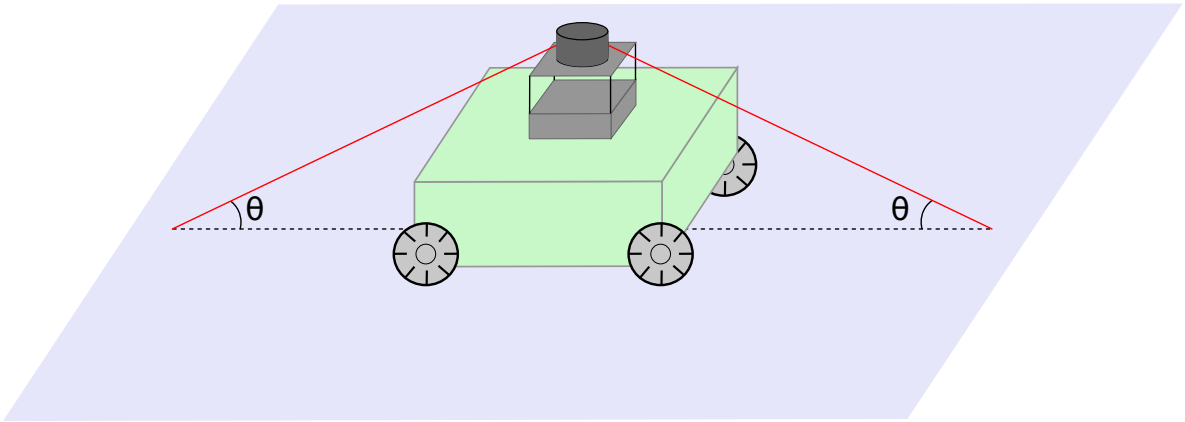


Figure 2.3: The incidence angle θ between the LiDAR beam and the reflective surface significantly impacts the returned energy level, often reducing the point intensity. Points affected in this way would typically be removed by standard threshold filtering methods.

Intensity-based filtering has previously been investigated in autonomous driving applications to remove noise caused by weather phenomena like snow or rain. Early work by Hui et al. [49] used a simple intensity threshold to filter noisy points from raindrops and snowflakes. More recent and sophisticated methods include Low-Intensity Outlier Removal (LIOR) by Park et al. [88], Dynamic Distance–Intensity Outlier Removal (DDIOR) by Wang et al. [142], and Low-Intensity Dynamic Statistical Outlier Removal (LIDSOR) by Huang et al. [47].

These methods are specifically designed to address the sporadic and random noise characteristic of weather events. Consequently, they are often ineffective against consistent reflections (see left images in Fig. 2.2), as they struggle to distinguish a steady, incorrect reflection from a real environmental point. Moreover, they do not intrinsically account for scenarios where erroneous low-intensity readings arise from non-weather factors, such as a low incidence angle between the LiDAR ray and a reflective surface (see Fig. 2.3). This is a critical point in indoor environments where surfaces, like the floor, may exhibit low-intensity values due to a combination of material reflectance and low incidence angles, a challenge that weather-based filters do not directly address.

In this thesis, a simple but effective approach for filtering out reflection points without removing the ones belonging to the ground is proposed. This solution enables the maintenance of ground information essential for navigation while obtaining a correct map of the environment (see Fig. 2.2).

2.2.2 Robust loop closure detection

LCD is a critical function in SLAM, responsible for recognizing a previously visited location and thus providing a constraint that corrects the cumulative drift inherent in odometry. This section reviews the evolution of LCD, with a specific focus on LiDAR-based approaches. For a broader context, comprehensive surveys on LCD techniques across various sensing modalities are available from Tsintotas et al. [133] (focusing on visual LCD) and Arshad et al. [7] (examining both visual and LiDAR methods, with attention to deep learning).

Visual LCD

Visual LCD has been extensively investigated, leveraging the rich texture and feature information captured by RGB cameras.

Early influential methods include FAB-MAP [25], which used a probabilistic Bag-of-Words (BoW) model and Chow–Liu trees based on visual appearance. Later, SeqSLAM [80] improved robustness against environmental variations (*e.g.*, lighting, seasons) by comparing sequences of images rather than individual frames. The widely adopted ORB-SLAM3 [15] performs LCD using a BoW model constructed from Oriented FAST and rotated BRIEF (ORB) features, refining loop candidates through geometric verification and back-end pose graph optimization to mitigate accumulated drift.

More recently, the field has seen a shift toward learning-based approaches. NetVLAD [4] introduced a CNN-based layer that replicates Vector of Locally Aggregated Descriptors (VLAD) encoding [5], enabling end-to-end training for visual place recognition. Furthermore, hybrid methods like SymbioLCD [56] integrate CNN-extracted object semantics with traditional visual BoW features, exploiting the complementary strengths of deep representations and classical features to enhance robustness across diverse environments.

LiDAR LCD

LiDAR-based SLAM and LCD methods are often preferred in industrial and outdoor robotics due to their high sensor precision and robustness against environmental variations, where visual techniques struggle.

Early influential LiDAR SLAM systems like LOAM [159] and its successors, such as LeGO-LOAM [123], extract edge and planar features for pose estimation and incorporate a basic loop closure mechanism within their global optimization framework. A key advancement in traditional LiDAR LCD is SC [55], which forms the foundation

of the proposed method in Section 4.2.3. SC is an efficient and effective descriptor-based approach that generates a rotation-invariant representation of the point cloud. Subsequent improvements, such as SC++ [54], incorporate rotation invariance and acceleration strategies, and ISC [137] utilizes intensity values rather than height for feature representation.

Recent research has focused on enriching the LiDAR input to enhance loop closure accuracy. SA-LOAM [65] improves localization by integrating semantic features, while BoW3D [23] adapts the traditional BoW approach for real-time LCD using 3D point cloud data.

Learning-based solutions have become increasingly prevalent [56, 6, 141]. PADLoC [6] uses a transformer-based architecture that leverages panoptic segmentation during training to enhance both detection and registration. SGLC [141] presents a semantic graph-guided framework that differentiates foreground from background regions for efficient detection and precise 6-Degrees of Freedom (DoF) pose estimation. Similarly, Yang et al. [153] utilize semantic graphs and graph-attention networks to improve the loop closure process. A more holistic strategy is seen in DLC-SLAM [71], which integrates a learning-based denoising module with a dedicated LCD network to increase robustness in noisy environments.

While deep learning-based solutions offer cutting-edge performance, traditional methods remain a preferred choice within some robotics communities [54, 42] due to their lower computational demands and independence from Graphic Processing Unit (GPU) hardware, which is critical for energy-constrained platforms. For instance, Gupta et al. [42] propose a robust LCD pipeline based on point cloud density maps that efficiently detects loops across diverse LiDAR configurations without deep learning. Nevertheless, ongoing advancements in hardware and software optimization are steadily shifting the trend toward more resource-intensive, learning-based approaches.

A notable limitation of SC-based methods, despite their efficiency and effectiveness, is their susceptibility to outliers in LiDAR point clouds, a common issue in real-world mapping due to sensor noise and reflective surfaces. To mitigate this, GSC is introduced in Section 4.2.3. It is a robust variant of SC++ that incorporates statistical analysis of the point cloud to improve the reliability and accuracy of SC++ in the presence of noise and outliers.

2.3 Environmental semantic mapping and understanding

The semantic mapping problem, *e.g.*, integrating high-level semantic labels such as "chair", "wall", "door" into a geometric map has been a major focus in both robotics and autonomous driving. Several surveys offer valuable perspectives on this topic. Achour et al. [1] explore its application in HRC for indoor environments and in a similar way as Pericu et al. [92] employed semantic information to vocally commands robot actions; Xia et al. [147] provide a general analysis of semantic SLAM, examining perception, robustness, and accuracy; and Kostavelis et al. [60] offer a useful historical reference by reviewing the early foundational developments before 2014. One of the latest reviews is the one of Raychaudhuri et al. [103] which discusses semantic representations, map structures (spatial grids, topological, dense, hybrid), encodings (explicit/implicit), evaluation (task-level/extrinsic and map-level/intrinsic), and future directions. Also, Galagain et al. [35] review the semantic mapping under an embedded system perspective. They analyzed semantic-geometric SLAM, Neural Radiance Field (NeRF), and 3D Gaussian Splatting (3DGS) and tested some of them on a Nvidia Jetson Orin AGX to see if they are suitable for mobile robots with constrained performances. They demonstrated that although NeRF and 3DGS are good for realistic representation purposes, they are too slow for real-time applications. Latest trends try to adapt hardware structure to the new software, opposite to the common trend of adapting software to the available hardware.

The last decade in robotics has seen a rapid evolution of semantic mapping techniques, moving from feature-based object recognition to sophisticated deep learning and volumetric fusion approaches.

Initial successful examples demonstrated the performance benefits of incorporating semantic objects into the SLAM loop. Civera et al. [21] presented a monocular SLAM system that used a Speeded Up Robust Features (SURF) [9] feature extractor to check correspondences and reconstruct object geometry. Salas et al. [118] introduced an object-oriented 3D SLAM that used ICP [11] for object pose refinement, demonstrating that semantic objects can improve overall SLAM performance. Pillai et al. [94] developed a monocular SLAM-aware object recognition system based on multi-view object proposals and efficient feature encoding, producing a semi-dense semantic map.

The advent of RGB-D cameras (like Microsoft Kinect) enabled approaches that tightly coupled geometry and semantics directly in 3D. Tateno et al. [132] proposed

a framework that directly manages 3D objects, using a Kinect camera to reconstruct and classify objects while estimating their pose. Xiang et al. [148] introduced the Data Associated Recurrent Neural Networks (DA-RNN) for semantic labeling of RGB-D videos, fusing its output with the KinectFusion algorithm [84] to merge semantic and geometric data. McCormack et al. [78] leveraged a CNN with the ElasticFusion SLAM algorithm [146] to achieve long-term dense correspondences and semantic labeling, even in loopy trajectories. Sunderhauf et al. [131] built on ORB-SLAM2 [82] for geometric reconstruction, using Single-Shot multi-box Detector (SSD) [72] and unsupervised 3D segmentation to place objects.

Recent works have focused on dynamic environments, real-time performance, and structured map representations. Zeng et al. [155] introduced Contextual Temporal Mapping (CT-Map), modeling semantic inference as a Conditional Random Field (CRF) to account for contextual relations and temporal consistency. MaskFusion [115] provided a real-time, object-aware semantic and dynamic RGB-D SLAM that continuously labels and handles dynamic objects, a significant improvement over predecessors. Fusion++ [77] performed object-level SLAM based on a 3D graph map of reconstructed objects, utilizing RGB-D cameras, Mask-RCNN [45] instance segmentation, and the Truncated Signed Distance Function (TSDF) for semantic reconstruction. Grinvald et al. [39] incrementally built a volumetric object-centric map but suffered from limited time performance (1 Hz). Conversely, Pham et al. [93] achieved real-time dense reconstruction and semantic segmentation by using an efficient super-voxel clustering method and CRF with higher-order constraints, running in parallel with a real-time 3D reconstructor. Kimera [113] is an open-source C++ library for real-time metric-semantic Visual-Inertial SLAM, providing modular components for Visual-Inertial Odometry (VIO), pose graph optimization, and dense 3D metric-semantic reconstruction. Bultmann et al. [14] used a Unmanned Aerial Vehicle (UAV) equipped with LiDAR, an RGB camera, and a thermal camera to augment 3D point clouds and image segmentation, generating an allocentric map. Hughes et al. [48] presented a semantic mapping framework that uses only RGB data and exploits 3D dynamic scene graphs [114] to abstract different layers of inference (object, room, building), solving loop closure and mapping problems comprehensively. Hau et al. [43] used RGB-D cameras to reconstruct an allocentric semantic map, employing a CNN keypoint extractor trained on synthetic data for pose estimation and a variant of the Perspective-n-Point (PnP) algorithm to recover object poses from multi-camera views.

Instead, in autonomous driving, the challenge often revolves around multi-sensor fusion in large, outdoor environments. Liang et al. [67] addressed 3D object detection

using LiDAR and RGB camera fusion to estimate object positions via ground estimation and depth completion, training their multi-task network with an end-to-end approach. Chen et al. [19] demonstrated building a semantic map through laser-based semantic segmentation of the point cloud, without requiring any camera data. Li et al. [64] performed LiDAR-based SLAM for geometric mapping, then used a CRF to fuse and optimize camera semantic labels to obtain the final semantic map. Berrio et al. [10] used camera and LiDAR data to construct a probabilistic semantic octree map, explicitly accounting for the uncertainties of all involved sensors. Cheng et al. [20] presented a recent work that uses an RGB camera and LiDAR for semantic segmentation, combined with direct sparse visual odometry and global optimization to include Global Navigation Satellite System (GNSS) data in the mapping process.

This review indicates a key divergence in the SoTA: robotics platforms often rely solely on camera measurements, with experiments typically confined to small indoor environments. Conversely, the autonomous driving scenario already embraces camera-LiDAR fusion for semantic tasks, but often uses high-end, powerful LiDAR sensors (*e.g.*, 128-row vs. common 16-row in robotics) and focuses on outdoor driving challenges.

Hence, the proposed work in this thesis aims to stress the fact that RGB-D cameras and LiDAR sensors are complementary sensors also in robotic semantic applications. For the semantic mapping application, the synergistic use of both sensors allows for correctly localizing objects across different distance ranges, thus significantly improving overall detection accuracy.

A particularly interesting and rapidly emerging area in both research and industry is Open-World Semantic Mapping. In contrast to the traditional semantic mapping approaches reviewed thus far, which rely on a fixed, predefined set of classes (*e.g.*, "door", "chair", "fire extinguisher"), the open-world paradigm leverages the generalizability and grounding capabilities of Large Language Models (LLMs), or, more accurately, Vision-Language Models (VLMs), to provide a mapping framework that does not require retraining for novel objects or concepts. This allows a robot to recognize and localize any object described in natural language, extending the map's semantic utility far beyond its initial training data. Although this field is quite new, a recent overview of existing works on open-world semantic mapping is provided by Miao et al. [79].

The field is evolving quickly, with a few foundational works setting the current SoTA. OpenScene [90] is one of the first prominent open-world semantic mapping systems. OpenScene employs the powerful VLM Contrastive Language-Image Pre-training (CLIP) [101] to bridge the visual and textual domains. The core idea is to associate CLIP embeddings (high-dimensional vectors representing visual and semantic informa-

tion) with each point in the 3D point cloud. Once the map is built, a user can query the map using a text input (*e.g.*, "locating the fire extinguisher" or "show me all flat surfaces") and the system uses the text embedding to directly discriminate and segment the corresponding objects, affordances, or specifics within the map. This enables flexible and zero-shot semantic query capabilities.

Similar to OpenScene, OVO-SLAM (Open-Vocabulary Object SLAM) [76] integrates CLIP embeddings, but focuses on an online semantic mapping framework that maintains consistency as the robot moves. By structuring the map around individual objects identified through the VLM and integrating these into the process, OVO-SLAM achieves real-time, open-vocabulary localization and mapping, allowing the map to be continuously queried and updated with new semantic concepts as they are encountered.

Taking a different approach to representation and visualization, Yang et al. [152] integrated open-world semantic mapping into a 3DGS framework (OpenGS). 3DGS is a novel, highly efficient 3D rendering technique. OpenGS leverages this to provide not only open-vocabulary semantic recognition but also a smoother, high-fidelity visual experience when querying and visualizing the semantic map. This combination of open-world semantics with next-generation rendering addresses both the functional (recognition) and experiential (visualization) aspects of large-scale mapping.

The literature review on semantic mapping reveals a clear evolution from feature-based object recognition to sophisticated multimodal, deep learning-driven fusion systems. Initial works demonstrated the performance benefits of incorporating object geometry into the 3D loop [21, 118], leading to the wide adoption of RGB-D volumetric fusion methods that tightly coupled semantics and 3D structure [78, 115]. More recent robotics research emphasizes building structured, allocentric maps using semantic scene graphs [114, 48] and robust multi-sensor frameworks [113].

A fundamental divergence exists between domains: autonomous driving has successfully integrated high-power LiDAR-camera fusion for outdoor semantic tasks [67, 20], whereas many robotics solutions historically favored camera-only approaches in small indoor spaces [94]. This disparity highlights a crucial opportunity: combining the geometric accuracy of LiDAR with the rich semantic detail of RGB-D cameras is essential for creating highly accurate and semantically dense maps across the diverse operational range required by advanced mobile robots.

Finally, the emerging field of Open-World Semantic Mapping [79] signals the next frontier. By leveraging VLMs like CLIP [90, 76], these systems move beyond predefined classes to enable natural language querying of the environment, representing a significant step toward truly flexible and human-interpretable autonomous systems.

Chapter 3

Problem statement

Throughout this thesis, different problems were addressed. Some of them have similar foundations but evolve in various fields, while others are specific to their respective sectors. In the following, the main problems to enhance the context awareness of mobile robots are presented.

3.1 Data association and corresponding problem

The fields of robotics, computer vision, and sensor fusion heavily rely on establishing accurate relationships between sets of measurements or observations, a challenge encapsulated by two related concepts: data association and the correspondence problem. While often used interchangeably, subtle differences in context and scope exist.

The Correspondence Problem (CP) is fundamentally a geometric and often static challenge. It involves identifying pairs or sets of features, points, or observations that represent the same physical entity in the real world across different views, images, or datasets taken at the same or similar time. This problem has been primarily faced in stereopsis (3D reconstruction from two or more camera views), Structure from Motion (SfM), and image mosaicking. In these cases, the goal is to find which point in one image A maps to which point in another image B. Usually, it focused on spatial alignment and geometric consistency. The features are typically extracted from different sensors or viewpoints simultaneously.

Data Association (DA) is a broader and often dynamic problem rooted in state estimation and tracking. It concerns the task of assigning a set of new measurements (*e.g.*, from a sensor sweep) to a set of existing tracks or entities that are being monitored over time. It is essential in Multi-Target Tracking (MTT), SLAM, and sensor fusion

(*e.g.*, radar, lidar, and camera fusion). Its goal is to correctly label a new measurement as belonging to an existing track, a new track, or as clutter (false alarm). It focuses on temporal consistency and probabilistic modeling. It manages the uncertainty in measurement origins as targets move and sensors provide noisy data across sequential time steps.

The key differences are resumed as follows: CP focuses on spatial/geometric alignment while DA consider also temporal or tracking consistency among frames, the input in CP are generally features extracted from different views or contexts while DA has measurement extracted from sequential time steps and the goal is finding the correct geometric match in CP while determining the origin or the belonging of the current measurement in DA.

In essence, the correspondence problem is often a sub-problem that needs to be solved within a larger data association framework, especially in SLAM, where associating features between two successive frames of a moving camera is a form of correspondence that is critical for the overall tracking process. However, data association extends this to the entire history of tracked objects and the probabilistic management of target identities over time.

3.1.1 The data association problem in human re-identification

In the field of computer vision and surveillance, Human Re-ID is fundamentally cast as a specialized data association problem. Specifically, it is the challenge of establishing identity correspondence between a person's image, the query ($\mathbf{Q}$), and a collection of images from a database, the gallery ($\mathbf{J}$). Both sets are typically captured by non-overlapping camera views ($\mathbf{C}_i \neq \mathbf{C}_j$) at potentially different times ($t_1 \neq t_2$).

The core objective is to determine a binary correspondence function, $f : \mathbf{Q} \times \mathbf{J} \rightarrow \{0, 1\}$, where $f(\mathbf{Q_i}, \mathbf{J_i}) = 1$ if the query image $\mathbf{Q}_i \in \mathbf{Q}$ and the gallery image $\mathbf{J}_i \in \mathbf{J}$ depict the same individual, and 0 otherwise.

Unlike classical geometric correspondence (*e.g.*, matching a point in a stereo pair), the Re-ID data association problem is inherently ill-posed due to the extreme variability introduced by real-world surveillance environments. This variability is often categorized into two main challenges: intra-class variation (appearance discrepancy of the same person) and inter-class similarity (ambiguity between different people).

In intra-class variation, the same person's appearance can change drastically between camera views, making the correspondence difficult to establish. Key factors include:

- *Viewpoint and pose change*: A person's body structure and visible features (*e.g.*,

a backpack) change significantly when viewed from a side-angle versus a front-angle. This causes major spatial misalignment of corresponding body parts across images.

- *Illumination change and color distortion*: Different lighting conditions (shadows, indoor vs. outdoor, time of day) alter the color and texture of clothing, which are the primary visual cues for Re-ID.
- *Occlusion and deformations*: Partial or complete obstruction by other people or objects, along with non-rigid human body movement, can hide critical features, reducing the reliable information available for correspondence matching.
- *Appearance Changes*: Changes in a person's appearance, such as different clothes or hairstyle.

inter-class similarity (ambiguity between different people) Instead, inter-class similarity means that different individuals can appear visually similar, leading to false correspondences.

- *Clothing homogeneity*: In many environments, different people wear identical or near-identical clothing (*e.g.*, black coats, blue jeans), causing a high degree of visual ambiguity that confuses feature-based matching algorithms.
- *Background clutter*: Similarities in non-person regions (background or objects carried) can dominate appearance-based feature vectors, leading to incorrect matches between different individuals observed in similar environments.

The problem formulation for Re-ID

To address this ill-posed correspondence, the Re-ID problem is typically reformulated as a metric learning or similarity ranking task. Instead of directly seeking a geometric correspondence, the goal shifts to learning a robust feature representation $\phi(\cdot)$ and a distance metric $D(\cdot, \cdot)$ such that:

$$D(\phi(\mathbf{q}_i), \phi(\mathbf{j}_j)) \ll D(\phi(\mathbf{q}_i), \phi(\mathbf{j}_k)) \tag{3.1}$$

where $\mathbf{q}_i$, $\mathbf{j}_j$, and $\mathbf{j}_k$ are feature vectors representing a person (*e.g.*, embedding vectors extracted with a specialized neural network), and the person ID associated with $\mathbf{q}_i$ is equal to the one of $\mathbf{j}_j$ and different from the one of $\mathbf{j}_k$. The solution to the Re-ID correspondence problem, therefore, lies in developing a feature embedding and metric that

is discriminative for identity while simultaneously being invariant to the environmental and appearance variations outlined above. This transformation from raw pixel space to an identity-invariant feature space is the central challenge addressed by modern Re-ID research.

3.1.2 The corresponding problem in loop closure detection

In SLAM systems, the process of LCD represents a critical instance of the corresponding problem or a simplified data association one, as already stated before. It addresses the challenge of recognizing that the robot or sensor has returned to a previously visited location after a long, potentially drift-accumulating trajectory.

The data association task in LCD operates at a global, pose-graph level, focusing on establishing a connection between the current sensor measurement and a historical keyframe in the map.

The actors in this problem are

- *The measurement set*: The new measurement ($\mathbf{z}_k$) is the sensory data (*e.g.*, an image, a LiDAR scan) acquired at the current time step k.
- *The track/map set*: The existing tracks/entities correspond to the set of historical keyframes or map nodes ($\mathbb{M} = \{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_{k-L}\}$) stored by the SLAM system, where L is a large temporal separation.

The data association algorithm must decide if the current measurement $\mathbf{z}_k$ corresponds to any of the past map nodes $\mathbf{M}_i \in \mathbb{M}$. If an association is made ($\mathbf{z}_k \leftrightarrow \mathbf{M}_i$), it implies a spatial return to $\mathbf{M}_i$'s location, thereby creating a loop closure constraint.

The data association problem for LCD is particularly challenging because the relationship must be established across significant spatial and temporal gaps. Many difficulties arise in this problem, among which can be found:

- *Perceptual aliasing* (false positives): The most severe difficulty is perceptual aliasing, where distinct physical locations appear visually or structurally similar to the sensor (*e.g.*, repeating corridors, symmetrical buildings, or similar foliage). The system must avoid falsely associating the current location with a similar-looking, but incorrect, past location. This failure introduces an erroneous constraint that can corrupt the entire map.
- *Appearance change* (false negatives): Over long time spans, the environment's appearance can change significantly (*e.g.*, lighting, weather, seasonal variations,

presence of transient objects). These changes can prevent a correct association, leading the system to fail to recognize a true loop closure and thus continue to accumulate localization drift.

- *Accumulated uncertainty*: Since the system's pose estimate has drifted significantly over the long trajectory, the expected transformation between the current pose and any past map node is highly uncertain. This makes probabilistic validation of the association much more difficult compared to local data association between sequential frames.

In summary, the data association in LCD is a high-stakes, long-range identity validation task that must overcome the ambiguity of perceptual aliasing using techniques that are robust to drastic appearance changes. Once the data association is successfully established, the system leverages this constraint to perform global map optimization.

3.1.3 Connection between loop closure detection and person re-identification

The LCD and Person Re-ID problems, while applied in different domains (SLAM vs. surveillance/tracking), are fundamentally and structurally connected because they are both specialized variations of the DA problem.

The key connection is that LCD can be formally defined as a place Re-ID problem.

The most direct connection lies in their shared objective function: finding a match across a significant gap in observation time and/or viewpoint, using only appearance-based features.

Both problems structure the correspondence as a retrieval task. While LCD uses the current keyframe in input, Re-ID uses the current image. While LCD uses a database of historical map nodes, Re-ID uses a gallery of person images captured in the past or a model as in the specific case considered. The goal of LCD is to find a location correspondence between keyframes, while in Re-ID the correspondence is between person identities.

In both cases, the solution relies on learning a powerful feature representation $\phi(\cdot)$ and a distance metric $D(\cdot, \cdot)$ to solve the identity/location correspondence by ranking similarity scores.

Some algorithmic principles are the same. Among the most relevant are:

- *Feature embedding*: both LCD and Re-ID use Deep CNNs or geometric metrics to map the input to a compact, low-dimensional embedding vector or represen-

tation. The goal is to make this embedding highly discriminative yet robust to environmental variations.

- *Metric learning*: Both tasks commonly employ metric comparison to ensure the learned metric space separates positive pairs (same identity/location) from negative pairs (different identity/location) by a large margin.

- *Robustness to ill-posed challenges*: Both systems must overcome issues of high intra-class variation (where the appearance of the target, either a person or a place, changes) and Perceptual Aliasing (where different targets look visually similar).

Fundamental differences and cross-pollination

While the foundations are similar, the inherent constraints and resulting feature focus differ significantly. Place recognition often works with wide static environment structures, with difficulties stemming from dynamic objects and transient lighting changes, or reflections/refractions. In person Re-ID, the features are extracted from transient identity attributes, and the challenges arise from the background clutter, different camera viewpoints, and pose variation.

The structural similarity means that breakthroughs in one field often transfer to the other. For example, techniques developed for the highly complex Person Re-ID task (*e.g.*, specialized attention mechanisms) can be adopted to create more robust, appearance-based descriptors for LCD. Also, the idea of dividing the image into local parts and establishing local correspondence before a global match (*e.g.*, body parts in Re-ID) is sometimes applied in LCD to mitigate the effect of dynamic objects, where a matching static image block can still confirm a loop closure.

3.2 The localization and mapping problem

One core challenge faced by autonomous systems is the SLAM problem. SLAM is defined as the computational problem of incrementally building a map of an unknown environment while simultaneously keeping track of the agent's location within that map. The difficulty lies in the co-dependency: an accurate map is needed for localization, and accurate localization is needed to build a consistent map.

Formally, SLAM aims to estimate the sequence of agent poses $\mathrm{X}_{1:t} = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ and the map of environmental features $\mathbf{M}$ given a history of sensor observations $\mathrm{Z}_{1:t} =$

$\{\mathbf{z}_1, \ldots, \mathbf{z}_t\}$ and control inputs $\mathrm{U}_{1:t} = \{\mathbf{u}_1, \ldots, \mathbf{u}_t\}$. The desired output is the joint posterior probability:

$$P(\mathrm{X}_{1:t}, \mathbf{M} | \mathrm{Z}_{1:t}, \mathrm{U}_{1:t}) \tag{3.2}$$

which is exactly Eq. (2.1) presented in Section 2.2. In that section, the SLAM problem and its back-end and front-end formulation have been introduced for the sake of completeness in reviewing the current works. Here, an additional and more complete problem presentation is provided.

One of the primary goals is often to estimate only the current pose $\mathbf{x}_t$ recursively and the map $\mathbf{M}$ in an online fashion:

$$P(\mathbf{x}_t, \mathbf{M} \mid \mathrm{Z}_{1:t}, \mathrm{U}_{1:t}) \tag{3.3}$$

This is typically computed in two steps: the prediction and the update step. The prediction step is modeled using the posterior probability of state $\mathbf{x}$ at time t and the map $\mathbf{M}$, knowing the control inputs $\mathrm{U}_{1:t}$ and the sensor measurements $\mathrm{Z}_{1:t-1}$: It is predicted from the state posterior probability and the previous joint posterior probability at time $t-1$ following Eq. (3.3)

$$P(\mathbf{x}_t, \mathbf{M} \mid \mathrm{Z}_{1:t-1}, \mathrm{U}_{1:t}) = \int P(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)\, P(\mathbf{x}_{t-1}, \mathbf{M} \mid \mathrm{Z}_{1:t-1}, \mathrm{U}_{1:t-1})\, d\mathbf{x}_{t-1} \tag{3.4}$$

where $P(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$ is the motion model used to estimate the robot motion in the last time step, a term often referred to as robot odometry, and $P(\mathbf{x}_{t-1}, \mathbf{M} \mid \mathrm{Z}_{1:t-1}, \mathrm{U}_{1:t-1})$ is the previous joint posterior probability (Eq. (3.3) at time $t-1$).

The prediction is then corrected in the update step using the latest observation $\mathbf{z}_t$:

$$P(\mathbf{x}_t, \mathbf{M} \mid \mathrm{Z}_{1:t}, \mathrm{U}_{1:t}) = \frac{P(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{M})\, P(\mathbf{x}_t, \mathbf{M} \mid \mathrm{Z}_{1:t-1}, \mathrm{U}_{1:t})}{P(\mathbf{z}_t \mid \mathrm{Z}_{1:t-1}, \mathrm{U}_{1:t})} \tag{3.5}$$

where $P(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{M})$ is the measurement model.

The previous formulation relies on two key assumptions:

- *Markov assumption for motion*: The current pose $\mathbf{x}_t$ depends only on the previous pose $\mathbf{x}_{t-1}$ and the current control $\mathbf{u}_t$:

$$P(\mathbf{x}_t \mid \mathrm{X}_{1:t-1}, \mathrm{U}_{1:t}, \mathbf{M}) = P(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$$

- *Observation independence*: The current observation $\mathbf{z}_t$ is conditionally independent of all previous poses and observations given the current pose $\mathbf{x}_t$ and the map $\mathbf{M}$:

$$P(\mathbf{z}_t \mid \mathrm{X}_{1:t}, \mathrm{Z}_{1:t-1}, \mathrm{U}_{1:t}, \mathbf{M}) = P(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{M})$$

The SLAM problem can be factored using the previous assumption into a joint probability of all states, the map, the observations, and controls as:

$$P(\mathrm{X}_{1:t}, \mathbf{M}, \mathrm{Z}_{1:t}, \mathrm{U}_{1:t}) = P(\mathbf{x}_0)P(\mathbf{M}) \prod_{k=1}^{t} P(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_k) \prod_{k=1}^{t} P(\mathbf{z}_k \mid \mathbf{x}_k, \mathbf{M}) \tag{3.6}$$

where $P(\mathbf{x}_0)$ is the initial pose distribution and $P(\mathbf{M})$ is the prior over the map, while the first product represents the motion model and the second the observation one. Modern SLAM methods often focus on finding the most probable estimate as the Maximum A Posteriori (MAP), which maximizes the term $P(\mathrm{X}_{1:t}, \mathbf{M} \mid \mathrm{Z}_{1:t}, \mathrm{U}_{1:t})$, which is equivalent to maximizing the full joint distribution in Eq. (3.6) assuming a uniform denominator. Nowadays, the above problem is practically formulated using factor graphs [29]. A deep review of the SLAM problem formulated using factor graphs can be found in the SLAM Handbook [73].

As already introduced, the SLAM problem is often divided into back-end and front-end. The back-end solves the optimization problem formulated in the factor graph following the assumptions and joint probabilities presented above. Instead, the front-end focuses on how the factor graph is created. In the following, the focus is on the front-end problem formulation, which is one of the main problems faced in this thesis.

In the following, the evolution of SLAM solutions is distinguished by the type of information used for environmental representation and data association: Geometric (relying on raw spatial features) and Semantic (incorporating cognitive-level information).

3.2.1 Geometric SLAM: foundational techniques

Classical SLAM focuses purely on geometric consistency and spatial fidelity within a local coordinate system. The environment is modeled using low-level sensor data (*e.g.*, points from LiDAR) or abstract spatial primitives derived directly from them:

- *Sparse point features*: These are highly repeatable, unique corner points or keypoints (*e.g.*, Scale-Invariant Feature Transform (SIFT), ORB features) used in V-SLAM for fast processing and robust tracking.

- *Dense representations*: Voxel grids or Signed Distance Functions (SDFs) are employed to model the volume of the scene, typically in RGB-D or dense LiDAR SLAM, providing a high-fidelity 3D map.

- *Structural primitives*: Line segments, planes, and cylinders, are utilized, particularly in LiDAR and 2D-SLAM, to capture the dominant structure of indoor environments.

The environment map $\mathbf{M}$ is typically represented as a factor graph, where nodes are keyframes (poses) and edges are the relative geometric transformations between them. Factor graphs are used to model and solve the probability in Eq. 3.3.

The previously presented prediction step, referred to here as the odometry step, suffers from drift accumulation. The primary weakness is that pose estimates from odometry are prone to accumulating small errors over time, causing the estimated trajectory to drift away from the true path. To correct drift, the LCD module is activated. LCD is a geometric data association problem that seeks to identify if the current observation $\mathbf{z}_t$ matches a previously mapped location $\mathbf{M}_i$. Upon a successful match, a new, highly accurate constraint is added to the factor graph, which is then globally optimized to distribute the accumulated error across the entire map.

Nevertheless, geometric features are fragile. They are highly sensitive to viewpoint change, illumination variation, and the presence of dynamic objects, leading to frequent false negative loop closures (missed opportunities) or, critically, false positive loop closures (incorrect associations that corrupt the map).

3.2.2 Semantic SLAM: cognitive integration

Semantic SLAM augments the geometric framework by integrating cognitive-level understanding of the environment, treating scenes not just as points and planes, but as collections of meaningful objects.

The SLAM problem is enriched with more meaningful semantic features extracted via deep neural networks, including:

- *Object instances*: Bounding boxes and instance masks for specific objects (*e.g.*, "chair 1", "table 2").

- *Semantic labels*: Per-pixel classification (semantic segmentation) for surface types (*e.g.*, "wall", "floor", "ceiling").

- *Place categories*: Classification of the scene (*e.g.*, "living room", "street intersection").

The map evolves into a semantic map that can be associated with a semantic factor graph. This graph retains the geometric structure but enriches it with semantic relationships and labels, enabling querying by object identity.

The introduction of semantics inherently enhances the robustness of data association by leveraging semantic invariance. A chair remains a chair regardless of a shift in lighting or a change in viewing angle. By associating keyframes based on the consistent presence and geometric configuration of invariant semantic objects, the system achieves a more robust form of place recognition (or place Re-ID), mitigating the risks of perceptual aliasing that plague geometric methods. Semantic relationships provide powerful constraints for global optimization. For instance, knowing that a "ceiling" is always parallel to a "floor" or that a "lamp" is typically above a "table" allows the system to enforce structural consistency, leading to a more accurate and globally plausible map than is possible with geometric errors alone.

The principal challenge of Semantic SLAM lies in effectively fusing two distinct modalities:

- *Low-level geometry*: Precise, but fragile and abstract.
- *High-level semantics*: Robust and meaningful, but inherently coarse and subject to object detection errors.

The problem transitions from a purely mathematical optimization problem to a challenge of robust fusion and data-driven reasoning to maximize the consistency and utility of the final map.

Chapter 4

Method

Due to the multiplicity of methods presented, the proposed method is structured into three main sections: (*i*) Re-ID for HRC in Section 4.1, which addresses the visual data association and tracking problem, (*ii*) environmental perception through geometrical matching in Section 4.2, which tackles the SLAM problem and the corresponding issue of loop closure, and (*iii*) robot contextual understanding and interaction with semantics in Section 4.3, where a solution to enhance the robot's cognitive capabilities is introduced.

4.1 Human-robot collaboration through re-Identification

For successful interaction, robots must be able to detect and recognize their human collaborators. To emulate human abilities, visual Re-ID is employed to address this challenge, as mobile robots are nearly always equipped with camera sensors. Incremental solutions are presented that offer several improvements over the current SoTA and address the limitations encountered in prior iterations.

4.1.1 Baseline approach for visual Re-ID

The first proposed approach, specifically FollowMe [111], utilizes visual and depth data obtained from an RGB-D camera to execute four consecutive steps: (*i*) Detection, (*ii*) Re-ID, (*iii*) Localization and filtering, and (*iv*) Gesture detection. The complete pipeline initially proposed for solving this problem is presented in Fig. 4.1.

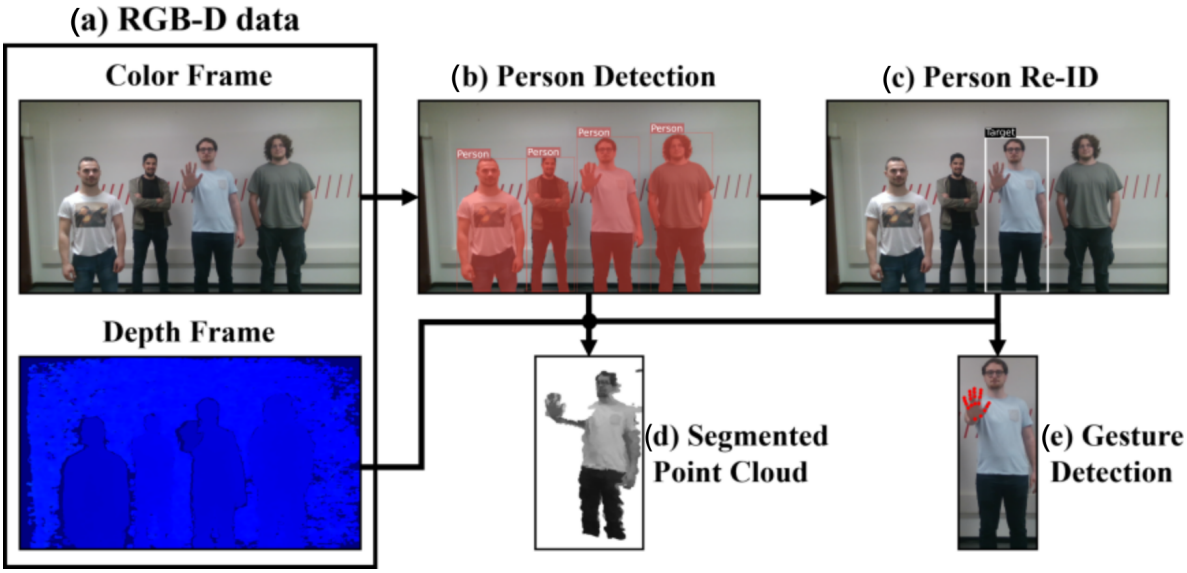


Figure 4.1: FollowMe perception pipeline: (a) RGB and Depth images acquisition; (b) person detection through Yolact++; (c) person Re-ID using a deep neural network and human features distance; (d) person localization using the point cloud and re-identified person mask; (e) gesture detection to send commands to the robot. Reprinted, with permission, from [111], © 2023 IEEE.

People detection

To detect individuals in the image (Fig. 4.1a), an instance segmentation algorithm is employed. Instance segmentation is a specific form of image segmentation that involves detecting object instances and delineating their boundaries. YOLACT++ [12] is a one-stage instance segmentation framework that offers several advantages over existing architectures, with prediction speed being one of the most significant. Although new solutions have been presented in the literature, YOLACT++ is among the existing frameworks capable of delivering "real-time" instance segmentation inference. In this work, a standard model pre-trained on the 80 classes of COCO [68] (which includes the “person" class) is adopted (Fig. 4.1b). YOLACT++ extracts both bounding boxes and segmentation masks corresponding to the trained classes, where the mask values are 1 for the pixels where the object is localized and 0 elsewhere. To precisely isolate the person’s contours and completely remove background information, the original images are element-wise multiplied by their corresponding binary masks. This operation allows the Re-ID module to extract cleaner personal features based solely on the individual by eliminating background noise. The resulting images, each representing a detected person, are subsequently made available to the Re-ID module.

Person re-Identification

To correctly re-identify an individual, an additional neural network is employed to extract distinctive features. This step is necessary because features derived from YOLACT++ [12] and YOLO [105] networks in general are highly generalized within the "person" class, meaning that different individuals possess very similar features in the YOLO embedding space.

Person Re-ID was originally developed for identifying a subject across non-overlapping camera views. Given a query image of a person, the objective is to recognize the same individual in images acquired by different cameras by analyzing and extracting appearance information alone (without relying on biometric cues). To mitigate the limitations of the YOLO embedding space, the popular MMT [36] framework is utilized for extracting features for the Re-ID module. Specifically, only the pre-training phase is considered. In this phase, a deep learning network is trained to recognize people in images. The segmented output images are passed into this network to extract the features that will subsequently be used to re-identify the target person. The features from the last layer of the trained network are specifically considered. The Re-ID process is divided into two distinct phases: calibration and identification.

The calibration phase is executed at the start of the application. The target person is instructed to move at various distances in front of the camera for a few seconds, simulating the movements and postures they will typically adopt while walking, in order to collect images. This calibration step is critical because the diversity in person representations captured during this stage directly influences the Re-ID system's robustness against illumination changes and occlusions. The collected images are divided into two groups: the person calibration set (with N_c elements) and the threshold set (with N_λ elements). The processed images from the person calibration set are input first into YOLACT++ for person masking and then into the Re-ID model for feature extraction.

Let D denote the dimension of the feature vector $\mathbf{x}$ extracted by the Re-ID model; each person can be associated with a vector $\mathbf{x} = [x_1, \ldots, x_D]^T$. For each image in the person calibration set, the corresponding features are extracted and then used to compute the associated mean $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_D]^T$ and standard deviation $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_D]^T$, which are required for target identification in the second phase.

Thus, given a new image of a person, the corresponding feature vector $\mathbf{x}'$ is computed. This person can be identified as the target by measuring the following feature-space weighted distance between $\mathbf{x}'$ and the distribution defined by $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$:

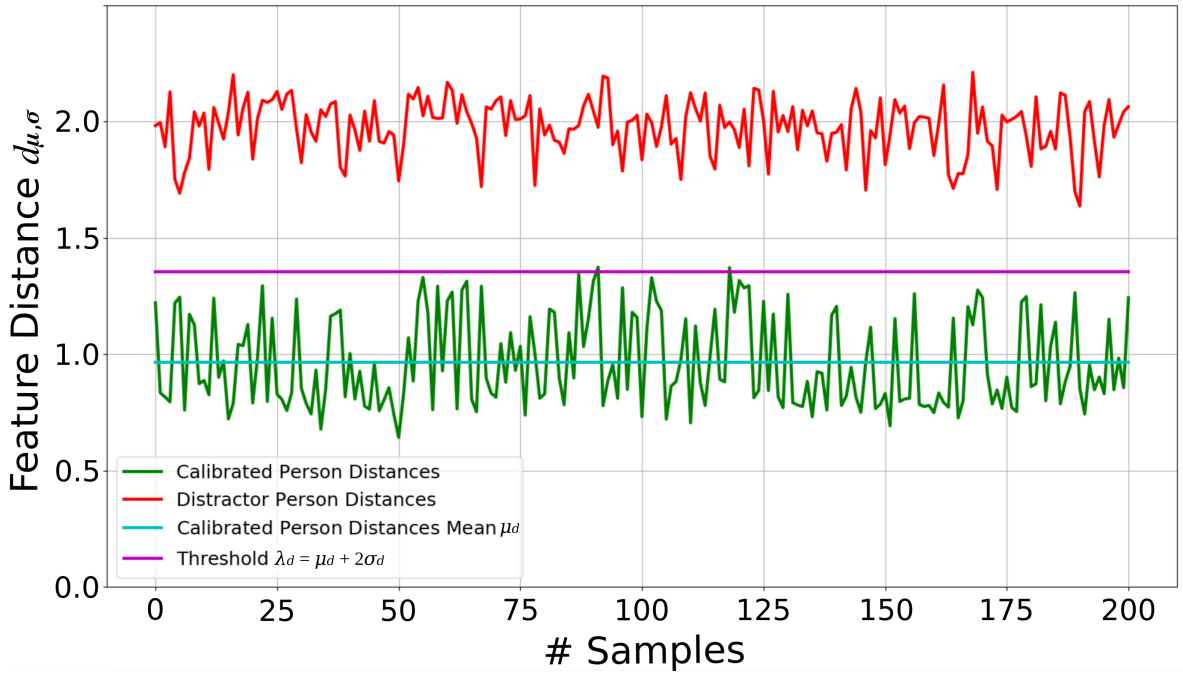


Figure 4.2: In green, the plot of the feature distances for a calibrated person used to compute the target threshold (magenta horizontal line). In red, the distances computed between the features of another person (the distractor) and the target template. The distances are computed over 200 sample images. Reprinted, with permission, from [111], © 2023 IEEE.

$$d_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\mathbf{x}') = \sqrt{\frac{1}{D}\sum_{i=1}^{D}\left(\frac{x'_i - \mu_i}{\sigma_i}\right)^2}, \tag{4.1}$$

where the subscript i represents the i-th index of the corresponding vector.

The previously acquired threshold set is utilized to compute a distance threshold λ_d: if the measured distance $d_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\mathbf{x}') \leq \lambda_d$, the person associated with feature $\mathbf{x}'$ will be identified as the target observed during the calibration phase.

Specifically, for each image in the threshold set, the associated feature distance, calculated using Eq. (4.1), is computed, yielding the overall mean distance μ_d and standard deviation σ_d. The calibrated person threshold is then set as $\lambda_d = \mu_d + 2\sigma_d$. This setting is chosen to encompass the majority of target samples without being excessively high, which would risk failing to filter out distractor people (any person who is not the calibrated target). Fig. 4.2 illustrates an example set of distances for a calibrated person and a distractor, along with the calculated λ_d value. In this work, we prioritize minimizing the number of wrongly re-identified targets (false positives)

because such errors are difficult to mitigate. Conversely, false negatives (the calibrated person is present but not re-identified) are managed. If the target is not recognized in certain frames, a KF integrates the last known detection for a specified duration, thereby enhancing the application's robustness.

During the execution of the FollowMe application, new images are continuously acquired, and people are detected and filtered. In the identification step, similarly to the calibration phase, the background is removed from the processed images, which are then passed to the Re-ID model. The feature vectors obtained from the Re-ID inference for all detected people are collected. Finally, the feature distances (Eq. (4.1)) are calculated. The person exhibiting the smallest distance, provided it is less than the selected threshold λ_d, is chosen as the target (Fig. 4.1c). If the distances for all detected people exceed the threshold, no target is detected.

Localization and filtering

In the localization and filtering step, the mask determined during the Re-ID process is used to compute the target's position, ${}^{\mathrm{c}}\mathbf{p}$, within the camera reference frame. Utilizing both RGB and depth data, and leveraging the re-identified person's mask, the segmented target point cloud is constructed (Fig. 4.1d). The position ${}^{\mathrm{c}}\mathbf{p}$ is then calculated as the 3D centroid of this point cloud.

To filter out noise caused by the robot's movements, vibrations, and inherent mask detection errors, a KF is applied to ${}^{\mathrm{c}}\mathbf{p}$. The KF stops the integration when measurement updates are not received for a predefined duration, known as the expiration time, t_{exp}. Since the KF-filtered position, ${}^{\mathrm{c}}\mathbf{p}_{filt}$, is defined in the moving camera frame, the following homogeneous transformation is applied:

$$ {}^{\mathrm{m}}\mathbf{p}_{filt} = {}^{\mathrm{m}}\mathbf{t}_{c}\,{}^{\mathrm{c}}\mathbf{p}_{filt} \tag{4.2} $$

where ${}^{\mathrm{m}}\mathbf{p}_{filt}$ is the target position expressed in the map reference frame, and ${}^{\mathrm{m}}\mathbf{t}_{\mathrm{c}}$ is the translation from the camera reference frame c to the map frame m, which is obtained from the navigation stack localization module.

Finally, the 2D goal position ${}^{\mathrm{m}}\mathbf{p}_{goal}$ is computed by projecting ${}^{\mathrm{m}}\mathbf{p}_{filt}$ onto the 2D map plane (XY plane). Concurrently, the heading angle θ between the camera reference frame and ${}^{\mathrm{m}}\mathbf{p}_{filt}$ is calculated. The goal data (i.e., ${}^{\mathrm{m}}\mathbf{p}_{goal}$ and θ) are managed by the FSM, which forwards them to the navigation module to drive the robot towards the target. An overview of the relevant transformation frames is presented in Fig. 4.3.

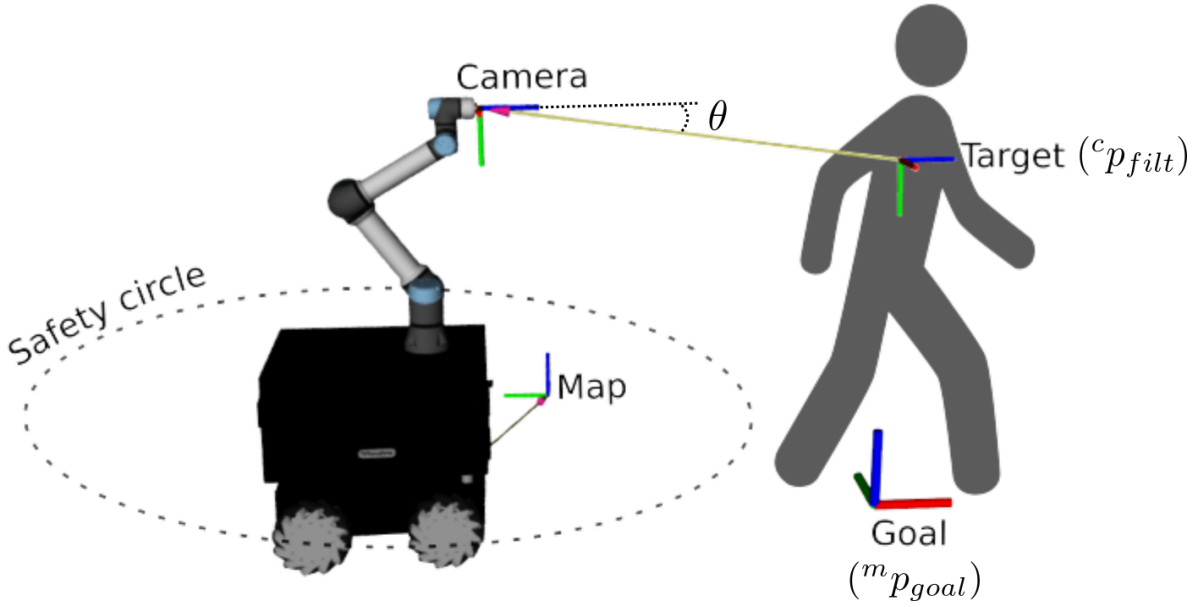


Figure 4.3: Overview of relevant transformation frames and representation of the safety circle used during the FollowMe application. Reprinted, with permission, from [111], © 2023 IEEE.

Gesture detection

To enable interaction between the robot and the human operator, hand gesture recognition capabilities have been developed for the robot. Hand gestures are a commonly used communication tool for exchanging non-verbal information, not just in robotics [53]. By utilizing gestures, it is possible to interact with the robot and convey commands that trigger specific robot functionalities and behaviors. In this work, gestures are used to stop and reactivate the FollowMe task, but other commands can be incorporated if necessary.

The gesture detection module was implemented using the Mediapipe hand tracking algorithm [157] combined with a SVM classifier. Mediapipe is a real-time framework that extracts 21 ordered hand key points in 2.5 dimensions from a standard RGB image, independent of the hand scale. These points provide the x, y, and z positions relative to the hand's palm center. Mediapipe performs inference on the 21 key points with high prediction quality and real-time speed (Fig. 4.1e).

To train the SVM classifier to distinguish between gestures, the 21 hand landmarks were flattened into a single feature vector of size $63 = 21 \times 3$.

The classes representing the gestures are three: *(i)* an open hand for the *wait* command (Fig. 4.4a), *(ii)* a closed hand for the *follow* command (Fig. 4.4b), and *(iii)* all other configurations which are ignored (Fig. 4.4c).

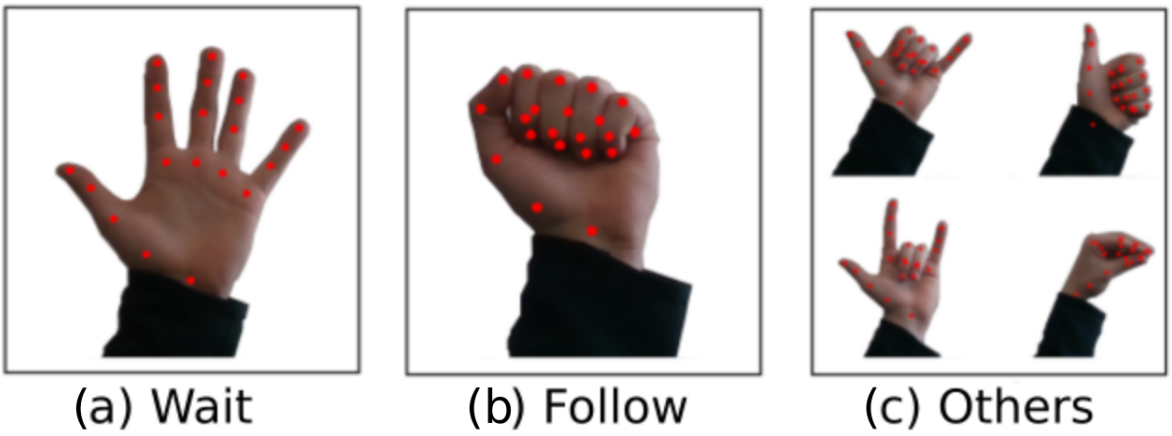


Figure 4.4: Classes considered to train the SVM for hand gesture detection. Mediapipe framework [157] is used to extract 3D key points (red points) relative to the centre of the hand. Reprinted, with permission, from [111], © 2023 IEEE.

A radial basis function kernel and a one-versus-one decision function were employed to train the multi-class SVM classifier. The classification output is filtered (*i.e.*, ξ equal consecutive classification outputs are required to send a command; the default choice is $\xi = 5$) and sent to the FSM, which processes it to command the robot's actions.

Navigation

The environments where humans and collaborative robots operate are typically noisy and populated. The robot must navigate freely while minimizing the probability of collision with obstacles to prevent damage. To accomplish this task, the ROS navigation stack[1] is utilized. Among the well-known advantages of using this navigation stack (*e.g.*, customization, structured organization), there is its generalization into a standard navigation architecture, which allows for simple re-adaptability across different robotic platforms.

For safety reasons, a circular safety region is established around the robot (see Fig. 4.3). If the estimated position of the target falls within this safety circle, signifying a dangerous area, the robot immediately deletes the last navigation goal and stops. Non-target people are treated as obstacles and are avoided if possible; otherwise, the robot stops and attempts to find an alternative path to the goal. In the worst-case scenario, where the target is not re-identified, and the obstacle avoidance module fails to recognize the person as an obstacle (a rare occurrence), the robot still has time to stop, owing to the KF expiration time, thereby nullifying the chances of collision.

[1]ROS Navigation Stack: `http://wiki.ros.org/navigation`

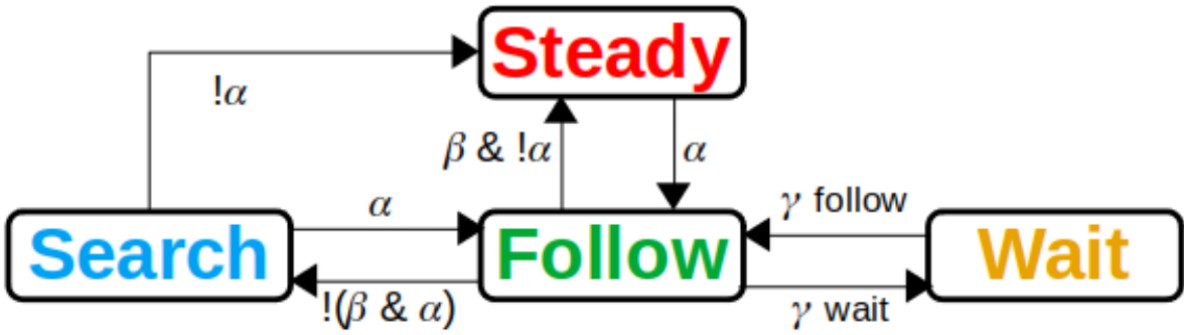


Figure 4.5: FollowMe FSM diagram. The transitions between the states are highlighted with Greek letters and boolean operators *and* (&), *not* (!). The transitions' meaning is: (α) the target is re-identified in the camera FOV, (β) the last target position is inside the safety circle, and (γ) the robot receives a command by the hand gestures module. Reprinted, with permission, from [111], © 2023 IEEE.

The safety distance d_{safe} can be computed as follows:

$$d_{safe} = 1.4\ v_{max}\ t_{exp} \tag{4.3}$$

where v_{max} is the maximum robot velocity and t_{exp} is the KF expiration time. In this way, the robot is designed to stop after traversing a distance of $v_{max} \times t_{exp}$ meters, leaving an additional 40% margin of the required stopping distance from the person to ensure safety.

Decision making

The application workflow is managed using a FSM, which collects all data from the perception and navigation modules and controls the robot's overall behavior.

The FSM has four defined states:

- *Steady*: The robot is stopped and waits for the perception module to issue a *follow* command.
- *Follow*: The robot actively tracks and follows the target person.
- *Search*: When the robot stops receiving target position updates, it initiates a search by rotating around its current position (for a maximum of one complete turn) toward the direction of the last known position.
- *Wait*: The robot was paused by the target person using the *wait* gesture command and remains stopped until the *follow* gesture command is received.

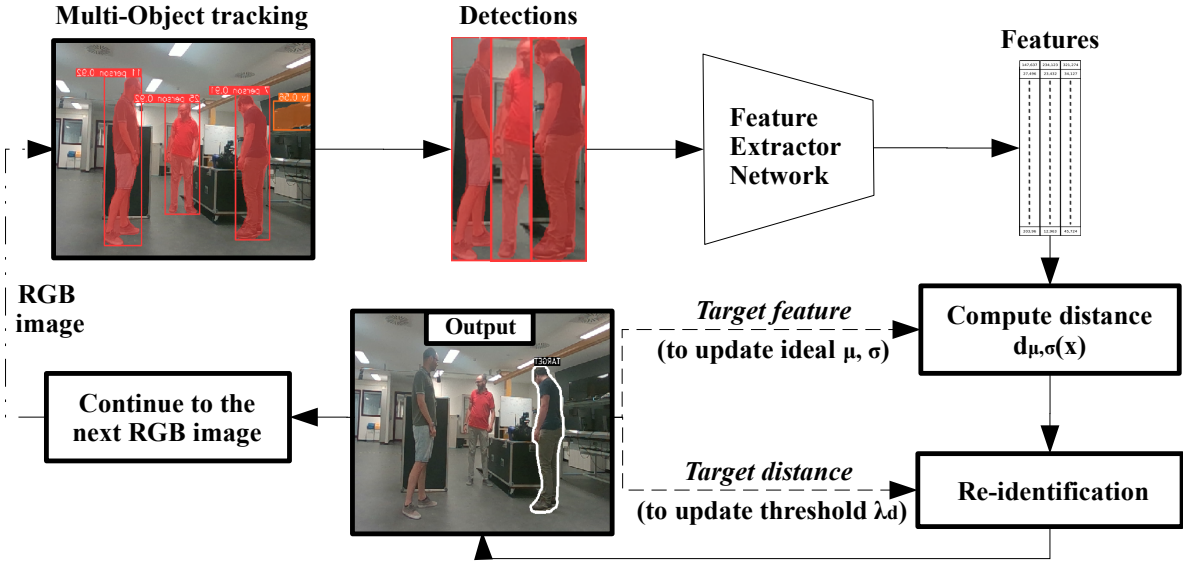


Figure 4.6: The pipeline framework begins with an image input and ends with a re-identified target output. The first module is the MOT, where a neural network provides a rough tracking of objects present in the image. From the MOT module, the detections obtained are passed into a feature extractor, which generates the output feature vectors $\boldsymbol{x}$. The feature vectors $\boldsymbol{x}$ allow to compute the statistical distance $d_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\mathbf{x})$ (as shown in Eq. (4.1)), which the Re-ID module will use. If the target is successfully re-identified, the target statistical distance $d_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\mathbf{x})$ and the target features $\boldsymbol{x}$ are used to update the re-identifier threshold λ_d and the ideal target representation $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ (represented by dashed lines). Reprinted, with permission, from [112], © 2024 IEEE.

The FSM's structure is visualized in Fig. 4.5. The events that trigger the transitions (represented by the edges) can be summarized as follows: (α) the target is re-identified within the camera's FOV, (β) the last target position is inside the safety circle, and (γ) the robot receives a command from the hand gestures module.

4.1.2 Introducing autonomous adaptation for smooth interaction in challenging situations

Although the Re-ID framework presented in Section 4.1.1 is sufficient for detecting and tracking a person in an image to perform a collaborative task, it requires initial calibration based on the current person's appearance. This presents a limitation when an individual is prone to changing their appearance frequently throughout the day or, certainly, over the course of a week. To overcome this issue, a new approach for

tracking people, presented in Fig. 4.6, is proposed. A pseudo-code implementation is also provided in Algorithm 1. The tracking process begins when a user selects an ID. Initially, the detected people's appearances, obtained using the MOT system, are collected in a database. This collection allows the user to choose the specific person they wish to follow.

Once the ID is selected, the RGB image is fed into the MOT algorithm to obtain a preliminary detection and tracking, which assigns a unique IDs to each person. These detections are used to crop the people images, which are then input into a deep neural network trained for person recognition (more details are available in Section 5.1.2). This network extracts features, represented as a one-column vector, which are subsequently used for Re-ID and tracking of the target.

For each frame, the application executes two sequential steps: Re-ID of the target and updating the target's ideal representation.

Re-identification

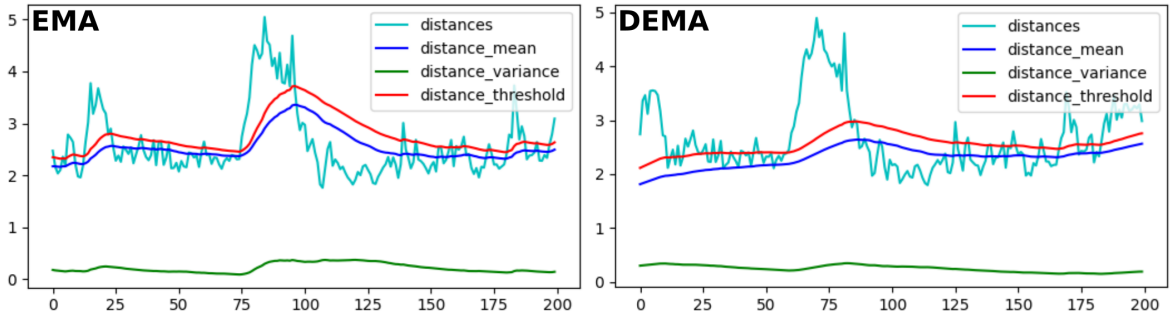
In the Re-ID step, the same statistical distance presented in Eq. (4.1) used in FollowMe is computed from the features $\mathbf{x}$ extracted for the detected people.

As a reminder and to better specify the components of Eq. (4.1), i denotes the i-th index of a vector. The values of mean and standard deviation, denoted by $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ respectively, represent the model of the ideal target to track. The mean $\boldsymbol{\mu}$ is initialized with the initial feature vector $\mathbf{x}$, while the standard deviation $\boldsymbol{\sigma}$ is initialized with a zero vector; both have dimension D.

In the next step, it is checked whether the tracking algorithm provides a detection with the same ID as the one selected by the user. If such a detection exists, it is output directly with its ID. If not, the algorithm proceeds to the Re-ID module. Here, the module compares the adaptive threshold λ_d with the feature vector that has the smallest distance from the target model. If the module is unable to re-identify any person among the detected ones, the framework continues by analyzing the next RGB frame. This step is similar to the one presented in Section 4.1.1, but includes a few improvements.

Model update

The primary contribution compared to the previous work lies in the model update step presented here. After the target is re-identified, its feature vector and corresponding statistical distance are used to continuously adjust the ideal target representation ($\boldsymbol{\mu}$



(a) EMA function effects. (b) DEMA function effects.

Figure 4.7: This figure illustrates a comparison between DEMA and EMA effects on threshold filtering and the ideal target representation. On the left, they follow the light blue line (*i.e.*, the statistical distance) with a small delay. On the right, the damping guarantees that their values do not track the distance behaviour when peaks are present. Instead, they slow down, waiting for the adaptation of the target model to the newly acquired appearances. Reprinted, with permission, from [112], © 2024 IEEE.

and $\boldsymbol{\sigma}$), as well as the threshold λ_d used during the Re-ID process, according to the following equation:

$$\lambda_d = \mu_d + 2\ \sigma_d \tag{4.4}$$

where μ_d and σ_d are the mean and the standard deviation of the target's statistical distances.

The variables $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, μ_d, and σ_d are updated online using a Damped version of the EMA ($\boldsymbol{\chi}^{dema}$), which is referred to as DEMA. The formula for DEMA is expressed as:

$$\boldsymbol{\chi}_t^{dema} = \alpha_{damp}\ \boldsymbol{\psi} + (1 - \alpha_{damp})\ \boldsymbol{\chi}_{t-1}^{dema}, \tag{4.5}$$

In this equation, $\boldsymbol{\psi}$ represents the new value at time t. The term α_{damp} is a weighting factor that defines the importance of the newly acquired value $\boldsymbol{\psi}$ relative to the DEMA $\boldsymbol{\chi}_{t-1}^{dema}$ from the previous time step. The calculation of α_{damp} depends on the specific implementation, following the formula:

$$\alpha_{damp} = \frac{2}{\mathrm{N}_{damp} + 1}, \tag{4.6}$$

To calculate the damping factor for the DEMA, the formula $\mathrm{N}_{damp} = \mathrm{N} \cdot \Delta$ is used, where N represents the number of DEMA updates, and Δ depends on the type of information being updated. It is optimal to initialize $\mathrm{N} = 0$ for faster initial convergence and to set a fixed upper-bound N_{max} to ensure that the DEMA remains affected by new values over time, since α_{damp} approaches 0 as N increases.

Two distinct damping factors are used for the DEMA. One is for the ideal target features, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, which is designated Δ_f (see Eq. (4.7)). The other is for the threshold λ_d, which is computed using μ_d and σ_d and is called Δ_{λ_d} (see Eq. (4.8)).

$$\Delta_f = min(1,\ \frac{d_{\boldsymbol{\mu},\boldsymbol{\sigma}}}{2}), \tag{4.7}$$

$$\Delta_{\lambda_d} = max(1,\ 2\frac{d_{\boldsymbol{\mu},\boldsymbol{\sigma}}}{\lambda_d}). \tag{4.8}$$

Fig. 4.7 compares the EMA with and without these damping factors. It demonstrates that adding the damping factor Δ_{λ_d} attenuates the high-frequency noise of the threshold λ_d (shown in red) and smoothes the entire line. This mechanism helps to prevent incorrect Re-IDs by reducing the λ_d peaks that typically appear in the standard EMA.

When the damping factor Δ_f from Eq. (4.7) is applied within DEMA Eq. (4.5) using N_{damp}, it helps mitigate situations where the distance in Eq. (4.1) from the current feature is minimal. Such small distances could otherwise lead to overfitting the ideal representation, particularly when the target is stationary in the same pose. The damping factor Δ_{λ_d} in Eq. (4.8) is utilized to prevent the threshold from increasing excessively when the ratio between the distance $d_{\boldsymbol{\mu},\boldsymbol{\sigma}}$ and the threshold λ_d is too large. This actively helps prevent incorrect Re-ID.

To enhance the algorithm's reliability, a *blacklist manager* is introduced. This manager evaluates the IDs provided by the MOT and identifies which ones belong to distractors. By doing so, these distractor IDs are excluded during the Re-ID step. Whenever the target ID provided by the MOT remains consistent during tracking, the manager adds the IDs of the other individuals in the image (the distractors) to the blacklist. This approach minimizes Re-ID errors that might occur in challenging situations.

4.1.3 Using unsupervised continual learning and parallel twin neural network to improve robustness online

The CARPE-ID framework [112] described in Section 4.1.2 is a robust approach for tracking people in images that can adapt to moderate changes in appearance. However, it is prone to a phenomenon known as catastrophic forgetting, meaning it tends to lose the memory of previously seen target appearances as it continually updates its model to accommodate new ones. This limitation is particularly challenging when the target person frequently changes their appearance.

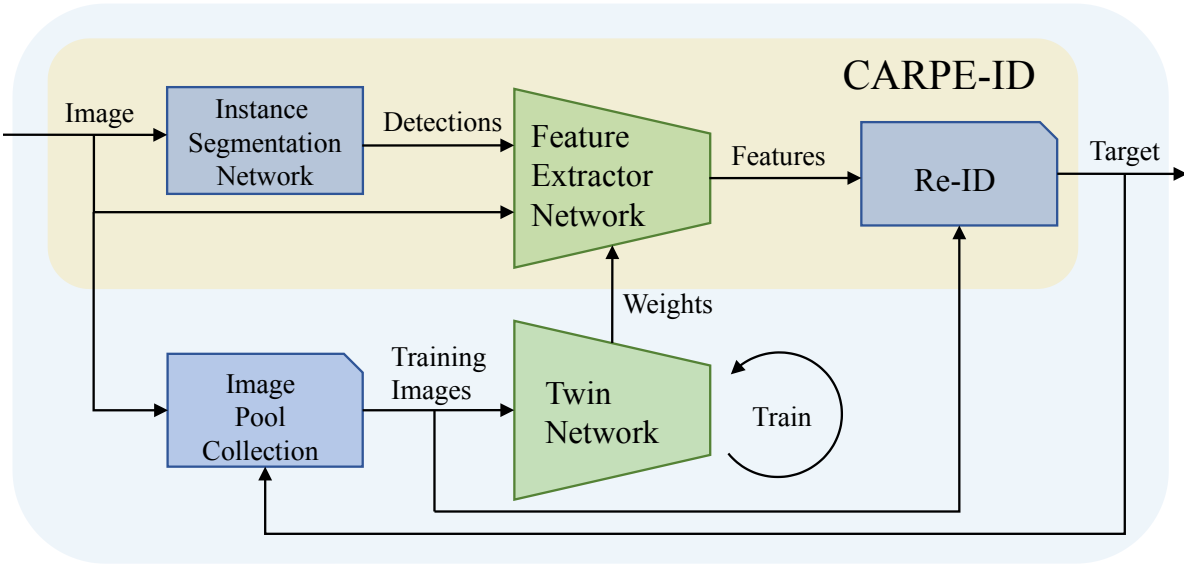


Figure 4.8: This work pipeline, integrated with the CARPE-ID framework, outlines the algorithmic flow from the input image to the Re-ID output. The arrows symbolize the data exchange among the various components within the framework. Reprinted, with permission, from [110], © 2025 IEEE.

To address this issue, an autonomous continual learning module is presented, aimed at fine-tuning the feature extraction network to extract features that are less reliant on the individuals' frequently changing appearances [110]. This continuously trained network specializes in recognizing the target by focusing on characteristics that are more consistent and unique to that person, making them distinguishable from the other actors present in the scene, known as distractors. The module autonomously acquires target and distractor images and performs parallel training once a sufficient number of samples is collected, thereby avoiding any delays in the tracking performance. A qualitative explanation of this concept is provided in Section 5.1.3, where saliency maps, an explainability technique commonly used in computer vision, are employed to support the presented hypothesis.

This section is divided into several subsections to describe and validate the contributions of this work. It begins with the presentation of the idea of parallel training, along with considerations regarding its implementation. The explanation of the continual training setup and the smart image pool acquisition process is then presented. Finally, the features of post-training statistical model adaptation and early training are reported. These two modules play a central role in improving the overall accuracy of this framework.

Parallel network

An essential requirement for online tracking is achieving optimal performance in terms of both tracking time and accuracy. It is crucial to maintain continuous detection at a high frame rate without losing the target. Conversely, implementing an online continual learning approach necessitates training the feature extractor neural network online. This training process can become impractical when the feature extractor network is rendered unavailable for inference while training [87]. To address this challenge, a twin neural network structure is introduced, as illustrated in Fig. 4.8. This arrangement enables the concurrent training of the twin network while the principal network continues to execute the main tracking task. Once the twin network's training is completed, its new weights can be substituted into the principal network, thereby minimizing any delays in the tracking task.

Duplicating a neural network on GPUs may impose a considerable computational burden. While feature extractor networks are generally lightweight, duplicating them can be difficult to handle in systems with standard computational capabilities, such as mobile robots, even though the advantages for the tracking task are clear. On the other hand, relying on a single network would result in losing track of the target during network training, especially if the target becomes occluded during this process, making subsequent re-recognition impossible.

Training setup

The people image buffer, obtained from the CARPE-ID framework, consists of both target images and distractor images. In this context, a distractor refers to any person other than the target individual. This buffer is subdivided into N sets. Each set has a dimension that matches the batch size, N, and contains $\frac{\mathrm{N}}{2}$ target and $\frac{\mathrm{N}}{2}$ distractor images. For each batch, a training iteration of the neural network is performed using the *Soft Triplet loss* proposed by Qian et al. [99], which is the standard Triplet loss function [120] in Eq. (4.9) embedded within the Log Softmax loss in Eq. 4.10.

$$\mathcal{L}_{\mathcal{T}} = \sum_{i=1}^{\mathrm{N}} \left[\|\mathbf{x}_i^a - \mathbf{x}_i^p\|_2^2 - \|\mathbf{x}_i^a - \mathbf{x}_i^n\|_2^2 + \alpha \right], \tag{4.9}$$

$$\mathcal{L}_{\mathcal{S}} = -\log \frac{\exp \mathbf{x}_i}{\sum_{j=1}^{K} \exp \mathbf{x}_j} \tag{4.10}$$

where α is a bias weight. This combined loss function helps the model in discerning the similarities and differences between distinct people. It operates by considering three

types of samples:

1. *Anchor* samples $\mathbf{x}_i^a$ representing target image features;
2. *Positive* samples $\mathbf{x}_i^p$ corresponding to other image features belonging to the same target;
3. *Negative* samples $\mathbf{x}_i^n$, denoting distractor image features.

In practice, the method computes the Euclidean distances between all the embeddings within the batch, and for each positive and negative sample, the hard choices are maintained. Specifically, for each anchor, the farthest positive sample and the nearest negative sample are considered to obtain stronger feature embeddings. The Log Softmax is then computed for each resulting value, along with their mean weighted by a predefined margin α. By using these samples, the network is trained so that the anchor sample is closer to the positive sample and further away from the negative one. This approach enables the network to learn how to distinguish between similar and dissimilar targets more effectively.

Image pool collection

In contemporary times, it is evident that datasets play a key role in the effectiveness of neural network training. If a dataset is poorly structured, it can adversely affect network performance. This holds not only in the context of continual learning but is further accentuated by the fact that datasets acquired in real-time have lower dimensions. Additionally, since such training data is collected online, there may not be substantial variation among the samples. Insufficient diversification among dataset samples can lead to overfitting in favor of the most recent inputs, thereby aggravating the problem of catastrophic forgetting [87].

To address this issue of improper network specialization, an intelligent image pool collection procedure is implemented that leverages the statistical model of CARPE-ID to merge the latest acquired images with a set of older images to form the dataset. However, it is not feasible to retain all old images indefinitely, as this would lead to an unmanageable increase in dataset size, potentially causing costly training and GPU overflow.

To overcome this challenge, an image selection mechanism is implemented that retains those images in the dataset that significantly differ in appearance from the current statistical model, as depicted in Fig. 4.9. To achieve this, the features extracted

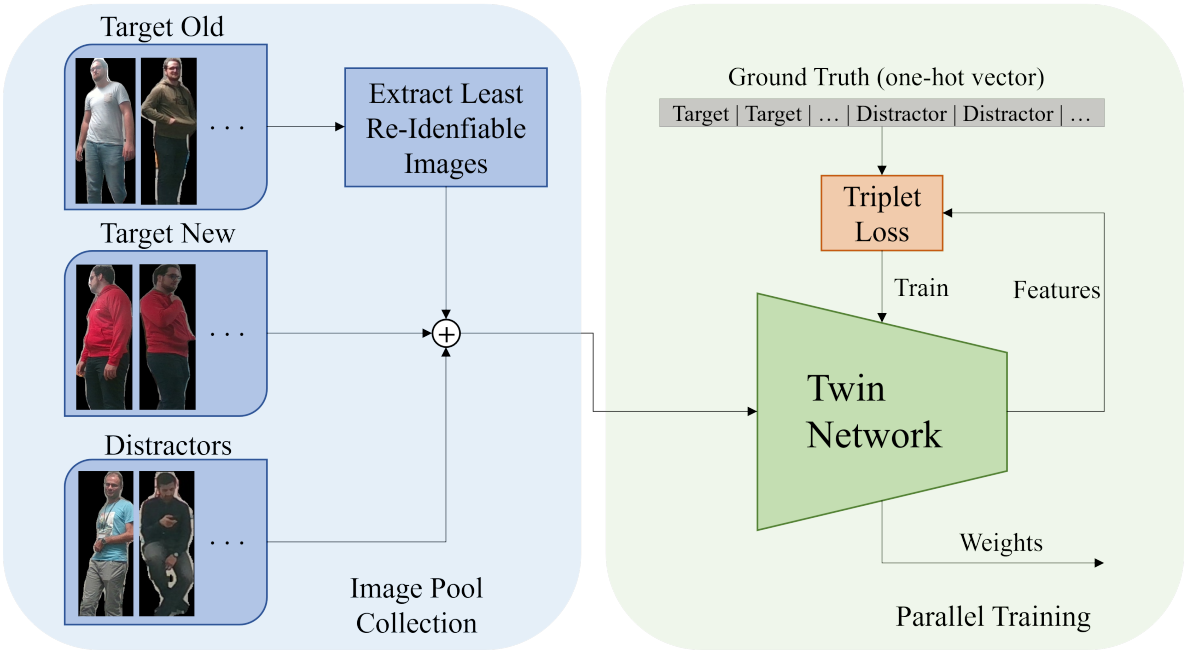


Figure 4.9: On the left, the pool collection of training images for adapting the twin network to target appearances, and on the right, the training process with Soft Triplet loss computation, is shown. Reprinted, with permission, from [110], © 2025 IEEE.

from the older target images and the statistical distance formula described in Eq. (4.1) are utilized to assess the similarity between each image embedding and the current model.

Next, the older images are arranged based on their calculated distances, sorting them in ascending order. The most recently acquired target images are then merged with those whose visual appearance differs the most from the model, provided the images still belong to the target of interest. Once this process is completed, the target image buffer is shuffled and added to the concurrently acquired distractor image buffer. The final combined image buffer is subsequently fed into the twin neural network for training. The size of the image buffer to be acquired is determined by the product of the selected batch size and the training iteration parameters set during initialization.

Post-training statistical model adaptation

The feature vectors computed using the new network weights and the images used during the training described in the previous sections are subsequently used to refine the features and threshold statistical models. To ensure the compatibility of the new neural network weights with the statistical appearance model of CARPE-ID, it is essential to

update the model accordingly, as the introduction of these new weights can lead to changes in the neural network's output features that are no longer aligned with the previously acquired model. This adaptation approach allows the proposed framework to rapidly adjust to structural changes in the feature space, preventing misclassification and tracking losses (in this work, this variant is referred to as *CARPE-ID statistical model update*).

Early training

While evaluating this framework, another challenge was encountered: cases where the tracked target moves out of the camera's FOV before the image buffer is sufficiently filled. This issue particularly occurs when the batch size and training iterations are set to high values. Under these circumstances, the framework may fail to start a single training iteration, consequently failing to incorporate the latest target appearances, which are essential for the Re-ID phase when the target re-enters the camera's FOV.

To address this issue, a solution that monitors the duration of target loss is introduced. If this period exceeds a predefined threshold, an early training session is initiated using the images currently available in the buffer. This approach ensures the framework can quickly overcome such challenging scenarios while maintaining tracking effectiveness.

4.2 Environmental perception through geometrical matching

As stated in Section 3.2.1, robots navigating in unknown environments require a robust and accurate SLAM algorithm to build a map of the surroundings and localize themselves within it. Section 4.2.1 presents LEO-SLAM, a novel LiDAR-based SLAM algorithm designed to address the challenges of real-time mapping and localization in large-scale and dynamic environments. The proposed approach combines multi-level scan alignment with submap-based LCD (Loop Closure Detection) to achieve accurate and efficient SLAM performance. In Section 4.2.2, a novel point cloud pre-processing technique is introduced that enhances the quality of the input data, thereby improving scan alignment and LCD. Finally, in Section 4.2.3, an improved LCD method based on SoTA algorithms is proposed, which enhances the robustness of the overall SLAM system in noisy environments.

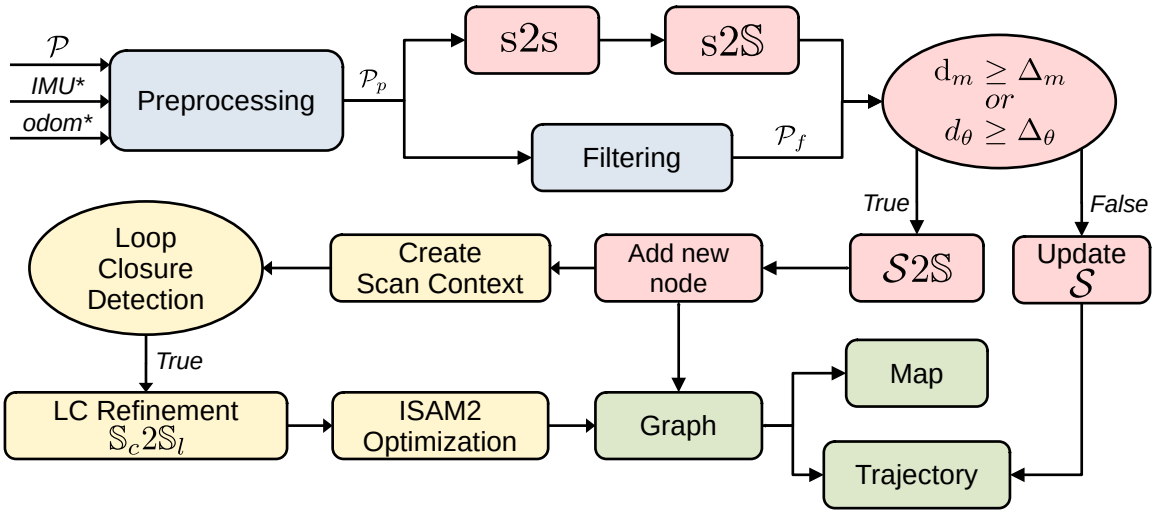


Figure 4.10: LEO-SLAM system architecture. The algorithm receives as input a point cloud $\mathcal{P}$ and optionally external odometry or IMU data. The point cloud is preprocessed, and the output $\mathcal{P}_p$ is passed in parallel into the scan-to-scan (s2s) and scan-to-submaps (s2$\mathbb{S}$) alignment modules, as well as into a filtering module, which produces $\mathcal{P}_f$, later used for building the final map. If the robot has travelled a distance d_m greater than Δ_m, or rotated by an angle d_θ greater than or equal to Δ_θ, a submap-to-submaps ($\mathcal{S}$2$\mathbb{S}$) alignment is performed before adding a new node to the graph. If these distance and orientation conditions are not met, the current submap $\mathcal{S}$ and the trajectory are updated. When a new node is created, its submap $\mathcal{S}$ is used to compute a SC++, and a LCD step is performed. If a loop is detected, a submap-to-submaps ($\mathcal{S}$2$\mathbb{S}$) alignment is executed between the current node's submap $\mathcal{S}$ and the submaps around the detected loop-closure node $\mathbb{S}$. If the alignment converges, the loop is added to the pose graph, which is then optimized. The output of this optimization is used to correct both the trajectory and, consequently, the map. Reprinted, with permission, from [107], © 2025 IEEE.

4.2.1 Submap-based 3D LiDAR SLAM with multi-level scan matching

This section presents the LEO-SLAM algorithm, beginning with the system overview illustrated in Fig. 4.10. This figure depicts the SLAM workflow, with different colors representing the key conceptual blocks of the pipeline. Point cloud pre-processing and filtering are highlighted in light blue. The multi-level scan alignment strategy is shown in pink. Submap-based LCD and pose graph optimization blocks are indicated in yellow. Finally, the algorithm outputs, namely the map and the reconstructed trajectory, are represented in green.

Algorithm 2 provides a simplified pseudocode representation of a single LEO-SLAM loop, which is the processing of an incoming point cloud $\mathcal{P}$ and, optionally, external

odometry information ${}^{\mathrm{o}}\mathbf{H}_{\mathrm{b}}$. It outputs ${}^{\mathrm{m}}\mathbf{H}_{\mathrm{o}}$, the homogeneous transformation from the map frame m to the odometry frame o, which corrects the robot's pose. If no external odometry ${}^{\mathrm{o}}\mathbf{H}_{\mathrm{b}}$ is provided, the odometry frame o and the robot base frame b coincide, resulting in ${}^{\mathrm{m}}\mathbf{H}_{\mathrm{o}} = {}^{\mathrm{m}}\mathbf{H}_{\mathrm{b}}$. The function output directly represents the robot's pose relative to the initial map frame m.

This SLAM system is based on the concept of a node n, which consists of a submap $\mathcal{S}$, an accumulation of point clouds, and a pose ${}^{\mathrm{m}}\mathbf{H}_{\mathrm{n}}$ relative to the fixed map frame m.

Point cloud preprocessing

The input point cloud $\mathcal{P}$ undergoes two main steps: the *preprocessing* and the *filtering*, as illustrated in Fig. 4.10. In the *preprocessing* stage, a set of 3D crop box filters is applied to remove points associated with the robot itself and those located too far from the LiDAR, as distant points tend to be sparse and less useful for accurate mapping. Following this, a statistical outlier-rejection filter removes spurious points caused by motion, thereby improving the overall quality of the point cloud. These preprocessing operations ensure a balance between accuracy and computational efficiency, which is particularly crucial when dealing with sparse point clouds in the Multi-Level Alignment Strategy discussed in the next section. Additionally, the preprocessing block can incorporate external odometry or IMU data to provide an initial pose estimate for ICP alignment.

Once this initial step is completed, the preprocessed point cloud $\mathcal{P}_p$ is forwarded to both the Multi-Level Alignment layer and the subsequent *filtering* stage. In this second stage, a ground-aware intensity filter is employed to discard points exceeding a specified intensity threshold, such as those from reflections on glass surfaces, while retaining low-intensity ground points. To further refine the data, a voxelization filter is applied to regulate the point cloud's density, creating a more uniform distribution while preserving the environment's structure. While $\mathcal{P}_p$ serves as input to scan matching alignment, the final filtered point cloud $\mathcal{P}_f$ contributes to constructing a coherent map of the environment. This refined representation enhances visualization and facilitates navigation, ensuring that the generated map accurately reflects the real-world surroundings.

Multi-level alignment strategy

At each LiDAR scan reception, the preprocessed point cloud $\mathcal{P}_p$ is first used to compute the robot's displacement. It is then transformed into the map frame m and accumulated

within a submap $\mathcal{S}$. The scan collection continues until the robot's travelled distance d_m from the last created node $\mathrm{n}_{\mathrm{c}-1}$ exceeds a predefined Euclidean threshold Δ_m, or its relative orientation change d_θ surpasses a set threshold Δ_θ. Once either of these conditions is met, a new node n_{c} is created, containing both the collected submap and the robot's current pose, which serves as the submap keyframe. This node is then added to a pose graph, which is structured as a factor graph that includes only pose factors.

To mitigate odometry drift, which commonly occurs in scan matching algorithms, a Multi-Level Alignment Strategy composed of three hierarchical alignment steps is employed:

1. *Scan-To-Scan* s2s;
2. *Scan-To-Submaps* s2$\mathbb{S}$;
3. *Submap-To-Submaps* $\mathcal{S}$2$\mathbb{S}$.

where s represents a single point cloud scan, $\mathcal{S}$ denotes a submap composed of multiple consecutive scans, and $\mathbb{S}$ is a subset of the N_s previously created submaps. Each level of alignment refines localization accuracy, ensuring both local and global consistency in the constructed map.

Alignment is performed at these three levels by exploiting the GICP algorithm [121], specifically the small-GICP implementation [58]. The s2s alignment corrects small drifts by aligning the current LiDAR scan $\mathcal{P}_t$ with the previous one $\mathcal{P}_{t-1}$. The resulting transformation serves as an initial guess for the s2$\mathbb{S}$ alignment, where the scan is aligned with a collection of the N_s previous submaps to ensure local consistency.

Approaches such as DLO [18] use keyframes as their nodes, *i.e.*, a structure composed of a single point cloud and its associated pose. They utilize the N_s nearest keyframes in Euclidean space, summed with those extracted on the convex and concave hulls of the stored trajectory, to build the submap. In this way, they exploit keyframes from the entire trajectory rather than just the previous consecutive ones, as in the presented case. While this approach benefits robot pose refinement in an odometry system, it is less suitable for long trajectories and final map construction, as it may introduce discontinuities when the robot revisits a previously seen area, thereby skewing the final output map and trajectory.

Once a new node is ready, a final $\mathcal{S}$2$\mathbb{S}$ alignment is performed between the node submap $\mathcal{S}$ and $\mathbb{S}$, which is the subset of the N_s previously created submaps. This step improves long-term accuracy and reduces drift accumulation. Finally, after the multi-level alignment concludes, the submap keyframe is fixed, the node is finalized, and it

is added to the pose graph. The node is then analyzed using SC++ for submap-based LCD, as presented in the next section.

Loop closure with submap-based Scan Context

When a new node n_c is finalized, its associated submap $\mathcal{S}$ point cloud is used to compute a SC, as presented by Kim et al. [54]. The SC is a 2D descriptor that partitions the point cloud into rings and sectors, recording the maximal point height within each cell. This LCD technique performs a global search against all previously generated SCs. It utilizes a Ring Key (a 1D vector encoding ring values) to execute a preliminary, optimized check to determine if two SCs are sufficiently similar to warrant a full comparison. Furthermore, a Sector Key embeds sector values into a vector, similar to the Ring Key, enabling the generated SC++ image to be circularly shifted to achieve orientation invariance.

Although this technique has proven effective in outdoor environments, it exhibits several limitations, particularly in indoor settings. The first limitation stems from the low channel count typical of mobile robot LiDAR sensors (typically 16 channels) compared to the 64 or more channels commonly used in autonomous vehicles. To address this, a submap-based SC++ is introduced. Since each submap is an aggregate of multiple point clouds, it contains more environmental information, thereby yielding descriptors that resemble scans obtained from high-ring-count LiDAR sensors.

Another limitation is that SCs records the maximum point height for each entry, which is suboptimal indoors due to the presence of ceilings. To mitigate this, the ceiling is explicitly removed from the submap point cloud in indoor environments. This prevents the SC++ from becoming overly uniform and ignoring crucial environmental details. For example, in a corridor with obstacles, the maximum height would be uniformly defined by the ceiling, thus eliminating valuable height variations caused by the obstacles or the corridor structure itself.

Finally, structured environments often contain visually similar locations, leading to potential incorrect matches, even when spatially distant, as SC++ performs a global search across all previously acquired contexts. To address this, a search area ellipsoid $\mathbf{a_s}$ is introduced. This area expands as the robot moves, growing proportionally to the motion uncertainty, which can be estimated from the robot's motion or LiDAR odometry.

For each node n_c added to the graph, the ellipsoid is centered at the node's pose ${}^{m}\mathbf{H}_{n_c}$ and defines which SCs fall within its area $\mathbf{a_s}$. SC++ LCD is then performed

only on the variable-dimension node set $\mathbb{K}$, which consists of k nodes whose distance vector $\mathbf{d_j}$, with $j = 1, \ldots, k$, in Eq. (4.11), falls within the ellipsoid search area $\mathbf{a_s}$, *i.e.*, satisfying Eq. (4.12).

$$\mathbf{d_j} \;=\; {}^{\mathrm{n_c}}\mathbf{t}_{\mathrm{n_j}} \;=\; {}^{\mathrm{m}}\mathbf{t}_{\mathrm{n_c}} \;-\; {}^{\mathrm{m}}\mathbf{t}_{\mathrm{n_j}}, \tag{4.11}$$

$$\left(\frac{\mathbf{d_j^x}}{\mathbf{a_s^x}}\right)^2 + \left(\frac{\mathbf{d_j^y}}{\mathbf{a_s^y}}\right)^2 + \left(\frac{\mathbf{d_j^z}}{\mathbf{a_s^z}}\right)^2 \leq 1, \tag{4.12}$$

where x, y, and z are the components of vectors $\mathbf{d_j}$ and $\mathbf{a_s}$.

If a loop is detected between $\mathrm{n_c}$ and $\mathrm{n_l}$, the translation vector $\mathbf{d_j}$ with $j = l$ in Eq. (4.11) is used as the initial guess for the GICP algorithm. Two point cloud sets, $\mathbb{S}_c$ and $\mathbb{S}_l$, are created for the current node $\mathrm{n_c}$ and the loop node $\mathrm{n_l}$, respectively. Each set contains the nodes in the neighborhood of the associated node. Specifically, $\mathbb{S}_c$ contains the v nodes preceding the current node, while $\mathbb{S}_l$ contains the v nodes preceding and the v nodes succeeding the loop node $\mathrm{n_l}$, where $v \in \mathbb{N}$ is a predefined positive integer. The GICP algorithm is then computed to align the $\mathbb{S}_c$ point clouds with the $\mathbb{S}_l$ point clouds, using $\mathbf{d_l} \;=\; {}^{\mathrm{n_c}}\mathbf{t}_{\mathrm{n_l}}$ as the initial guess. This $\mathbb{S}_c 2 \mathbb{S}_l$ alignment is employed to improve loop closure accuracy and mitigate false alignments in low-texture environments.

If the alignment converges, the GICP homogeneous transformation output $\mathbf{H}_{lc}^{ICP}$ is used to correct the current node pose ${}^{\mathrm{m}}\mathbf{H}_{\mathrm{n_c}}$, as expressed in Eq. (4.13):

$${}^{\mathrm{m}}\mathbf{H}'_{\mathrm{n_c}} = \mathbf{H}_{lc}^{ICP} \;*\; {}^{\mathrm{m}}\mathbf{H}_{\mathrm{n_c}}, \tag{4.13}$$

where ${}^{\mathrm{m}}\mathbf{H}'_{\mathrm{n_c}}$ is the corrected pose of node $\mathrm{n_c}$. The estimated final pose between the two nodes in Eq. (4.14) is then used to close the loop, adding the corrected pose factor ${}^{\mathrm{n_c}}\mathbf{H}'_{\mathrm{n_j}}$ between node $\mathrm{n_c}$ and node $\mathrm{n_j}$:

$${}^{\mathrm{n_c}}\mathbf{H}'_{\mathrm{n_j}} \;=\; {}^{\mathrm{m}}\mathbf{H}'^{-1}_{\mathrm{n_c}} \;*\; {}^{\mathrm{m}}\mathbf{H}_{\mathrm{n_j}}. \tag{4.14}$$

When the loop is correctly closed between two non-consecutive nodes, the pose graph is optimized using iSAM2 [51], and the resulting new node poses are used to refine the nodes' keyframes.

4.2.2 Point cloud ground-aware intensity filtering

As stated in Section 4.2.1, point cloud filtering is a crucial step in the LEO-SLAM pipeline, as it enhances the quality of the input data, leading to improved scan alignment and LCD. In this section, a novel ground-aware intensity filter is presented that

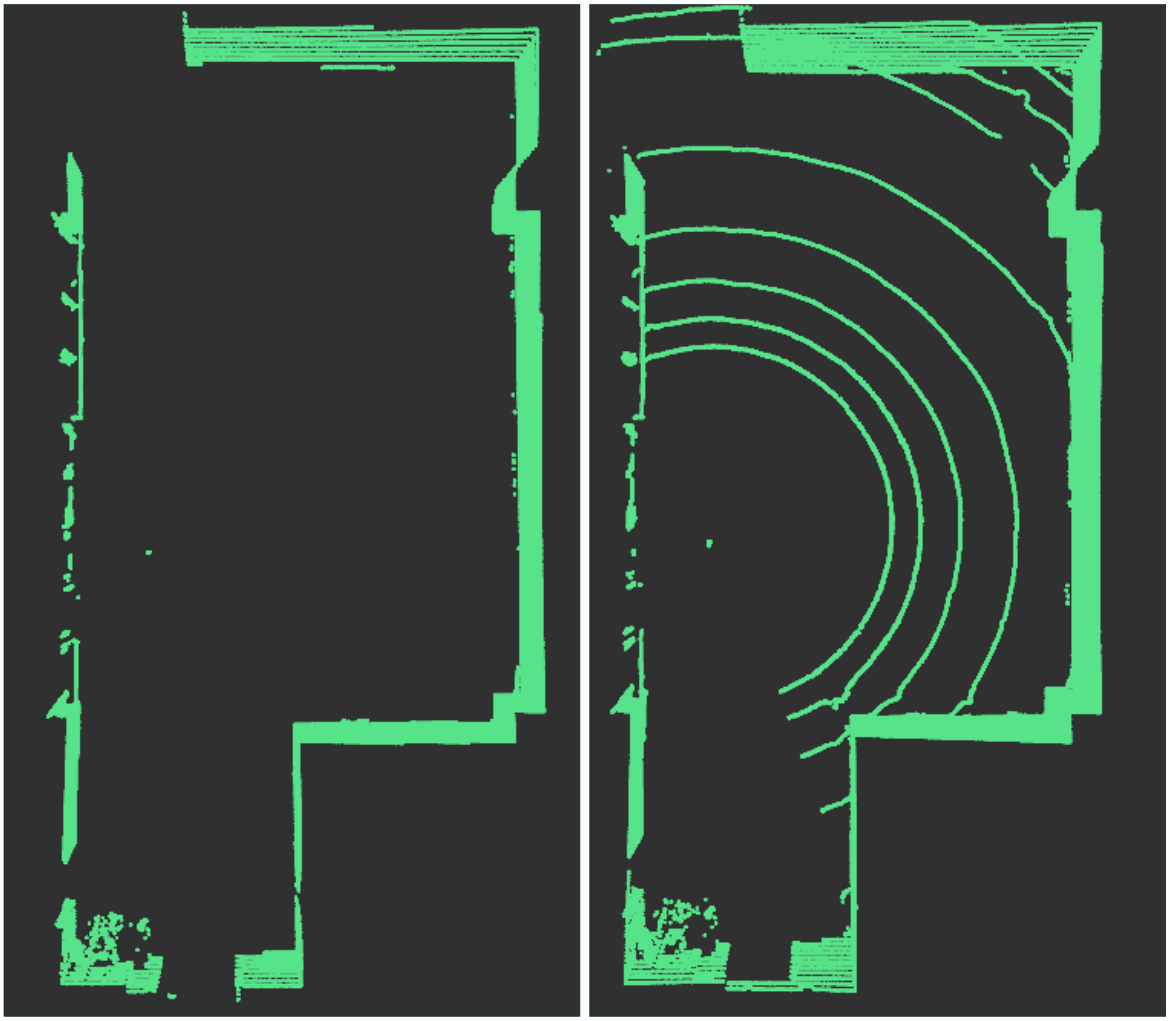


Figure 4.11: Comparison of intensity filtering: on the left, using only intensity thresholding; on the right, incorporating the ground awareness into the filtering process.

effectively removes points with intensities outside a specified range while preserving low-intensity ground points. Two lightweight implementations of this filter are proposed: the first, named the *naive intensity filter*, is simpler and faster; while the second implementation, referred to as the *normal intensity filter*, is more complex.

Both filters iterate through all the points ${}^{l}\mathbf{p}_i$ of the point cloud ${}^{l}\mathcal{P} = \{{}^{l}\mathbf{p}_1, {}^{l}\mathbf{p}_2, \ldots, {}^{l}\mathbf{p}_n\}$, expressed in the LiDAR frame l, and check whether their intensity ψ_i falls outside the specified boundary $\psi^* = [\psi^*_{min}, \psi^*_{max}]$. This condition defines the intensity range to be filtered out, as expressed in Eq. (4.15):

$$\psi_i \notin \psi^* \equiv (\psi_i < \psi^*_{min}) \;\&\; (\psi_i > \psi^*_{max}) \tag{4.15}$$

However, points belonging to the ground often return low reflected energy due to the low incidence angle θ between the LiDAR ray and the ground surface. These ground points are typically filtered out by a standard thresholding filter, as seen in the left image of Fig. 4.11, which can cause problems for LiDAR odometry and SLAM systems. Fig. 4.11 demonstrates the benefits of incorporating ground awareness through a comparison between the standard intensity thresholding filter and one enhanced with ground awareness.

Naive intensity filter

As the name suggests, the first implementation, also known as the *naive intensity filter*, is a straightforward application of a bound filter with basic ground awareness.

To address the issue of ground filtering, all points located below the robot's footprint, *i.e.*, the projection of the robot's base link onto the ground, are retained. This approach ensures that ground points, which provide valuable environmental information, are preserved. However, it also retains incorrect reflections detected below the robot's height. While such false detections could be considered minimal and insignificant compared to the other points being filtered out, it is important to acknowledge their presence.

Considering ${}^{\mathrm{l}}\mathbf{t}_{\mathrm{f}}$ as the translation vector between the LiDAR frame l and the footprint frame f, the third component, ${}^{\mathrm{l}}z_{\mathrm{f}}$, translates from the LiDAR frame l to the footprint frame f along the z-axis. This component can be used to check whether the points ${}^{\mathrm{l}}\mathbf{p}_i$ in the LiDAR frame l lie below the footprint frame f.

Note that for fixed-wheeled robots, ${}^{\mathrm{l}}\mathbf{t}_{\mathrm{f}}$ is static and does not change during navigation. In this experimental setup, however, a quadruped robot is used, where ${}^{\mathrm{l}}\mathbf{t}_{\mathrm{f}}$ can be computed using the positions of the feet, ${}^{\mathrm{l}}\mathbf{t}_{\mathrm{f_i}}$, with $\mathrm{f_i}$ representing the pose of the feet in contact with the ground. Another important consideration is that the LiDAR xy-plane is assumed to be parallel to the base footprint xy-plane, meaning that only LiDAR-to-footprint translations are necessary. If this assumption is not valid, such as when the quadruped robot is tilted, or the LiDAR is positioned with pitch or roll angles, fthe full homogeneous transformation ${}^{\mathrm{l}}\mathbf{H}_{\mathrm{f}}$ should be used to roto-translate the LiDAR points into the footprint frame before checking their height. The details of these platform-dependent factors are beyond the scope of this paper and are left to the reader.

Taking into account the above considerations and assumptions, the final *naive in-*

tensity filter equation is:

$$ {}^{\mathrm{l}}\mathcal{P}^{*} = \{ {}^{\mathrm{l}}\mathbf{p}_i \ \in \ {}^{\mathrm{l}}\mathcal{P} \mid (\psi_i \notin \psi^{*}) \ \& \ ({}^{\mathrm{l}}p_i^{z} < \ {}^{\mathrm{l}}z_{\mathrm{f}}) \} \tag{4.16} $$

where ${}^{\mathrm{l}}\mathcal{P}^{*}$ is the ground-aware intensity filtered point cloud, ${}^{\mathrm{l}}p_i^{z}$ is the z component of point ${}^{\mathrm{l}}\mathbf{p}_i$, "|" and "&" denote "*such that*" and "*conditional and*" respectively, and $\psi_i \notin \psi^{*}$ is defined in Eq. (4.15).

Normal intensity filter

The second proposed solution is more sophisticated than the previous one; however, its benefits do not necessarily outweigh the issues it introduces. Similar to the naive approach, the normal filter checks whether the point intensity ${}^{\mathrm{l}}\mathbf{p}_i^{\psi}$ lies outside the boundaries $\psi^{*} = [\psi^{*}_{min}, \psi^{*}_{max}]$, and whether ${}^{\mathrm{l}}\mathbf{p}_i$ belongs to the ground. However, the ground check is now more advanced, involving the computation of the normal vector ${}^{\mathrm{l}}\boldsymbol{\eta}_i$ for each cloud point ${}^{\mathrm{l}}\mathbf{p}_i$.

The *Normal Intensity Filter* consists of two primary steps:

1. Compute the normal vectors ${}^{\mathrm{l}}\boldsymbol{\eta}_i$ for the points ${}^{\mathrm{l}}\mathbf{p}_i$.

2. Check if the point's normal vector ${}^{\mathrm{l}}\boldsymbol{\eta}_i$ is parallel to the z-axis of the robot's footprint frame f.

In the norm computation step, the normal vectors of the points are calculated using the technique presented by Rusu et al. [117], with the implementation provided by the Point Cloud Library (PCL)[2].

To check whether a normal vector ${}^{\mathrm{l}}\boldsymbol{\eta}_i$ associated with a point ${}^{\mathrm{l}}\mathbf{p}_i$ is parallel to the z-axis of the robot's footprint frame f, the vector must first be roto-translated into the footprint frame using the following transformation:

$$ {}^{\mathrm{f}}\boldsymbol{\eta}_i = \ {}^{\mathrm{f}}\mathbf{H}_l \ {}^{\mathrm{l}}\boldsymbol{\eta}_i \tag{4.17} $$

where ${}^{\mathrm{f}}\mathbf{H}_{\mathrm{l}}$ is the homogeneous transformation between the LiDAR and the footprint frame, and ${}^{\mathrm{f}}\boldsymbol{\eta}_i$ is the normal vector expressed in the footprint frame.

Once the normal vector is expressed in the footprint frame, the angle between the vector ${}^{\mathrm{f}}\boldsymbol{\eta}_i$ and the z-axis of the f frame, defined as ${}^{\mathrm{f}}\mathbf{z} = [\mathbf{0}, \ \mathbf{0}, \ \mathbf{1}]^{\mathbf{T}}$, can be calculated.

[2]PCL library normal estimation: `https://pointclouds.org/documentation/tutorials/normal_estimation.html`

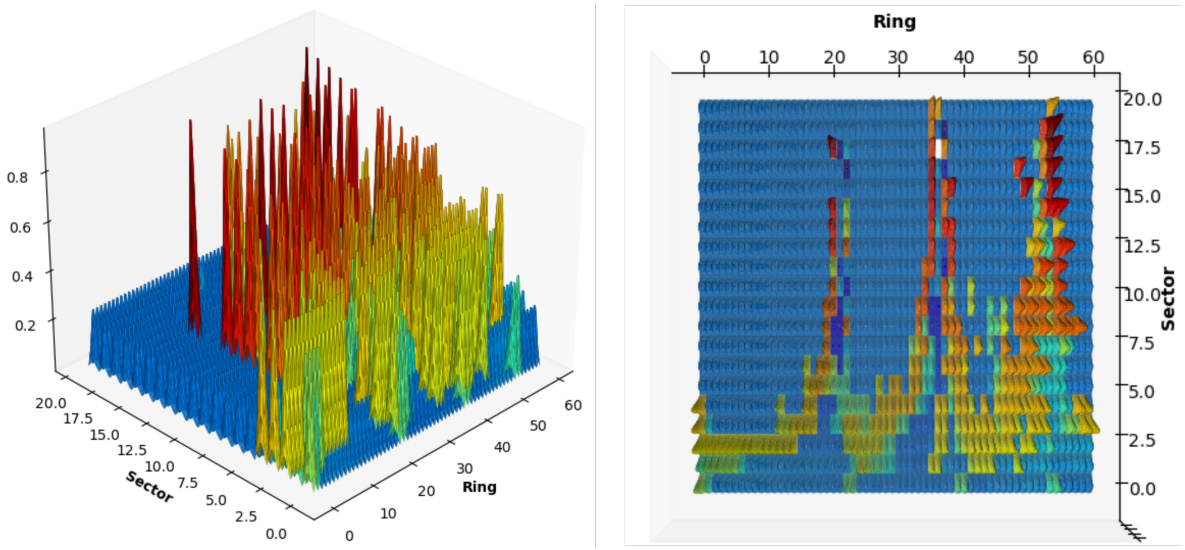


Figure 4.12: The diagonal and top views of a GSC, where each entry represents a multivariate normal distribution, are shown on the left and right, respectively. Each Gaussian distribution contributes to the computation of the GSC matrix.

To compute this angle, the vector cosine similarity measure, θ_{η_i}, is used, which is given by:

$$\theta_{\eta_i} = \frac{\langle {}^{\mathrm{f}}\boldsymbol{\eta}_i, {}^{\mathrm{f}}\mathbf{z} \rangle}{\|{}^{\mathrm{f}}\boldsymbol{\eta}_i\|_2 \|{}^{\mathrm{f}}\mathbf{z}\|_2} \tag{4.18}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product , and $\|\boldsymbol{x}\|_2$ represents the vector L_2-norm.

Using the cosine similarity θ_{η_i} from Eq. (4.18), it can be estimated whether the point ${}^{\mathrm{l}}\mathbf{p}_i$, associated with its normal vector ${}^{\mathrm{l}}\boldsymbol{\eta}_i$, lies on a surface parallel to the robot's base xy-plane (*i.e.*, the ground). When ${}^{\mathrm{f}}\boldsymbol{\eta}_i$ is parallel to ${}^{\mathrm{f}}\mathbf{z}$, the cosine value θ_{η_i} equals 1 (since the angle is 0 or π, but typically normalized to 0 for parallel surfaces). However, due to noisy measurements and small delays in sensor data collection, it is good practice to apply a minimal threshold ϵ_η and check whether θ_{η_i} falls above a value close to 1.

Given the above explanations, the final filter equation can be expressed as:

$${}^{\mathrm{l}}\mathcal{P}^* = \{ {}^{\mathrm{l}}\mathbf{p}_i \in {}^{\mathrm{l}}\mathcal{P} \mid \theta_{\eta_i} > 1 - \epsilon_\eta \} \tag{4.19}$$

4.2.3 Statistical loop closure detection with Gaussian Scan Context

In Section 4.2.1, the use of SC++ for the LCD step of LEO-SLAM is reported. Although this approach is one of the most widely used among the SoTA algorithms due to its simplicity and effectiveness, it relies on certain assumptions regarding the noise present

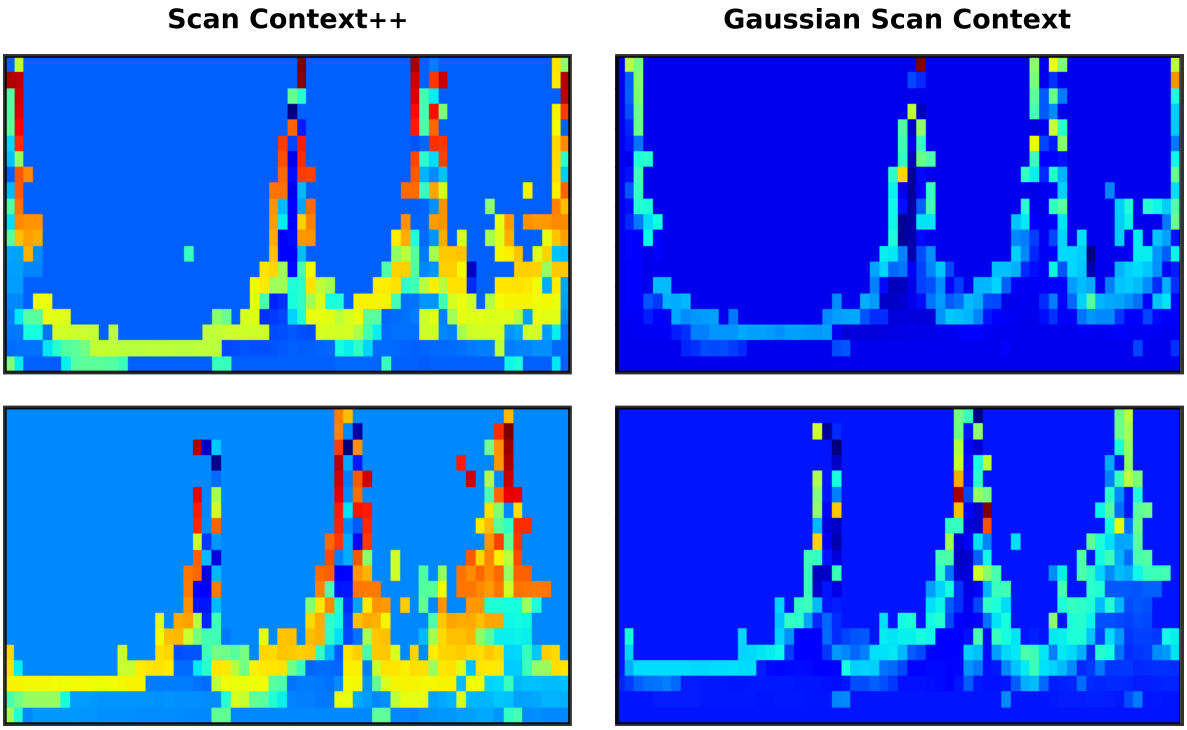


Figure 4.13: The plots compare SC++ (left) and GSC (right) during a loop closure in the KITTI sequence 00. The GSC matrix, displayed at the lower right, is derived from the Gaussian distributions illustrated in Fig. 4.12.

in the point clouds. This technique performs well when the input point cloud contains few outliers or minimal noise, a condition not always met in real-world scenarios. To address this problem, the GSC, illustrated in Fig. 4.12, is introduced. The GSC is a statistical enhancement designed to improve the robustness and performance of LCD by explicitly incorporating Gaussian-based analysis of the point cloud data. Before proposing the specific methodology, a deeper overview of SC++ is provided in the next section to thoroughly establish its foundations.

Scan Context++: methodology overview

SC++ is an enhanced LiDAR-based place recognition framework designed to address rotational and lateral variations prevalent in urban driving environments.

The process extends the original SC descriptor by introducing mechanisms that improve efficiency and robustness without relying on learning-based components. The procedure begins by transforming the 3D LiDAR point cloud data into a compact 2D matrix $\mathbf{C} \in \mathbb{R}^{N_r \times N_s}$, referred to as the SC++ (see Fig. 4.13). Here, N_r represents the number of radial bins (rings) and N_s the number of angular bins (sectors).

The point cloud is first projected onto a polar grid $\mathcal{P}$, defined by N_r concentric rings and N_s angular sectors. Each cell (bin) in the polar grid is denoted by $\mathbf{b}_{i,j}$ and contains $D_{i,j}$ points $\boldsymbol{p} \in \mathbb{R}^4$, each representing a single point (p^x, p^y, p^z, p^ψ) expressed in the LiDAR frame, falling within the spatial extent of that bin, such that $\mathbf{b}_{i,j} \in \mathbb{R}^{D_{i,j} \times 4}$, where $i = 1, \ldots, \mathrm{N}_r$ and $j = 1, \ldots, \mathrm{N}_s$. The ring index i and the sector index j are computed as:

$$i = \max(\min(\mathrm{N}_r, \lceil \frac{p_r}{\max_r} \rceil \cdot \mathrm{N}_r), 1), \tag{4.20}$$

$$j = \max(\min(\mathrm{N}_s, \lceil \frac{p_\theta}{\max_\theta} \rceil \cdot \mathrm{N}_s), 1), \tag{4.21}$$

where p_r and p_θ denote the radial distance and azimuthal angle of the point $\boldsymbol{p}$ in polar coordinates; $\max_r$ and $\max_\theta = 360°$ represent their respective maximum values; and $\lceil \cdot \rceil$ denotes the ceiling function.

Each bin $\mathbf{b}_{i,j}$ is subsequently used to compute the SC++ cell $\mathbf{C}_{i,j}$ by storing the maximum height value of the points contained within the corresponding bin $\mathbf{b}_{i,j}$, i.e., p^z, as defined by the following equation:

$$\mathbf{C}_{i,j} = \max_{\boldsymbol{p} \in \mathbf{b}_{i,j}} \; p^z. \tag{4.22}$$

This representation captures the egocentric spatial structure of the environment but remains inherently rotation-variant.

To achieve *rotation invariance*, SC++ computes a summary vector, referred to as the *ring key*, which is obtained by taking the L_0-norm (i.e., counting the non-zero entries) across each row of $\mathbf{C}$. Formally, the ring key $\mathbf{k} \in \mathbb{R}^{\mathrm{N}_r}$ is defined as:

$$k_i = \|\mathbf{C}_{i,:}\|_0 \quad \text{for } i = 1, 2, \ldots, \mathrm{N}_r. \tag{4.23}$$

These ring keys are stored in a k-d tree to enable efficient nearest-neighbor search during the place-recognition query stage.

Once potential matches are retrieved using the ring keys, the system performs *descriptor alignment* to compensate for rotational discrepancies. This is achieved by cyclically shifting the columns of the candidate matrix $\mathbf{C}_c$ and computing the cosine similarity with the query matrix $\mathbf{C}_q$ at each shift. The optimal alignment is determined as:

$$\mathrm{sim}(\mathbf{C}_q, \mathbf{C}_c^{(\theta)}) = \frac{\langle \mathbf{C}_q, \mathbf{C}_c^{(\theta)} \rangle}{\|\mathbf{C}_q\|_F \|\mathbf{C}_c^{(\theta)}\|_F}, \tag{4.24}$$

where θ denotes the circular shift index and $\|\cdot\|_F$ represents the Frobenius norm.

To enhance robustness against *lateral variations*, which arise when the same location is revisited from different lateral offsets (*e.g.*, distinct driving lanes), SC++ augments the original point cloud by synthetically shifting it laterally before generating new SC++ descriptors. These *virtual SCs* emulate the effect of observing the same scene from slightly displaced viewpoints, enabling the system to maintain reliable matching across such offsets.

Subsequently, a *false positive rejection* mechanism is employed. This procedure re-evaluates the similarity of the full SC++ descriptors (beyond the ring keys) and applies a predefined threshold τ to determine whether the match is accepted or rejected according to the following criterion:

$$\mathrm{sim}(\mathbf{C}_q, \mathbf{C}_c^{(\theta^*)}) > \tau, \tag{4.25}$$

where θ^* is the optimal rotation alignment identified earlier.

Through these combined steps, SC++ attains efficient and accurate place recognition, making it suitable for real-world autonomous navigation scenarios.

Gaussian extension

Although the latest version of SC++ offers substantial improvements and is sufficiently robust for integration into 3D LiDAR SLAM systems, it employs a relatively simplistic strategy to populate the context matrix $\mathbf{C}$ (see Eq. (4.22)). Specifically, each matrix entry is determined by selecting the maximum height value among the points within the corresponding bin $\boldsymbol{b}_{i,j}$. While computationally efficient, this maximum-value approach exhibits limited robustness to outliers, such as spurious reflections or sensor noise, which are commonly encountered in both indoor and outdoor LiDAR mapping environments.

To overcome this limitation, a statistical enhancement that accounts for the full distribution of points within each bin is introduced. For each bin $\boldsymbol{b}_{i,j}$, an associated multivariate normal distribution $\boldsymbol{\mathcal{G}}_{i,j} \equiv \mathcal{N}(\boldsymbol{\mu}_{i,j}, \boldsymbol{\Sigma}_{i,j})$ is defined, where the mean vector $\boldsymbol{\mu}_{i,j}$ and covariance matrix $\boldsymbol{\Sigma}_{i,j}$ are computed as follows:

$$\boldsymbol{\mu}_{i,j} = \frac{1}{\mathrm{N}} \sum_{\boldsymbol{p} \in \mathbf{b}_{i,j}} \mathbf{p}, \tag{4.26}$$

$$\boldsymbol{\Sigma}_{i,j} = \frac{1}{\mathrm{N}-1} \sum_{\boldsymbol{p} \in \mathbf{b}_{i,j}} (\mathbf{p} - \boldsymbol{\mu})(\mathbf{p} - \boldsymbol{\mu})^\top, \tag{4.27}$$

where $\mathrm{N} = |\mathbf{b}_{i,j}|$ is the number of points within bin $\mathbf{b}_{i,j}$. The covariance matrix is

calculated using a denominator of $N - 1$ to yield an unbiased estimator.

As demonstrated in Section 5.2.3, this straightforward statistical model yields significant improvements in both the accuracy and robustness of LCD. While a univariate Gaussian $\mathcal{N}(\mu, \sigma^2)$ based solely on height values could have been employed with the same results, the multivariate formulation is used because it captures the spatial structure of the data more effectively, which can support future extensions and more sophisticated implementations.

After computing all Gaussian distributions $\boldsymbol{\mathcal{G}}_{i,j}$, the GSC matrix $\mathbf{C}$ can be populated as follows:

$$\mathbf{C}_{i,j} = \boldsymbol{\mu}^z_{i,j} + \alpha \; \Sigma^{zz}_{i,j}, \tag{4.28}$$

where $\boldsymbol{\mu}^z_{i,j}$ and $\Sigma^{zz}_{i,j}$ represent the mean and variance of the height (z) component of the bin $\boldsymbol{b}_{i,j}$, and $\alpha \in \mathbb{R}^+$ is a tunable hyperparameter that controls the contribution of the variance term. This formulation preserves the core concept of estimating the maximum height, as originally applied in Eq. (4.22), but achieves this by statistically modeling the complete height distribution of the points within each bin. Consequently, the resulting context descriptor becomes more resilient to outliers and noise while retaining meaningful geometric discrimination between locations.

To further improve robustness against outliers and heavy-tailed noise, the weighted Gaussian matrix $\boldsymbol{\mathcal{G}}_h \equiv \mathcal{N}_h(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)$ is introduced, which employs Huber weighting to produce a more compact and representative Gaussian distribution for each bin. This method further reduces the influence of outlier points that might otherwise distort the statistical characterization of the point cloud segment.

The central principle of the Huber weight is to down-weight points that deviate substantially from the estimated mean $\boldsymbol{\mu}$, thereby mitigating their effect on the resulting distribution $\boldsymbol{\mathcal{G}}_h$. The residual r is defined as the Euclidean distance between a point $\boldsymbol{p}$ and the bin mean:

$$r = \|\boldsymbol{p} - \boldsymbol{\mu}\|_2 \tag{4.29}$$

The corresponding weight is derived from the Huber loss function, parameterized by a user-defined threshold δ:

$$\rho_\delta(r) = \begin{cases} \frac{1}{2}r^2 & \text{if } |r| \leq \delta \\ \delta(|r| - \frac{1}{2}\delta) & \text{if } |r| > \delta \end{cases} \tag{4.30}$$

This piecewise function exhibits quadratic behavior for small residuals, enhancing sensitivity to inliers, and linear behavior for large residuals, thereby reducing the influ-

ence of outliers. By incorporating this weighting scheme into the computation of the Gaussian parameters, a more robust and representative characterization of the spatial structure within each bin is achieved.

To further suppress the effect of outliers, a simplified Huber weight function derived from the original Huber loss is employed, defined as:

$$\omega_k = \begin{cases} 1 & \text{if } |r_k| \leq \delta \\ \frac{\delta}{|r_k|} & \text{if } |r_k| > \delta \end{cases} \tag{4.31}$$

where r_k denotes the residual for point $\mathbf{p}_k$, and δ is a user-defined threshold that controls the transition between full weighting and reduced weighting.

Using these weights, the Huber-weighted mean $\boldsymbol{\mu}_h$ and covariance matrix $\boldsymbol{\Sigma}_h$ for each bin is computed as follows:

$$\boldsymbol{\mu}_h = \frac{\sum_{k=1}^{\mathrm{N}} \omega_k \ \mathbf{p}_k}{\sum_{k=1}^{\mathrm{N}} \omega_k} \tag{4.32}$$

$$\boldsymbol{\Sigma}_h = \frac{\sum_{k=1}^{\mathrm{N}} \omega_k \ (\mathbf{p}_k - \boldsymbol{\mu}_h)(\mathbf{p}_k - \boldsymbol{\mu}_h)^T}{\sum_{k=1}^{\mathrm{N}} \omega_k} \tag{4.33}$$

where N denotes the number of points in the bin, and ω_k represents the Huber weight associated with point $\mathbf{p}_k$.

Applying Huber weighting to estimate Gaussian statistics offers several notable advantages:

- The different regime for large residuals mitigates the influence of outliers, preventing them from disproportionately affecting the statistics.
- The parameter δ offers a tunable trade-off between sensitivity to inliers and robustness against outliers.
- Huber weighting preserves more information than hard outlier rejection or trimming approaches.
- The resulting Gaussian estimates (mean and covariance) more accurately capture the underlying structure of noisy or imperfect data.

After computing all Huber-weighted Gaussian distributions, the final Huber-enhanced GSC matrix $\mathbf{C}_h$ is built using Eq. (4.28), where $\boldsymbol{\mu}^z_{i,j}$ and $\Sigma^{zz}_{i,j}$ are replaced by their Huber-weighted counterparts $\boldsymbol{\mu}^z_{h_{i,j}}$ and $\Sigma^{zz}_{h_{i,j}}$ respectively. As shown in Fig. 4.13, this process

yields a denoised matrix that highlights the scan characteristics, thereby improving LCD performance.

4.3 Robot environmental contextual understanding and interaction with semantics

In this section, a solution is proposed to augment the geometric mapping capabilities of a mobile robot with semantic high-level understanding. A mobile robot would benefit enormously from this information, as it would expand the knowledge of its surroundings. This additional context can be utilized by the robot to make informed decisions and to interact with and manipulate the environment.

4.3.1 Semantic mapping of objects of interest

This section presents Artifacts Mapping, a semantic mapping framework designed to map objects of interest in the 3D space. As shown in the pipeline in Fig. 4.14, this framework comprises two main functional blocks: object perception and object management.

Artifacts detection and position estimation

The perception part can be further divided into two components: *(i)* 2D object segmentation, and *(ii)* 3D object position estimation using camera-LiDAR filtering.

2D object segmentation In this phase, a deep neural network [12] is used to infer predefined object classes and their masks from RGB images (see Fig. 4.14a). During navigation, the robot captures images of the environment using its mounted camera. These images are then processed by an instance segmentation deep neural network, which outputs the classification labels and masks (i.e., a binary image with 1 where the object is found and 0 elsewhere) for each object recognized on the image (see Fig. 4.14d).

The outputs are grouped and passed to the next module, which converts these 2D data into 3D data. An optional feature provided in this module is the possibility to filter out specific classes in real-time upon request, allowing the robot to map different objects online depending on the current requirements. Other implementation aspects will be further explained in Section 5.3.1.

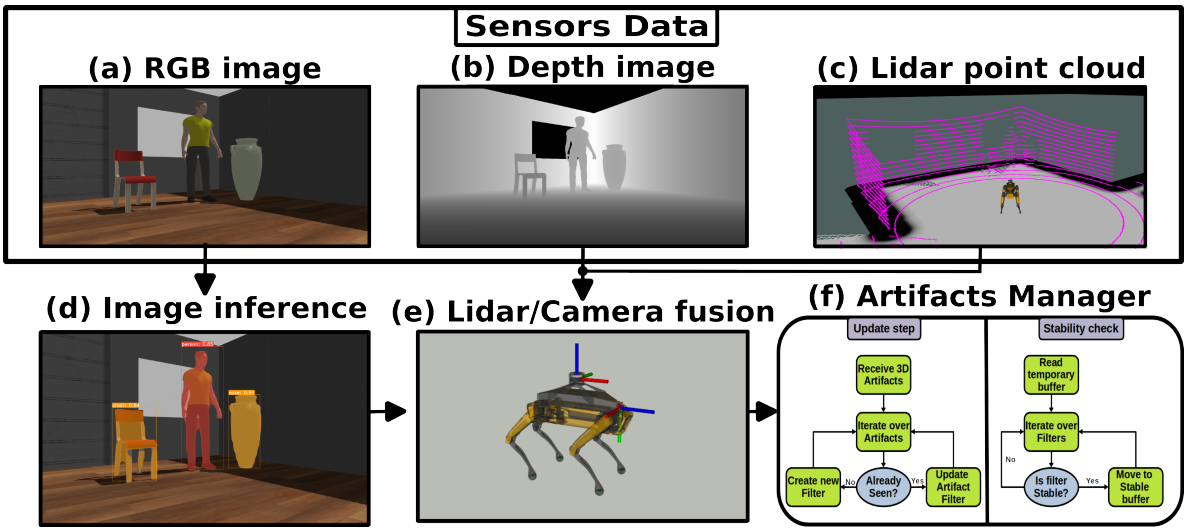


Figure 4.14: This figure represents the whole Artifacts Mapping pipeline. The top block groups the sensors' data readings: (a) camera RGB image, (b) camera depth image, and (c) LiDAR point cloud. At the bottom, there are (d) the RGB image inference performed with a Deep Neural Network for instance segmentation, (e) the multi-modal sensor fusion for detection and localization which uses as input the camera depth, the LiDAR point cloud and the Neural Network inference, and (f) a representation of the artifacts manager state-machines used to handle the sensor fusion detections and stabilize them. Reprinted, with permission, from [109], © 2023 IEEE.

3D object position estimation using camera-LiDAR filtering This module fuses RGB-D camera and LiDAR measurements to provide a precise estimate of the objects' positions in the environment. The input is composed of the classification labels and masks found in the previous module, along with depth information extracted from the camera (see Fig. 4.14b) and the LiDAR (see Fig. 4.14c). Sensors' depth measurements are first analyzed separately in the following discussion.

The depth image obtained from the camera (see Fig. 4.14b) is filtered using the recognized object masks through element-wise matrix multiplication. The resulting data, which contains only the depth information of the object plus sensor noise and environment outliers, is used to construct a 3D point cloud by projecting the 2D image points into 3D space using the following camera model equation:

$$\begin{bmatrix} {}^{\mathrm{c}}x \\ {}^{\mathrm{c}}y \\ {}^{\mathrm{c}}z \end{bmatrix} = \begin{bmatrix} \frac{1}{f_x} & 0 & -\frac{p_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{p_y}{f_y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} {}^{\mathrm{c}}z \tag{4.34}$$

where ${}^{\mathrm{c}}x$, ${}^{\mathrm{c}}y$, ${}^{\mathrm{c}}z$ are the 3D point coordinates with respect to the camera frame; u, v are the pixel coordinates on the image plane; and f_x, f_y, p_x, and p_y are the camera intrinsic parameters (focal distances and sensor principal point coordinates). Note that ${}^{\mathrm{c}}z$ is the depth value measured by the camera depth sensor.

The obtained point cloud ${}^{\mathrm{c}}\mathcal{P}$is first processed using a voxel grid downsampling filter[3] to reduce the point count. Subsequently, a radius outlier filter[4] is applied to remove outliers induced by sensor noise and segmentation imperfections. The final filtered point cloud is then used to compute the camera artifact centroid ${}^{\mathrm{c}}\boldsymbol{\zeta}$ as the mean of its constituent points.

The 3D LiDAR centroid estimation is computed as follows. The 3D LiDAR points ${}^{\mathrm{l}}p \in {}^{\mathrm{l}}\mathcal{P}$ (see Fig. 4.14c) are projected onto the 2D detected mask images using the extrinsic and intrinsic calibration parameters, as described by Eq. (4.35). The object points of interest are then extracted from the LiDAR point cloud (*i.e.*, the points whose

[3]Downsampling filter: `https://pointclouds.org/documentation/tutorials/voxel_grid.html`

[4]radius outlier removal: `https://pointclouds.org/documentation/tutorials/remove_outliers.html`

2D projections fall within the mask).

$$ {}^{\mathrm{l}}z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{\mathrm{c}}\mathbf{R}_{\mathrm{l}} & {}^{\mathrm{c}}\mathbf{t}_{\mathrm{l}} \end{bmatrix} \begin{bmatrix} {}^{\mathrm{l}}x \\ {}^{\mathrm{l}}y \\ {}^{\mathrm{l}}z \\ 1 \end{bmatrix} , \tag{4.35} $$

where ${}^{\mathrm{c}}\mathbf{R}_{\mathrm{l}} \in \mathbb{R}^{3\times3}$ and ${}^{\mathrm{c}}\mathbf{t}_{\mathrm{l}} \in \mathbb{R}^{3\times1}$ are the rotation matrix and the translation vector between the LiDAR and the camera frames, and ${}^{\mathrm{l}}x$, ${}^{\mathrm{l}}y$, ${}^{\mathrm{l}}z$ are the 3D coordinates of the LiDAR point ${}^{\mathrm{l}}p$. The remaining parameters are the same as those defined in Eq. (4.34).

The extracted point cloud representing the noisy artifact will then be filtered using a radius-based outlier filter similar to the one used for the camera. Both radius filter parameters are directly dependent on the number of point cloud points, because different distances and object sizes affect point cloud density and, consequently, the filtering efficacy. Finally, the point cloud mean is computed to obtain the LiDAR artifact centroid ${}^{\mathrm{l}}\boldsymbol{\zeta}$.

Once both centroid measurements are available, they are fused into the final artifact centroid $\boldsymbol{\zeta}$ following the piecewise rules defined in the equation:

$$ \boldsymbol{\zeta} = \begin{cases} 0 & \text{If } \mathrm{dist}_c < \mathrm{min}_c \\ {}^{\mathrm{c}}\boldsymbol{\zeta} & \text{If } \mathrm{min}_c \leq \mathrm{dist}_c \leq \mathrm{acc}_c \\ \xi \cdot {}^{\mathrm{c}}\boldsymbol{\zeta} + (1-\xi) \cdot {}^{\mathrm{l}}\boldsymbol{\zeta} & \text{If } \mathrm{acc}_c \leq \mathrm{dist}_c \leq \mathrm{max}_c \\ {}^{\mathrm{l}}\boldsymbol{\zeta} & \text{If } \mathrm{dist}_c > \mathrm{max}_c \ , \end{cases} \tag{4.36} $$

where dist_c is the Euclidean distance between the 3D point estimates and the camera; min_c and max_c are the minimum and maximum perception distances of the depth camera; acc_c is the distance within which the camera provides sufficiently accurate measurements to be used alone for object localization (a value typically provided by sensor vendors); ${}^{\mathrm{l}}\boldsymbol{\zeta} \in \mathbb{R}^3$ and ${}^{\mathrm{c}}\boldsymbol{\zeta} \in \mathbb{R}^3$ are the LiDAR and camera 3D centroid estimates, respectively; and $\xi \in [0,1] \in \mathbb{R}$ is the fusion weight represented by the blue slope of the segments between acc_c and max_c in Fig. 4.15, computed as follows:

$$ \xi = 1 - \frac{1}{\mathrm{max}_c - \mathrm{acc}_c}(\mathrm{dist}_c - \mathrm{acc}_c) \tag{4.37} $$

Using the filtered camera and LiDAR point clouds, a rough 3D radius estimate ρ for the objects is obtained. The camera radius ${}^{\mathrm{c}}\rho$ and the LiDAR radius ${}^{\mathrm{l}}\rho$ are computed as

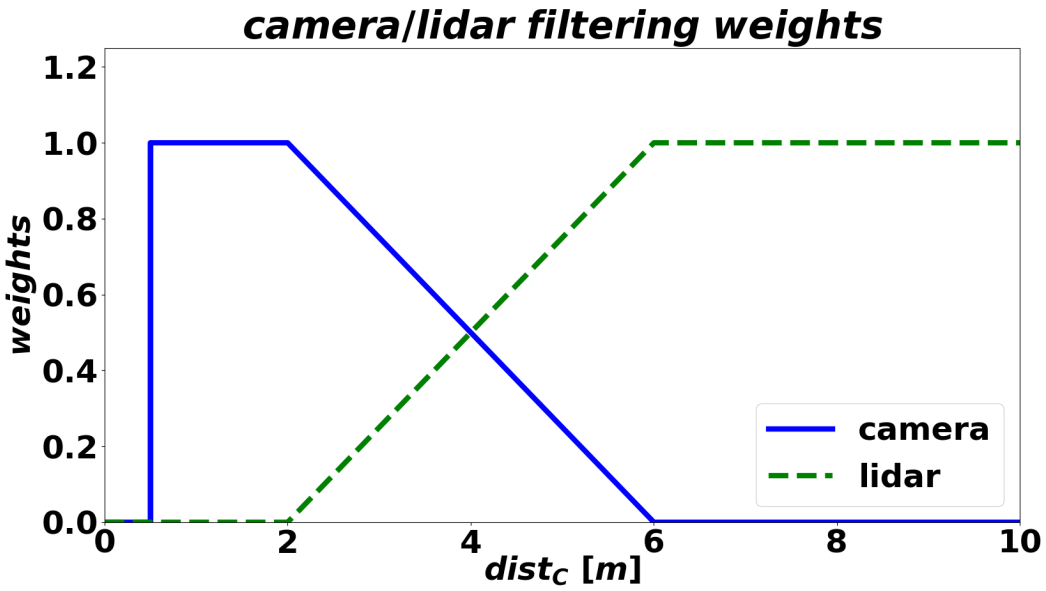


Figure 4.15: An example of the contribution weights of camera and LiDAR for sensor fusion. The camera weight (ξ) is shown in blue, while the LiDAR weight ($1-\xi$) is shown in dashed green. In this specific example, the specifications of a generic RGB-D camera are considered: $\min_c = 0.5$, $\text{acc}_c = 2.0$ and $\max_c = 6.0$. Reprinted, with permission, from [109], © 2023 IEEE.

the mean of the two largest dimensions along the x, y, and z point cloud axes. The final radius ρ is computed following the same centroid fusion rules of Eq. (4.36), substituting $\boldsymbol{\zeta}$ with ρ, ${}^{\text{c}}\boldsymbol{\zeta}$ with ${}^{\text{c}}\rho$ and ${}^{\text{l}}\boldsymbol{\zeta}$ with ${}^{\text{l}}\rho$.

Additionally, the artifact's view angle ϕ with respect to the robot is computed. This angle is rotated with respect to the map reference frame for implementation reasons, using the equation:

$$\phi = \arctan 2(r_{21}, r_{11}) + \arctan 2({}^{r}y,\ {}^{r}x) \tag{4.38}$$

where r_{ij} is the entry at row i and column j of the rotation matrix ${}^{\text{m}}\mathbf{R}_{\text{r}} \in \mathbb{R}^{3\times3}$ between the map frame m and the robot frame r, and ${}^{\text{r}}y$ and ${}^{\text{r}}x$ are the y and x positions of the artifact centroid with respect to the robot base. The two additive terms of Eq. (4.38) represent, respectively, the heading angle between the robot and the map and the angle between the robot and the 3D centroid.

Artifacts manager for data association

The manager (see Fig. 4.14f) is necessary to filter outliers and stabilize the artifact position estimations provided by the sensor fusion module. This entire procedure is commonly known as data association [1, 148]. The manager is composed of two asynchronously running, parallel modules: *(i)* object position filtering and *(ii)* object position stabilization.

Position filtering The perceived artifacts are temporarily stored in a dedicated data structure called the *temporary buffer*. Once the manager receives the 3D artifacts' position estimations from the perception module, it checks if the artifacts were previously observed, *i.e.*, whether the distance between one of the stored artifacts and the current measurement is less than the stored artifact's 3D radius. If an artifact has been seen before, the corresponding entry in the temporary buffer is updated. Otherwise, for each newly observed artifact, the manager creates a new artifact instance in the temporary buffer. These instances have their own moving-average filter, which estimates the average of the artifact centroid position and its radius using Eq. (4.39) and computes a variance based on the distances between the position measurements and the moving average within the filter horizon using Eq. (4.40).

$$\boldsymbol{\mu} = \frac{1}{\mathrm{N}} \sum_{\boldsymbol{\kappa} \in \mathrm{K}_N} \boldsymbol{\kappa} \tag{4.39}$$

$$\sigma = \frac{1}{\mathrm{N}} \sum_{\boldsymbol{\kappa} \in \mathrm{K}_N} ||\boldsymbol{\kappa} - \boldsymbol{\mu}||^2 \ , \tag{4.40}$$

where $\mathrm{N} \in \mathbb{N}$ is the number of measurements in the moving average set K_N of 3D points, $\boldsymbol{\kappa} \in \mathbb{R}^3$ represents the current 3D position measurement, $\boldsymbol{\mu} \in \mathbb{R}^3$ is the 3D mean position, and $\sigma \in \mathbb{R}$ represents the variance of the filter.

Position stabilization This module evaluates the stability of the artifacts in the temporary buffer and stores stable artifacts in a secondary, similar structure, the *stable buffer*. If an artifact in the temporary buffer is determined to be stable, the stabilizer transfers the artifact from the temporary buffer to the stable one. An artifact is considered stable when its moving average filter variance σ is less than half its 3D artifact radius ρ, and at least half the average filter set K_N is filled. This condition ensures there are enough consistent object position estimates, allowing the average object position to be used to fix the object position on the map.

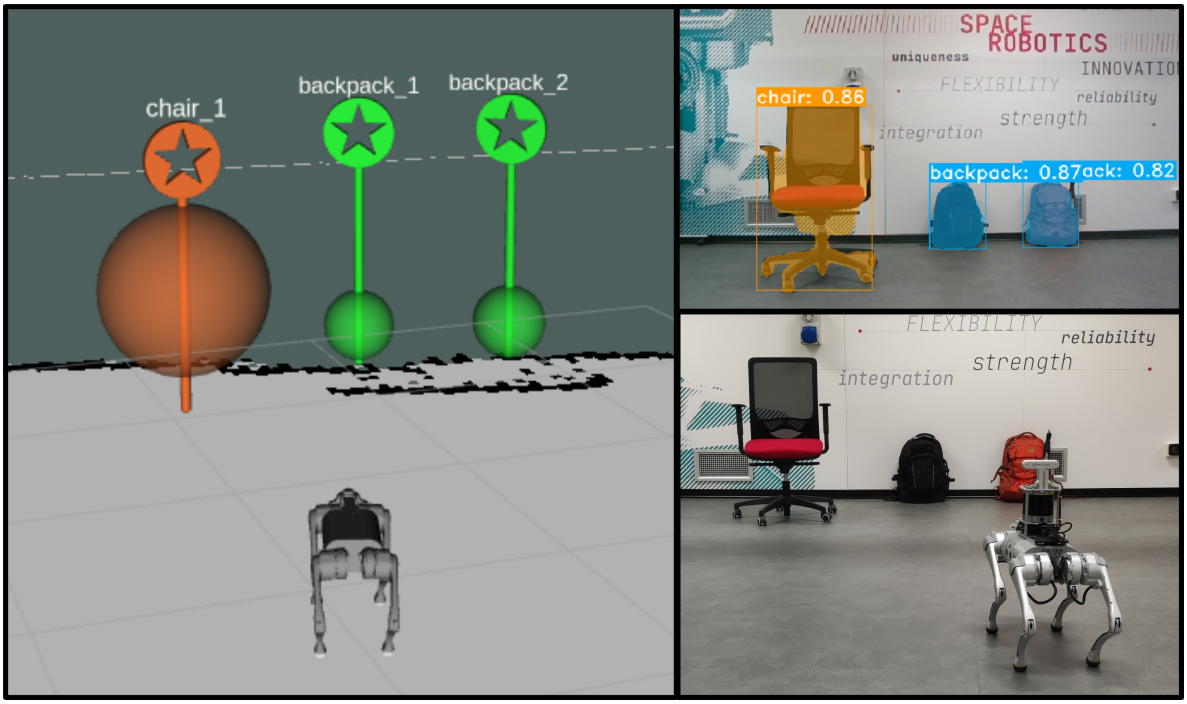


Figure 4.16: An example of the framework during an experiment. On the left is the visual application where objects are shown with a landmark and a spherical region of interest for the location in the RViz. On the top right is the instance segmentation inference of the image taken from the robot camera, while on the bottom right is the external representation of the experimental scene. Reprinted, with permission, from [109], © 2023 IEEE.

At the end of the Artifacts Mapping application, an additional final data association step is performed. Artifacts of the same class that exhibit overlap on the XY plane are merged into a single artifact. This step reduces the number of duplicated objects that can appear on the map due to different FOV and occlusions. Following this process, the stable artifacts buffer is saved in a YAML file, which can then be loaded into the user interface application presented in the next section.

User Interface for goal sending A User Interface (UI) application based on a RViz plugin (see Fig. 4.16) was developed to provide intuitive visualization of the artifacts on the map, to send commands to the robot for moving near an artifact of interest, and to delete unnecessary or incorrectly identified artifacts. These artifacts can be loaded from the YAML file obtained using the Artifacts Mapping application. Through the UI application, the user can send *nav_msgs/goal* ROS messages, which the robot can utilize to navigate towards the object (*e.g.*, using the ROS navigation stack, see Section 5.3). The user can interact with the artifacts in RViz by right-clicking them

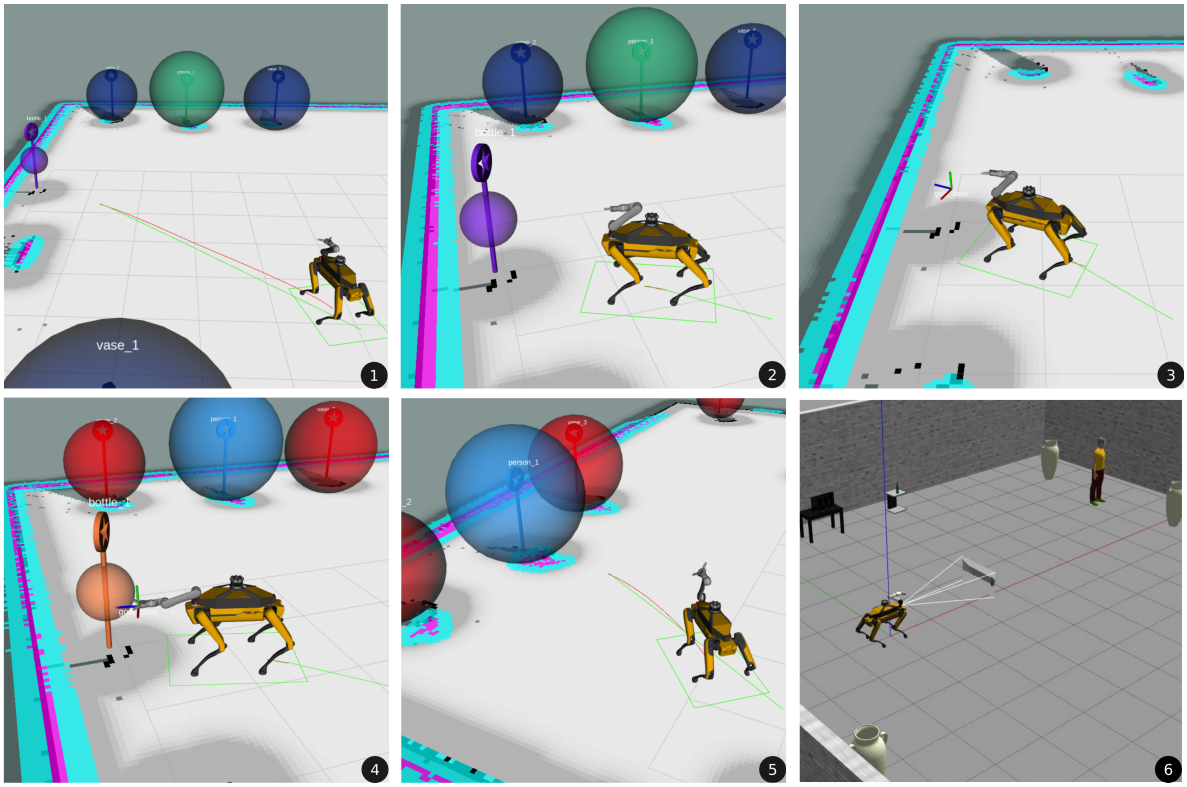


Figure 4.17: Autonomous actions held during the execution of the application. Image 1 represents the navigation towards the requested object. In image 2, the object is approached, while in image 3, the object pose is estimated. In images 4 and 5, the object is picked and brought to the person. Image 6 shows the whole simulated world. Reprinted, with permission, from [108], © 2023 Springer Nature.

and selecting the *Go To* or *Delete* action. Since the artifact's centroid lies within its bounding shape, the navigation goal is automatically offset to a position in front of the artifact, ensuring the robot stops before colliding with the object.

The other available option is artifact deletion. Suppose the user notices that an artifact is wrongly identified (in terms of classification or position). The user can delete it, and once the UI application is closed, the loaded YAML file is updated with only the remaining artifacts.

Example of semantic mapping usage in loco-manipulation applications

In this section, an overview is provided of how Artifacts Mapping can be employed in a simple pick-and-place loco-manipulation task.

To manage the application workflow, a Behaviour Tree (BT)[5] is utilized. The

[5]BT library: `https://py-trees.readthedocs.io/en/devel/`

BT is composed of a sequence of six actions: *navigate_ to_ object*, *approach_ object*, *pose_ estimation*, *pick_ object*, *bring_ to_ user*, and *release_ object*, along with a safety action *abort* which is activated when any of the other actions fails. Most of these actions are represented in Fig. 4.17. The BT waits until the user initiates a request, which is defined by the string associated with the artifact, e.g., "bottle_1". After the request is received, the BT starts ticking and executes the behaviors explained hereafter.

navigate_ to_ object The rough positions of both the requested object and the target person are already known, having been previously documented during the mapping process [109]. This action involves retrieving the object's stored position and sending a navigation goal to the robot. The goal is offset to an obstacle-free region located immediately in front of the required object, prompting the robot to calculate its path and navigate toward the location.

approach_ object The object approaching phase is necessary to facilitate a smooth picking action by refining the robot's position relative to the artifact. This refinement is required because the goal position received in the previous step is imprecise. During this phase, the robot autonomously approaches the artifact, striving to center it in the camera's field of view and reduce the distance such that the object falls within the robot's manipulability space.

To accomplish this, the robot employs an instance segmentation network to segment the object and uses the resulting mask to crop both the RGB-D images. Subsequently, a point cloud is built using the masked depth image and the camera's intrinsic parameters. For each non-zero pixel of the masked depth, the point ${}^{\mathrm{b}}\mathbf{p}$ in the robot base frame b is computed as ${}^{\mathrm{b}}\mathbf{p} = {}^{\mathrm{b}}\mathbf{t}_c + {}^{\mathrm{c}}\mathbf{p}$, where ${}^{\mathrm{b}}\mathbf{t}_c$ is the translation vector from the base frame b to the camera frame c. The point ${}^{\mathrm{c}}\mathbf{p}$ in the camera frame c is calculated like in Eq 4.34:

$$ {}^{\mathrm{c}}\mathbf{p} = \begin{bmatrix} \frac{1}{f_x} & 0 & -\frac{p_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{p_y}{f_y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} {}^{\mathrm{c}}d \tag{4.41} $$

where f_x, f_y, p_x, and p_y are the camera intrinsic parameters (*i.e.*, the focal lengths and principal points along the x and y image axes), u and v are the depth pixel position along the x and y axes, and ${}^{\mathrm{c}}d$ is the depth value of the considered pixel.

Using this point cloud, the object's centroid ${}^{\mathrm{b}}\boldsymbol{\zeta}$ expressed in the base frame b, is computed. Two key metrics are derived from this centroid: the 3D Euclidean distance

to the robot base, d_{obj}, and the heading angle between the robot and the object, θ_{obj}, calculated by:

$$\theta_{obj} = \arctan 2(y, x) \tag{4.42}$$

where x and y are the coordinates of the centroid ${}^{\mathrm{b}}\boldsymbol{\zeta}$ (with the xy plane assumed parallel to the ground floor), and $\arctan 2$ is the two-argument arc tangent function that accounts for the quadrant.

Two linear proportional controllers (*e.g.*, Eq. (4.43)) are then used to regulate the robot base position by sending velocity commands (linear velocity along the x-axis, v_x, and angular velocity around the z-axis, ω_z). The controllers use optimal distance d^* and optimal angle θ^* as references:

$$u = P(p - p^*) \ , \tag{4.43}$$

where u is the linear v_x or angular ω_z velocity output, P is the proportional gain, p is the current distance d_{obj} or the current heading angle θ_{obj}, and p^* is the optimal distance d^* or angle reference θ^*.

pose_estimation Once the robot is well positioned, the homogeneous transformation ${}^{\mathrm{b}}\mathbf{H}_{\mathrm{g}}$ of the grasping pose g expressed in the base frame b is computed. Using the instance segmentation neural network and the RGB-D camera, the translation vector ${}^{\mathrm{b}}\mathbf{t}_{\mathrm{g}}$ from base frame b to the grasping pose g is computed in the same way as ${}^{\mathrm{b}}\boldsymbol{\zeta}$ in the *approach_object* behavior. The rotation matrix ${}^{\mathrm{b}}\mathbf{R}_{\mathrm{g}}$ between the robot base b and the grasping pose g is simply set equal to an identity matrix $\mathbf{I_3}$ (${}^{\mathrm{b}}\mathbf{R}_{\mathrm{g}} = \mathbf{I_3}$). This yields the homogeneous transformation:

$${}^{\mathrm{b}}\mathbf{H}_{\mathrm{g}} = \begin{bmatrix} {}^{\mathrm{b}}\mathbf{R}_{\mathrm{g}} & {}^{\mathrm{b}}\mathbf{t}_{\mathrm{g}} \\ 0\ 0\ 0 & 1 \end{bmatrix} \tag{4.44}$$

pick_object In this phase, the robot receives the desired grasping homogeneous transformation ${}^{\mathrm{b}}\mathbf{H}_{\mathrm{g}}$ and roto-translates it into the end-effector frame e to determine the final end-effector pose for the action:

$${}^{\mathrm{e}}\mathbf{H}_{\mathrm{g}} = {}^{\mathrm{e}}\mathbf{H}_{\mathrm{b}}\ {}^{\mathrm{b}}\mathbf{H}_{\mathrm{g}} \tag{4.45}$$

In this experiment, the simulation uses an optimization-based whole-body inverse dynamics controller [102] to move the arm towards the target pose. However, the specific method for achieving this target pose depends entirely on the robot configuration used.

When the robotic end-effector reaches the picking position, the gripper closes to grasp the artifact, and the arm subsequently moves to a folded position for safe transportation.

bring_to_user Once the artifact is successfully grasped, the robot navigates through the environment to deliver it to the user.

release_object Finally, when the robot reaches the user's position, it releases the requested object to the user. After this step, the pipeline is completed, and the robot becomes available to receive other tasks.

abort The abort status is essential for managing unexpected behaviors or failures. In this state, all currently running goals or requests are immediately halted, the mission is deleted, and the robot enters an idle state, promptly reporting the error to the user while awaiting further commands.

Chapter 5

Experiments

5.1 Re-identification validations

In this section, the proposed evaluation methodology is presented to formally validate the solutions detailed in Section 4.1 for the Re-ID problem stated in Section 3.1.1.

5.1.1 Re-ID baseline evaluation

The evaluation begins with the first Re-ID pipeline presented in Section 4.1.1, FollowMe [111]. Three different experiments are conducted to validate the system: individual modules (*i.e.*, hand gesture classification, visual Re-ID and the integrated system framework.

Experimental setup

The experimental setup utilizes an *Intel® RealSense™ D415* [1] camera, a notebook equipped with an *Intel® Core™ i9-11950H* processor and an *NVIDIA Geforce RTX 3080 Laptop* GPU, and a Robotnik RB-Kairos+ 5e [2] serving as the assistant robot.

The camera was fixed onto the UR5e robotic arm wrist using a 3D printed adapter, and the robotic arm was positioned vertically to point the camera forward (see Fig. 4.3). The safety circle radius is set to 1.25 m, determined by applying Eq. (4.3) with a maximum velocity $v_{\max} = 0.3\,\mathrm{m/s}$ and an expected response time $t_{\exp} = 3\,\mathrm{s}$. The navigation relies on an Adaptive Monte Carlo Localization algorithm [149] to estimate the robot's position using odometry and laser scan data.

[1] *Intel® RealSense™ D415*: `https://www.intelrealsense.com/depth-camera-d415/`

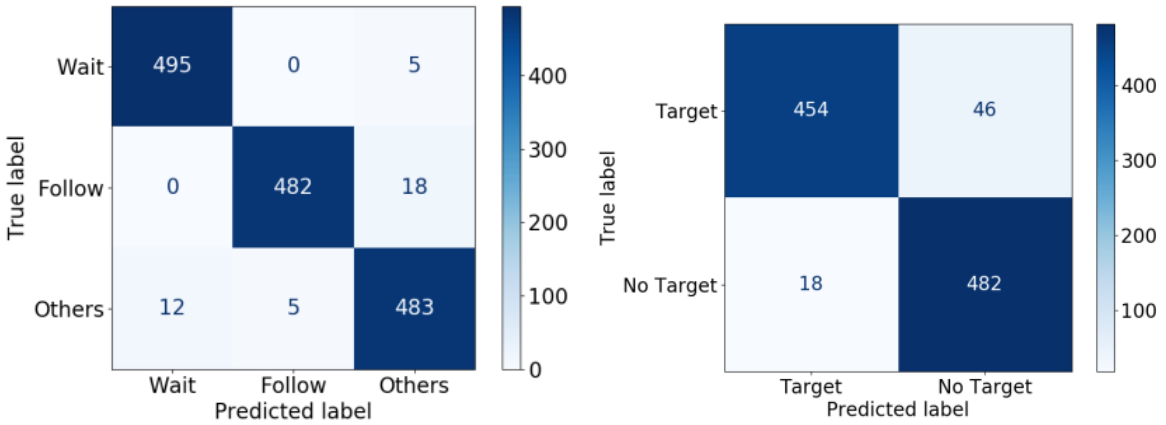
[2] RB-Kairos+ Mobile Manipulator: `https://robotnik.eu/products/mobile-manipulators/rb-kairos/#more-versions`

Regarding the Re-ID experiments, a pre-trained IBN-ResNet-50 network [85] on the popular MSMT17 dataset [145] is employed. The feature dimension D of the network is set to 256. This dimension is a popular choice in Re-ID settings as it represents a good trade-off between performance and feature vector length.

Table 5.1: Classification metrics over a test set of images. on the left, hand gestures, and on the right, Re-ID. Reprinted, with permission, from [111], © 2023 IEEE.

	Hand Gestures			Re-ID	
	Wait	*Follow*	*Others*	*Target*	*No Target*
Precision	0.98	0.99	0.96	0.96	0.91
Recall	0.99	0.96	0.97	0.91	0.96
F1-score	0.98	0.97	0.96	0.93	0.93
Accuracy	0.97			0.94	

Hand gesture classification experiments



(a) Average confusion matrix for hand gesture over a test set of images.

(b) Confusion matrix for Re-ID over a test set of images.

Figure 5.1: Comparison between panoptic and instance segmentation inferences. Reprinted, with permission, from [111], © 2023 IEEE.

To train the SVM classifier, an initial dataset of 400 hand images in different positions was collected. For the experimental validation of this module, a standard classification validation was used involving a group of eight subjects, for each subject and each class (i.e., *Wait*, *Follow*, *Others*), hand key-points were extracted from 500 images,

resulting in a total of 4000 images per class (12000 images total). These data were classified using the trained SVM. Using the results and the ground truth labels, the mean confusion matrix (averaged over the subjects) and various classification metrics were computed and are presented in Fig. 5.1a and the left part of Table 5.1, respectively. The resulting metrics confirm the algorithm's strong ability to distinguish between the intended gestures across a diverse group of actors.

Re-identification experiments

To validate the Re-ID module, a dataset comprising 8500 images was collected. The calibration phase required 500 images with a single subject for each person ($500 \times 8 = 4000$ images). The remaining 4500 images were collected according to the following scheme: 500 images with all subjects present, and 500 images for each person, with all subjects except the person of interest included.

The person Re-ID module was calibrated using the 500 calibration images, of which two-thirds were used for feature extraction and one-third for threshold computation. For the testing phase, 1000 images per subject were used, divided into two sets: the set where all people were present (Set A) and the set where all people except the calibrated one were present (Set B). The classification was evaluated as follows:

- *Set A* (Target Present): Classification is correct if and only if the module correctly re-identifies the target, *i.e.*, it does not re-identify another person or no one.

- *Set B* (Target Absent): Classification is considered correct if and only if the module does not re-identify anyone, *i.e.*, all feature distances are above the pre-computed threshold.

The results of this process are represented in the confusion matrix in Fig. 5.1b and the right part of Table 5.1. The numbers are averaged per subject: one subject is used for calibration, and the others are used as distractors, alternating between them and then averaging the final results. The numerical results confirm that the chosen threshold computation favors False Negatives (top-right value in Fig. 5.1b) while significantly penalizing False Positives (bottom-left value in Fig. 5.1b), which is the desired conservative behavior for reliable Re-ID. This is also evident from the *Target* column of Table 5.1, where Precision exceeds Recall.

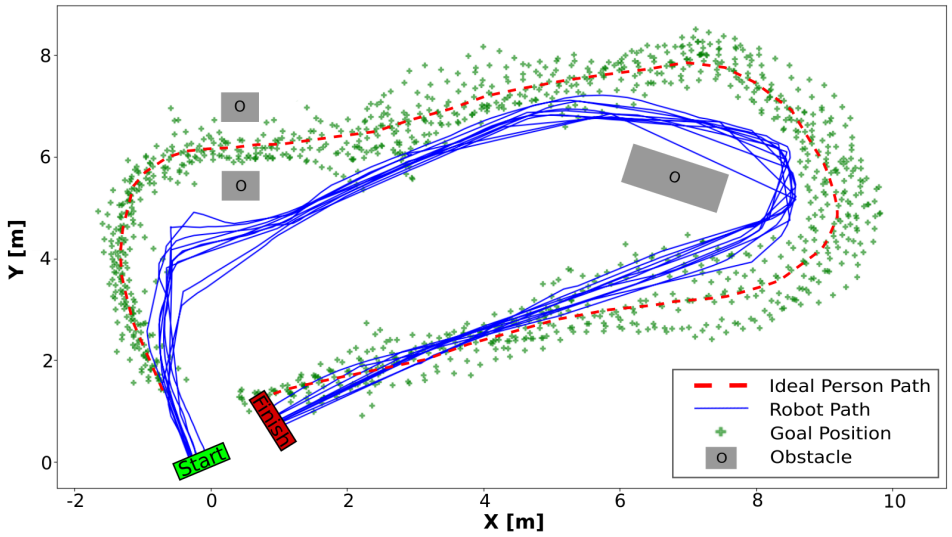


Figure 5.2: Results of the FollowMe whole experiment: each person has to follow an ideal path (red dashed line) while the robot (blue line) has to follow him/her. The goal positions computed from the perception module data are represented with green plus signs. The robot is placed in the green start position and has to follow the target until the red finish position is reached. Reprinted, with permission, from [111], © 2023 IEEE.

FollowMe whole experiments

The complete FollowMe framework was validated with a group of ten subjects. The robot starts from a fixed position in a laboratory area (approximately 100 m^2) where its empty map is known and some random unknown obstacles are placed. Each person was instructed to follow a predefined path while other subjects moved around (generally, no more than four people were present along the route). At the start, the robot waits for a *Follow* gesture command. When the target reaches the final station, the robot must be stopped using the *Wait* hand gesture. The experiment was performed once for each person. The qualitative results are shown in Fig. 5.2. The robot (blue path) successfully detects the person's position (green +). It follows the subject, avoiding unknown obstacles and maintaining a path pattern similar to the target's ideal path (dashed red line).

The time performance of the overall proposed framework was computed to estimate its time-frequency. The processing frequency, from camera acquisition to robot goal sending, ranges from 10 Hz to 7 Hz depending on the number of people in the image (1 to 10). This system runs online, with noticeable speed reductions as the number of detected people increases. However, in a typical real-world application scenario, it is rare to detect more than 10 non-occluded people.

5.1.2 Autonomous adaptation evaluation

In this section, a demonstration is provided of how the CARPE-ID method presented in Section 4.1.2 improves upon the FollowMe baseline approach described in Section 4.1.1.

To test the CARPE-ID framework, two experiments were conducted. Following a similar approach to the baseline evaluation in Section 5.1.1, the framework is first validated by capturing videos from different angles using a fixed camera in a laboratory setup. Secondly, to further validate the proposed method, it is tested in a HRI scenario, specifically a robot person-following scenario.

Similarly to the experimental setup in Section 5.1.1, the system was run on a notebook with an *Intel® Core™ i9-11950H* processor and an *NVIDIA Geforce RTX 3080 Laptop* GPU. For image acquisition, the *Intel® RealSense™ D455* camera and a Robotnik RB-Kairos+ 5e are used as the assistant robot. To extract features (a one-column vector of dimension $D = 256$), the same IBN-ResNet-50 [85] network trained on the MSMT17 dataset [145] is utilized.

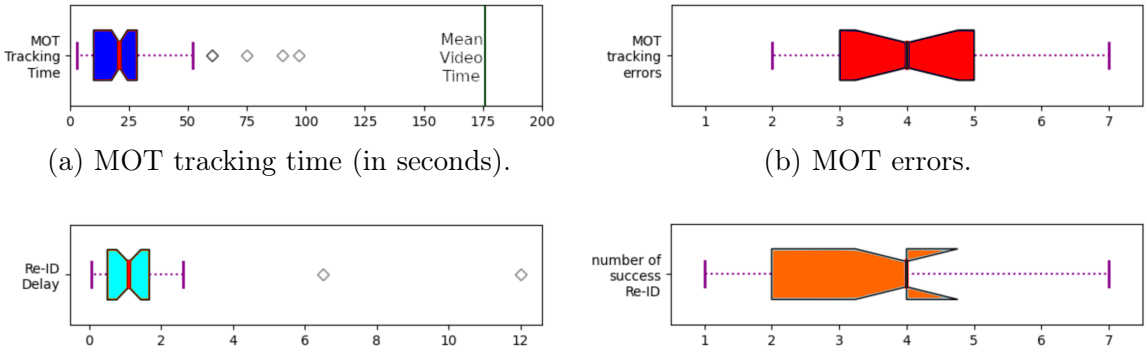
Individual experiments

The PersonPath22 dataset [127], a visual person-tracking dataset, was initially considered for evaluating this framework. However, this dataset was found to be unsuitable for the following reasons:

- The videos do not depict HRC scenarios. Instead, they mainly comprise security camera footage or videos of crowded environments where people are far from the camera and appear in the image for only short periods.
- The videos are too brief to accurately validate the Re-ID module's ability to handle identity changes over time.
- In most cases, the MOT algorithm used with this dataset already performed the correct tracking. People do not frequently enter and exit the camera's FOV, so the Re-ID module is unnecessary.

To address these limitations in validating the framework, a custom dataset was collected, comprising 18 videos totaling 53 minutes of footage. Additionally, for videos featuring multiple actors, the framework was evaluated separately for each person, bringing the total analyzed duration to 113 minutes. The videos were shot in two laboratory setups, featuring both single-person and group scenarios. Participants were asked to exit and re-enter the camera's FOV and to change their appearance (e.g., by varying their clothes)

to rigorously validate the framework's ability to re-identify individuals who are totally or partially occluded and to adapt the model based on newly acquired appearances.



(a) MOT tracking time (in seconds).

(b) MOT errors.

(c) The delay (in seconds) for Re-ID.

(d) The number of Re-ID.

Figure 5.3: Individual experiments statistics. Reprinted, with permission, from [112], © 2024 IEEE.

The experiments were analyzed, and the following performance statistics were computed:

- The SoTA MOT tracker exhibited minimum, mean, and maximum tracking lengths of 3.02, 21.2, and 52.2 seconds, respectively, with some outliers extending the maximum to 97.17 seconds (see Fig. 5.3a).
- The minimum, mean, and maximum Re-ID delay (the time taken by the framework to re-identify the target) were found to be 0.06, 1.1, and 2.6 seconds, respectively (as shown in Fig. 5.3c). However, there were two rare instances where the tracker took significantly longer, 6.5 and 12.1 seconds, for the Re-ID process.
- The MOT algorithm had a minimum and maximum failure rate of 2 and 7 times, respectively, with an average failure rate of 4 times for each video, as shown in Fig. 5.3b.
- The overall presented framework, across all videos, made only 2 errors in re-identifying the target. This low error count demonstrates the framework's reliability for tracking targets that are totally or partially occluded.
- The minimum, mean, and maximum number of successful Re-IDs made by the proposed framework were 1, 4, and 7, respectively (See Fig. 5.3d).

Following task experiments

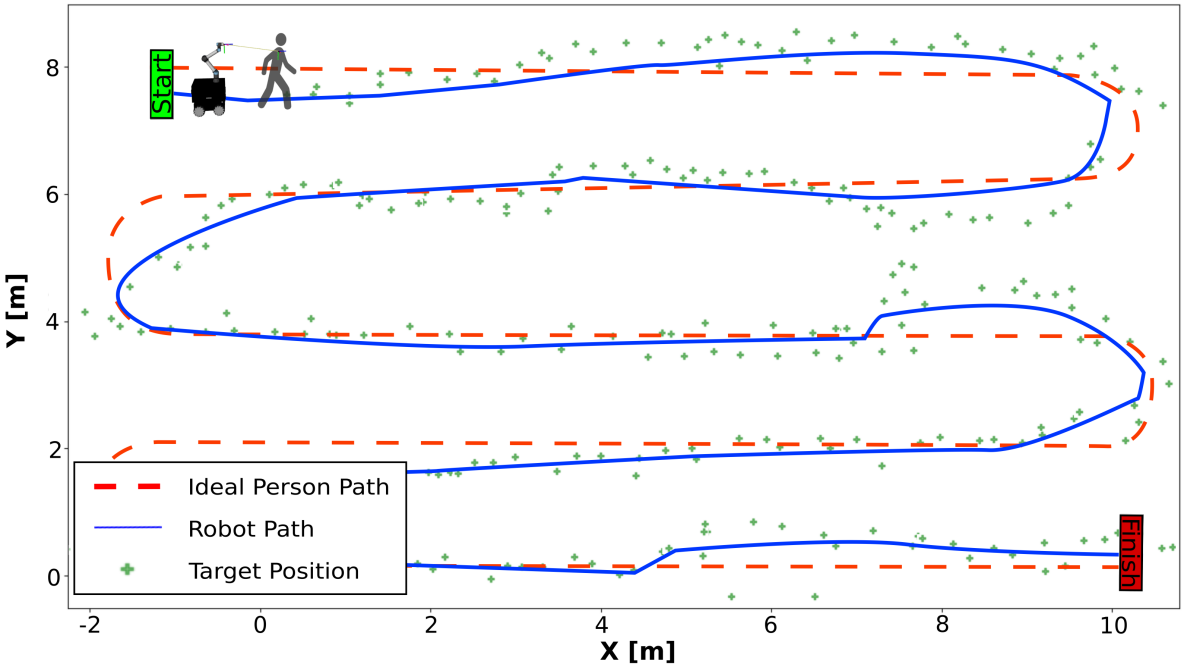


Figure 5.4: This is an example of a person-following experiment. The person being followed must roughly follow a particular path (indicated by the red dashed line), while the robot (indicated by the blue line) follows them. The green plus signs represent the target positions, which are calculated using the CARPE-ID framework tracking output. The robot starts at the green position and needs to follow the person until they reach the red finish position. Reprinted, with permission, from [112], © 2024 IEEE.

Real-world experiments were conducted to validate the proposed framework using a robot-person-following scenario similar to the one described in Section 5.1.1. In this setup, the target person was asked to move around while the robot followed them, avoiding obstacles. However, in this experiment, the targets actively changed appearance over time and navigated through barriers to test the framework's robustness to partial or complete occlusions.

The framework successfully tracked, re-identified, and followed the target person in all experiments, even in a laboratory setting where other people were working. An example of the experimental setup and results is represented in Fig. 5.4. The robot (blue line) successfully follows the target, who navigated along the ideal path (red dashed line). Ten experiments were performed with five different people as targets, covering a total distance of 837 meters.

Discussion

The evaluation results confirm the need for personalized approaches for Re-ID in HRI tasks. The required recognition capacities are challenging to achieve because the target appears in the image at different times and spaces and with varying appearances. The results achieved in Section 5.1.2 show that personalized approaches are necessary for re-identifying the target person in HRI tasks.

A more reliable application framework is required because SoTA and MOT methods proved limited in these experiments. This statement is supported by the results presented in Fig. 5.3a, where the MOT algorithm exhibits a mean tracking time of 21.2 seconds, significantly shorter than the mean video time of 176 seconds (minimum video time 94 seconds), indicating frequent track losses.

In Fig. 5.3c, the time delay between when the target person appears in the camera's FOV and the time they are re-identified is shown. The mean delay obtained (1.1 s) is acceptable for HRI applications. However, there were two cases in which the re-identifier failed to re-identify the target within 6 seconds. This occurred because the target person changed their appearance outside the camera's FOV (e.g., by changing clothing), which momentarily confounded the Re-ID module and led to a longer delay before successful Re-ID.

Fig. 5.3b and Fig. 5.3d compare the number of times the MOT algorithm lost track of the target with the number of times the Re-ID framework successfully re-identified the target. Both metrics have a mean value of 4, indicating that the MOT errors are reliably recovered by the proposed framework. However, the figures also reveal that MOT mistakes occur more frequently than successful re-identifications. This disparity arises because the MOT algorithm may quickly switch between track IDs over a short period, while the Re-ID delay takes longer to confirm the true target identity. For example, in one video, the MOT changed the target's ID three times in 1.4 seconds, but the re-identifier could only successfully re-identify the last correct track in real-time.

In the person-following scenario, Fig. 5.4 demonstrates a practical application of this framework. Despite minor Re-ID delays, the robot completed the task of following the target without any human intervention in all experiments.

5.1.3 Contrasting catastrofic forgetting evaluation

The evaluation procedures for the proposed framework are outlined in the following. The obtained results are presented, followed by the results from the ablation study. Finally, the findings are elucidated using saliency analysis.

Experimental setup

Similar to the previous Re-ID frameworks presented in Section 5.1.1 and Section 5.1.2, and for the same reasons, the proposed framework is validated on a custom dataset comprising 10 videos. Its performance is compared with the results obtained using the CARPE-ID method and a plain MOT approach. This evaluation focuses on two key aspects: tracking time (see Fig. 5.5a) and the number of successful re-identifications (see Fig. 5.5b). For each video, the experiments are repeated five times for each framework to ensure stable results.

The batch size and training iteration parameters for the network adaptation process are set to 16 and 18, respectively, as these values represent the optimal choice determined in the ablation study presented later. These parameters are used for both the *CARPE-ID statistical model update* and the *early training* process presented in Section 4.1.3.

In the video sequences considered, the target person changes their appearance (*e.g.*, wearing different clothing), and crucially, initial appearances are occasionally reused upon returning to the camera's FOV. This specific condition was intentionally introduced to highlight the challenge of catastrophic forgetting. Videos where such challenging situations do not occur have been intentionally avoided, as the frameworks yield identical results in simpler scenarios.

Custom Dataset Requirements A custom dataset was created since limitations similar to those described in Section 5.1.2 were encountered with publicly available datasets. Publicly available datasets were not suitable for evaluating the proposed framework due to the following typical characteristics:

- They mainly consist of security camera footage recorded in crowded environments where individuals briefly appear in the frame without significant changes in appearance.
- The videos are typically short.
- Once individuals exit the camera's FOV, they rarely return to it.

Given these characteristics, validating this framework using such datasets would not be beneficial, as SOT and MOT algorithms can already handle these simpler scenarios effectively. To demonstrate the true effectiveness of the framework, the videos must require people to repeatedly exit and re-enter the camera's FOV from different

positions. Furthermore, the target's appearance is required to change during tracking to demonstrate two critical aspects: first, the proposed framework's ability to adapt to appearance changes during tracking (robustness), and second, its capacity to autonomously learn and distinguish the target's unique characteristics. In other words, the framework must learn to recognize the target by its distinctive features, thereby actively mitigating catastrophic forgetting.

System Configuration Exactly as in the previous two validations (Section 5.1.1 and Section 5.1.2), the framework is tested on a notebook equipped with an *Intel® Core™ i9-11950H* processor and an *NVIDIA GeForce RTX 3080 Laptop* GPU. An *Intel® RealSense™ D455* camera is used for image acquisition. To handle feature extraction, an IBN-ResNet-50 network [85] pre-trained on the widely adopted MSMT17 dataset [145], following the method outlined in Ge et al. [36], is used. This pre-training process establishes the initial network's weights and configures the output feature dimension D to be 256.

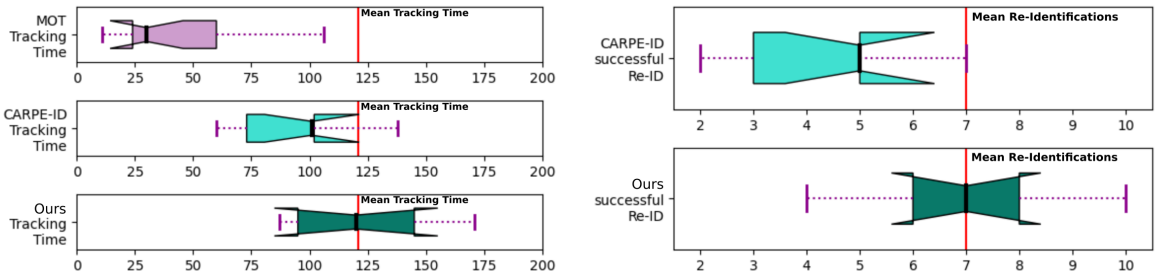
Table 5.2: Mean tracking times obtained with the ablation study. The bold numbers represent the best performance achieved among the four experiments for each batch size and iteration number trial, taking into account variations in statistical model updates and/or early training. Reprinted, with permission, from [110], © 2025 IEEE.

	model update	False	True	False	True	False	True	False	True	False	True	False	True	False	True	False	True	False	True
Early train	train iter → batch size ↓	8		10		12		14		16		18		20		22		24	
False	8	94.4	94	95.5	95.5	96.2	98.4	99.9	101.1	96.3	97.7	100.8	102.1	99	100.6	101.3	103.5	101.8	102.5
True		91.7	**99.4**	96.1	**102.6**	95.6	**113.3**	91.8	**108.9**	99.2	**114.2**	93.4	**109.9**	102.5	**108.5**	92	**117.3**	92	**117.3**
False	12	95.2	96	91.5	94.6	**105.6**	98.4	99.9	98.6	106.6	102.9	101.5	**109.3**	108.6	105.4	109.5	111.4	104.1	107.1
True		91.4	**99.3**	91.7	**113.4**	100.8	98.1	**104.4**	100.7	104.4	**108.8**	96.7	106.7	**111.1**	109.2	100	**111.7**	103.2	**119.3**
False	16	100.2	98.3	105.1	100	96.3	101.6	101.5	98.2	111	111.6	103.1	105	100.5	110.3	90.6	104.8	96.4	96.8
True		93.8	**106.4**	94.8	**116.6**	79.1	**115.2**	99	**118.8**	96.9	**115.2**	81.3	**121.8**	82.0	**118.9**	79.4	**119**	79.6	**119.8**
False	24	101.4	99.7	103.8	108.6	101.6	106.5	103.9	107.8	96.4	95.4	98.8	99.5	106.3	105.6	96.8	103.5	94.2	96.6
True		83.2	**118.4**	96.7	**116**	82.3	**119.3**	90.9	**119.6**	79.6	**120.84**	79.5	**116.1**	83.4	**119**	96.8	**116.9**	79.6	**120.1**
False	32	108.1	110.8	93.7	111.1	96.4	97.1	102.3	102	104	103.3	92	96.2	99.9	96.2	98.1	99	96.9	96
True		110.1	**118.1**	82.3	**118.7**	79.5	**119.5**	79.7	**115.4**	84.6	**118.9**	79.4	**118.6**	79.4	**119.6**	79.4	**119.2**	79.5	**118.6**

Obtained results

Fig. 5.5 presents a comparison among a SoTA MOT system[3], the CARPE-ID framework, and the proposed framework. The box plots in Fig. 5.5a and Fig. 5.5b clearly show that the proposed framework consistently outperforms both the original CARPE-ID approach and the baseline MOT system across the performance metrics. The MOT system is not directly included in Fig. 5.5b because it does not incorporate a Re-ID step; however, its tracking failures serve as the reference for determining whether CARPE-ID or the proposed framework successfully re-identifies the target.

[3]Yolov8 Tracking: `https://github.com/mikel-brostrom/yolo_tracking`



(a) Tracking time comparison.

(b) Successful Re-ID comparison.

Figure 5.5: Comparison between the experimental results of SoTA MOT algorithms: (in light purple), CARPE-ID (in light blue), and the proposed framework (in dark green); including tracking time (a) and successful Re-ID (b). The ground truth for the mean tracking time and the Re-IDs are displayed in red. The failures of the MOT algorithm are used as a reference to evaluate the successful Re-ID of CARPE-ID and the proposed framework; hence, the MOT algorithm has been excluded from comparison in (b). Reprinted, with permission, from [110], © 2025 IEEE.

The proposed framework achieves a remarkable mean tracking time of 121.8 seconds, which is nearly identical to the ground-truth mean appearance time of 122.7 seconds (the total duration during which the target remains in the camera's FOV). This results in only a negligible 0.9 seconds mean deviation in tracking accuracy. In contrast, CARPE-ID achieved a mean tracking time of 102.3 seconds, with a substantially larger mean tracking accuracy deviation of 20.4 seconds.

Furthermore, the proposed framework exhibits high reliability in handling ID switches within the MOT system, successfully re-identifying the tracked target in every case. Indeed, the proposed approach achieves a mean of 7 successful Re-IDs, matching the ground truth exactly. In contrast, CARPE-ID only achieves a mean of 5 because it fails to track the target until the end of several videos due to catastrophic forgetting.

It is important to note that the proposed framework operates in real time, as it does not rely on offline data or require the analysis of an entire dataset, which is beneficial for applications such as real-time camera streaming. This highlights its robustness, as it processes images at an online rate of 7 Hz. While a high presence of distractors in an image can hinder algorithm performance, this framework is specifically designed for middle-to-near distances between the robot and the human collaborator. This proximity typically keeps the number of distractors stable, preventing a significant drop in performance.

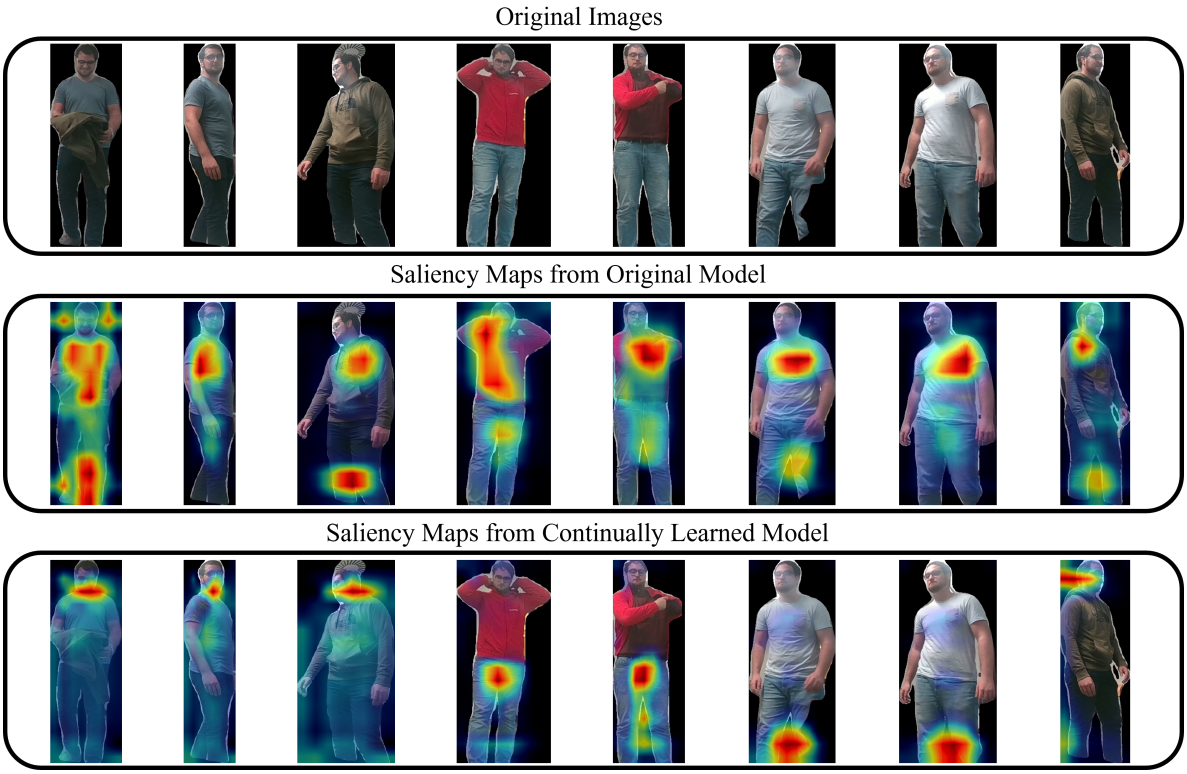


Figure 5.6: Saliency maps comparison obtained with Grad-Cam [122]. The saliencies of the original pre-trained network model (second row) are compared with the ones obtained using the weights of the continually learned model (last row). The image background is intentionally removed to reduce its influence on the extracted features, allowing the network to focus just on the people's characteristics. Reprinted, with permission, from [110], © 2025 IEEE.

Ablation study

An ablation study was conducted to determine the optimal batch size and number of training iterations for continual twin network training. This study is paramount for this application because it addresses the inherent trade-off between maximizing tracking performance and minimizing the risk of network overfitting during on-the-fly training.

The ablation study was organized using four distinct setup configurations for each parameter combination (see Section 4.1.3 for parameter descriptions):

- With the *CARPE-ID statistical model update* and with the *early training* process.
- Without the *statistical model update* but with the *early training* process.
- With the *statistical model update* but without the *early training* process.
- Without the *statistical model update* and without the *early training* process.

Consequently, for each combination of batch size and training iterations, the study involved these four configurations to compare the outcomes and understand the precise influence of each proposed feature on the overall framework.

The study systematically tested five batch sizes $(8, 12, 16, 24, 32)$ and ten training iterations $(8, 10, 12, 14, 16, 18, 20, 22, 24)$ to identify the optimal settings. This process was applied to each video in the experiments dataset, and the results were subsequently averaged and presented in Table 5.2 to eliminate redundancy and provide a consolidated view.

While several parameter combinations produced favorable results, a batch size of 16 and an iteration count of 18 were identified as the optimal parameters for this framework. Additionally, the study confirmed that both the *CARPE-ID statistical model update* and *early training* features are essential for achieving robust results.

The *CARPE-ID statistical model update* effectively improves Re-ID, particularly when the network is trained just before the target disappears from the image. In such scenarios, the network's feature output deviates from the previously established statistical model, necessitating post-training adaptation to prevent the loss of the target. This feature thus proves essential in the selected scenarios, fortifying the overall framework.

Furthermore, *early training* has proven crucial for achieving high accuracy. The value of this feature lies in enabling the feature network to train on all acquired images up to the point of target loss, and then fine-tune it with a larger number of valuable target appearances. This aligns perfectly with the *statistical model update*, as training occurs precisely when the target is lost, which can cause misalignment between the feature network's output and the previously acquired statistical model.

This dual dependency is evident in Table 5.2, where the impact of early training is heavily influenced by the presence or absence of the *statistical model update*, with most of the best and worst results observed accordingly within the early training row. These experiments provided clear guidance on the best parameters and configurations to maximize the performance of this framework.

Visualization with saliences

To visually demonstrate why the proposed approach enhances Re-ID performance, a comparison of saliency maps is conducted. These maps are obtained using two sets of weights: those from the standard feature extractor network pre-trained on the MSMT17 dataset [145], and those obtained using the continual learning approach, as illustrated in Fig. 5.6.

The well-established explainable AI framework Grad-CAM [122] is used to generate this visual output.

The comparison of saliency maps presented in the figure reinforces the thesis of network specialization. Saliency maps are generated for random images extracted from the custom experiments dataset. The figure clearly shows that the network's saliency maps with its original weights are relatively sparse and often focus on disparate image regions. In sharp contrast, the saliency maps generated by the network trained through continual learning are more specialized and tend to focus on visual cues that remain consistent across video sequences. For instance, in these experiments, as the targets frequently changed their upper-body clothing, the networks continually adapted and specialized in recognizing either their faces or their trousers/shoes.

Furthermore, incorporating the Soft Triplet loss during training serves a dual purpose: it not only compels the network to learn to recognize the most distinctive features of the target but also equips it to differentiate these features from those of the distractors effectively. This visual evidence explains why the proposed algorithm substantially outperforms the baseline model: the features learned by the continually trained network are finely tuned to focus on the most reliable and consistent visual cues for robust Re-ID.

5.2 Geometrical SLAM validations

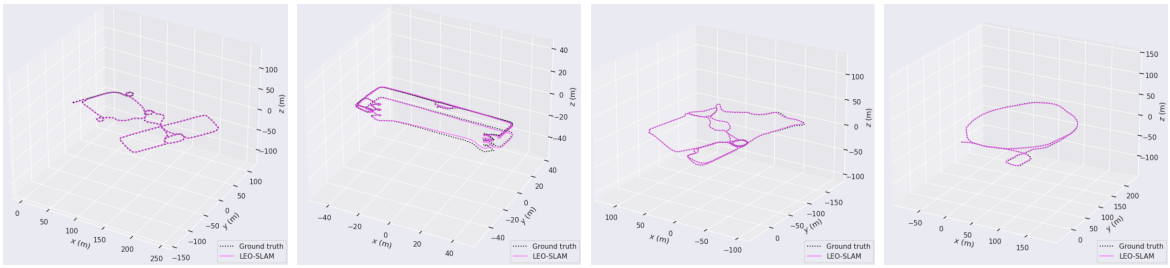
In this section, the results obtained to address the geometrical SLAM problem are presented. The results for the 3D LiDAR SLAM algorithm are reported in Section 5.2.1, while the results of the ground-aware filtering technique are presented in Section 5.2.2. Section 5.2.3 contains the comparative results of the statistical LCD proposal.

5.2.1 Submap-based 3D LiDAR SLAM with multi-level scan matching experiments

To validate the LEO-SLAM algorithm, the VBR dataset [13] is used. The proposed SLAM method is compared with two SoTA algorithms: KISS-ICP [136] and PIN-SLAM [86], as reported in Table 5.3. The proposed algorithm and KISS-ICP require only a single CPU to run, whereas PIN-SLAM requires a GPU to operate in real time. The VBR dataset is chosen because it provides a comprehensive set of sensor data and reliable ground-truth estimations. The dataset includes a sensor session containing LiDAR, IMU, and camera data. The LiDAR scan and IMU data are utilized in these

Table 5.3: Comparison of the proposed LiDAR SLAM method with SoTA approaches provided by the VBR dataset benchmark. Lower ATE and RPE values indicate more accurate trajectory estimation and, consequently, more precise map reconstruction. Reprinted, with permission, from [107], © 2025 IEEE.

Method	Metric	Colosseum	Diag	Pincio	Spagna
LEO-SLAM	ATE (m) ↓	**0.40748**	0.63319	1.019423	**0.775329**
	RPE (m) ↓	**0.33596**	**0.30345**	**0.328775**	1.027941
KISS-ICP	ATE (m) ↓	1.51690	1.39727	0.78472	1.04755
	RPE (m) ↓	0.45299	1.7908	0.48580	0.39860
PIN-SLAM	ATE (m) ↓	0.50647	**0.36240**	**0.64738**	5.68224
	RPE (m) ↓	0.49052	0.46836	0.45373	**0.35910**



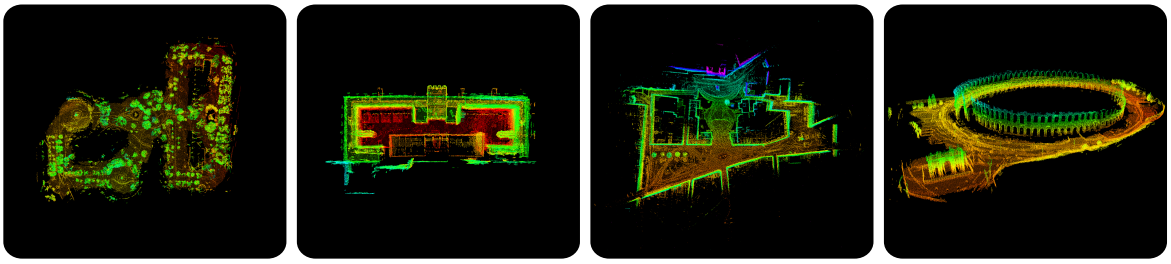
(a) Pincio session. (b) Diag session. (c) Spagna session. (d) Colosseum session.

Figure 5.7: Comparison between the trajectories generated by LEO-SLAM and the VBR ground truth across four different sessions. Reprinted, with permission, from [107], © 2025 IEEE.

experiments to provide an initial guess for the scan-to-scan (s2s) alignment. Four out of the eight sessions provided in the VBR training dataset are considered: Colosseum, Diag, Pincio, and Spagna.

The qualitative results of these experiments are shown in Fig. 5.7, which compares the estimated trajectories with the ground-truth trajectories, and in Fig. 5.8, which presents the final point cloud maps.

The accuracy metrics used are the RMSE of the ATE and the RPE, as expressed in Eq. (5.1) and Eq. (5.2), calculated on the translational component of the trajectory compared to the VBR ground truth. The EVO framework [40] is used to compute these metrics, which superimposes the estimated trajectory with the ground truth using the $SE(3)$ Umeyama alignment method [134].



(a) Pincio session. (b) Diag session. (c) Spagna session. (d) Colosseum session.

Figure 5.8: Final maps obtained using LEO-SLAM from the four analyzed VBR sessions. Reprinted, with permission, from [107], © 2025 IEEE.

$$ATE = \sqrt{\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} ||\hat{\mathbf{p}}_i - \mathbf{p}_i||^2}, \tag{5.1}$$

$$RPE = \sqrt{\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} ||(\hat{\mathbf{p}}_{i+1} - \hat{\mathbf{p}}_i) - (\mathbf{p}_{i+1} - \mathbf{p}_i)||^2}, \tag{5.2}$$

where $\hat{\mathbf{p}}_i$ denotes the estimated trajectory position at time i, $\mathbf{p}_i$ is the corresponding ground-truth position at time i, and N is the total number of estimated trajectory positions.

In the experiments, the voxelization leaf size of the small-GICP algorithm was set to 0.25 meters to improve processing efficiency while preserving relevant environmental features. The validation results are presented in Table 5.3, where the LEO-SLAM algorithm is compared against several algorithmic results provided by the VBR benchmark.

Discussion

In the experiments, outdoor environments are primarily considered, except for the *Diag* session, which transitions from outdoor to indoor environments, thereby increasing mapping difficulty.

As shown in Table 5.3, LEO-SLAM achieves improved mapping accuracy, enabling better map usability for navigation. The proposed algorithm outperforms current SoTA methods in RPE estimation across almost all sequences and achieves better ATE results in 2 out of 4 sequences. ATE measures the global accuracy of the estimated trajectory with respect to the entire ground truth trajectory, whereas RPE evaluates the between-node estimation accuracy, *i.e.*, the displacement error between consecutive nodes with

respect to the ground truth displacements. As shown in Table 5.3, ATE and RPE are not strictly correlated; thus, a better ATE does not necessarily imply a better RPE. This is because a trajectory with smaller between-node error (RPE) may still accumulate significant drift, leading to a higher overall ATE.

LEO-SLAM runs in real time on CPU-based platforms, with an average computation time of 185 ms per full processing loop (see Algorithm 2 for reference).

The introduction of submap keyframes and submap-to-submap alignment improves accuracy by yielding denser and richer point clouds. This is particularly beneficial for LiDAR sensors with fewer channels, a common scenario in robotics. In such cases, a keyframe consisting of a single point cloud may lack sufficient information to robustly handle potential alignment errors. The reconstructed maps generated by the proposed algorithm are sufficiently smooth for subsequent planning and navigation tasks or for visualization.

One limitation of LEO-SLAM is the potential drift introduced in featureless environments, such as long corridors. This is a common issue in pure LiDAR odometry SLAM algorithms and is difficult to correct solely through loop closure. However, using external odometry or IMU data as an initial guess helps mitigate this problem.

Planned improvements to address this limitation include the integration of multi-sensory solutions, such as the deployment of visual cues to refine odometry using cameras, and the autonomous adaptation of parameters based on the environmental context, aiming to reduce drift in low-feature environments. This work also encounters challenges on highly dynamic vehicles, where point clouds become distorted due to rapid motions, particularly during high-speed rotations. A potential solution is the application of point cloud deskewing, as proposed in Vizzo et al. [136].

5.2.2 Point cloud ground-aware intensity filtering experiments

Table 5.4: The RMSE results from the experiments on the custom dataset using the selected frameworks are presented, considering the following three scenarios: (i) no point cloud filter applied, (ii) applying only the intensity thresholding filter, and (iii) applying the intensity thresholding filter with additional ground awareness.

Framework	Filtering Mode	Translation Error (TE) [m]	Orientation Error (OE) [rad]	Translation Vector Error (TVE) [m] (x y z)
Kiss-ICP [135]	No Filter	22.529	0.270	18.848 12.129 2.298
	Intensity	2.585	0.100	**0.278** 1.152 2.297
	Ground + Intensity	**1.112**	**0.075**	0.478 **0.986 0.187**
DLO [18]	No Filter	**0.410**	0.068	0.164 **0.109** 0.359
	Intensity	2.395	0.098	0.179 0.408 1.979
	Ground + Intensity	0.428	**0.0637**	**0.156** 0.397 **0.031**

To validate the ground-aware intensity filter implementation, SoTA LiDAR odom-

etry algorithms such as KISS-ICP [135] and DLO [18] are used. Due to the sparsity of the point cloud in the proposed scenario, it was decided to remove one of the two downsampling filters used in KISS-ICP before scan matching, as the authors suggested, to obtain better results.

A custom dataset was created in an indoor scenario with a high number of windows, leading to numerous erroneous reflections detected by the LiDAR. The experimental setup consists of a quadruped robot, the Unitree B1, equipped with a Velodyne VLP-16 LiDAR. To filter out low-intensity points, a filtered range, $\psi^* = [0, 15]$, is selected from the full intensity range $\psi = [0, 255]$. The terrain is mostly flat, with some ramps along the paths. A custom dataset is created because, to the best of the author's knowledge, commonly used public datasets, such as the KITTI dataset [38], do not feature a significant number of erroneous reflections caused by intensity variations. As a result, these datasets are unsuitable for evaluating the proposed filtering methods.

Additionally, this dataset does not include a ground-truth trajectory to calculate the ATE or RPE, unlike the previous validation in Section 5.2.1. However, for each path, the starting and finishing poses are aligned, enabling the final odometry error to be computed with minimal uncertainty. Using the initial and final robot poses, the following three types of mistakes are evaluated:

- *TE* [m]: The Euclidean distance between the origins of the final and initial frames.
- *OE* [rad]: The rotation angle between the orientations of the final and initial frames, expressed in axis-angle form.
- *TVE* [m]: The translational error along the x, y, and z axes.

To validate the performance, the RMSE of these errors is computed across the following three scenarios:

1. Original LiDAR output, containing incorrect reflections.
2. Filtered LiDAR output, obtained by removing points with intensity values outside the threshold interval ψ^*.
3. Filtered LiDAR output with intensity filtering within ψ^*, augmented with the additional ground awareness as described in Section 4.2.2.

The RMSE results obtained in these experimental scenarios are reported in Table 5.4.

The TVE is computed by subtracting the initial position ${}^B\mathbf{t}_i$ from the final position ${}^B\mathbf{t}_f$. The TE is the Euclidean norm of TVE. In contrast, the OE was calculated by

determining the relative quaternion orientation ${}^{B}\mathbf{q}_r = {}^{B}\mathbf{q}_i^{-1} * {}^{B}\mathbf{q}_f$ and converting the resulting angle into axis-angle form. Finally, the RMSE is computed for each scenario using these error values.

It is important to note that only the *naive intensity filter* presented in Section 4.2.2 is used, since the alternative solution proved unusable due to the high point cloud sparsity.

The proposed intensity filter solution has proven to be robust and stable, offering improvements to SoTA odometry systems. The experimental results are evaluated in the following, and the two filter implementations are subsequently compared to provide insights into which solution is more suitable.

Ground-aware filter evaluation

The results reported in Table 5.4 demonstrate that this filtering method is beneficial and essential for the first framework evaluated, *i.e.*, KISS-ICP [135]. In contrast, the second framework, *i.e.*, DLO [18], proved to be robust enough to handle erroneous reflections, yielding comparable results in both unfiltered and ground-aware filtered scenarios.

It has been observed that KISS-ICP is sensitive to erroneous reflections, which consistently impact its performance. While introducing a standard intensity filter significantly improved the results, it still produced significant errors along the z-axis. This is because, as previously stated, the standard intensity filter removes ground points, which are crucial for the ICP algorithm to recover errors along the z-axis accurately.

An interesting observation is that in KISS-ICP, while the ground-aware filter performed better in almost all metrics, the x-axis component of the TVE showed slightly higher errors compared to the solution using only intensity filtering. This is due to ICP matching errors between ground-level range points in the current scan and those in the previous point cloud. LiDAR sensors emit infrared rays in rows (or channels), with each channel projecting a ring onto the ground, as shown in the right image of Fig. 4.11. Since the point cloud is LiDAR-centric, these projected rings move along with the LiDAR, making them difficult to distinguish in consecutive scans, as the robot remains centered within them. As the robot typically moves along the x-axis of the LiDAR, the ICP algorithm tends to match the rings of two consecutive point clouds, introducing small errors along the x-axis. These errors are partially compensated for by matching points on walls and obstacles, but the remaining errors accumulate over time, leading to slightly worse results when the ground points are preserved.

DLO, on the other hand, proved to be robust to erroneous reflections, yielding similar

results in both the unfiltered and ground-aware scenarios. However, an interesting consideration arises: if odometry is used to create a point cloud map, the presence of erroneous reflections will affect the final results, resulting in an erroneous map. This is clearly demonstrated in Fig. 2.2, where the comparison between the filtered and unfiltered point clouds in high-reflection environments shows that the unfiltered point cloud becomes essentially unusable. Additionally, when a standard intensity filter is applied to refine the map, the absence of ground points in the filtered point cloud negatively impacts the accuracy of ICP matching along the LiDAR z-axis, as evidenced in Table 5.4, resulting in degraded performance in the final application. This highlights that, in mapping scenarios, this ground-aware filter is essential for producing an accurate final map without compromising odometry performance.

Lastly, an interesting observation made during the experiments was that DLO experienced transient odometry errors in the unfiltered scenario, whereas its performance was smoother in the ground-aware filtering scenario. DLO seems to be able to recover these accumulated errors, at least partially, by performing scan matching between the most recent point cloud scan and a piece of the entire accumulated map when places are already seen and revisited. This hypothesis could be further supported by using more advanced metrics, such as the ATE and RPE, throughout the entire trajectory. However, due to the absence of ground truth in the experiments (caused by the lack of motion capture systems), the authors are unable to fully validate this hypothesis and leave it open for future studies and deeper analysis.

Solutions comparison

Although only one solution was experimentally tested, a conceptual comparison of both proposed ground-aware intensity filter implementations is provided, highlighting their respective pros and cons.

The first solution, referred to hereafter as the *naive solution*, is straightforward to implement. However, its main issue is that it retains all the erroneously reflected points below the robot's height. While many of these retained points may still correspond to the ground, facilitating their merging with actual ground points, this correspondence is not guaranteed.

The second solution, known as the *normal solution*, is more advanced and can distinguish and retain additional surfaces (like low tables) by comparing the LiDAR's homogeneous transformation with the point cloud normals. However, this comes at the cost of increased computational time and complexity. Furthermore, it still faces

challenges with misreflected points that are parallel to the LiDAR's base. Additionally, the normal computation is not robust in cases of LiDAR sparsity, which is common in robots using LiDAR sensors with fewer channels (*e.g.*, 16 or 32). This limitation makes the *normal solution* less effective in such situations, a frequent scenario in robotic applications.

The *naive solution* has a mathematical complexity of $\mathcal{O}(n)$, as it processes each point only once (a constant time operation per point). In contrast, the *normal solution* has a complexity of $\mathcal{O}(n^2)$. This is because it first computes the normal for each point (which takes $\mathcal{O}(n)$ time due to neighborhood searches), and then checks both the intensity and the cosine of the angle between the LiDAR's z-axis vector and the point's normal (which adds $\mathcal{O}(n)$ complexity).

In summary, the *naive solution* is faster, works well when the point cloud is sparse, and is easier to implement. On the other hand, the *normal solution* is computationally more expensive due to the norm computation. While it offers the advantage of maintaining xy-plane-parallel surfaces that are above the ground, this feature is less common in typical scenarios due to the generally low height of robots. However, such situations can still occur, making the *normal solution* beneficial in those specific cases.

The *naive solution* is generally preferred because delays are harmful in LiDAR odometry and SLAM systems. This choice becomes even more beneficial in outdoor environments, where the ground may not be perfectly parallel to the LiDAR, creating difficulties for the normal-based solution. Additionally, in scenarios where the robot is in a pit, the surrounding ground may be higher than the robot's height, which compromises the effectiveness of both solutions. However, there are two important considerations to note for outdoor use. First, when the ground is not parallel to the robot's base, the incidence angle of the LiDAR should be large enough to ensure the points fall outside the intensity threshold ψ^*. Secondly, in outdoor environments, the ground material is often opaque, which helps mitigate issues with incorrect reflections, especially when the incidence angle allows for proper point detection.

To address the limitations of these single-sensor solutions, a multi-sensor setup, such as integrating cameras and LiDAR, could be employed. Alternatively, intensity and semantic information could be used to more accurately segment true ground points. However, in robots with constrained resources, using these methods would lead to higher power and computational demands, which could degrade the performance of odometry and SLAM. Therefore, simpler yet effective solutions, like the one presented in this paper, remain the preferred choice.

Table 5.5: Comparison of maximum recall values, under the constraint of perfect precision (100%), between SC++ and GSC for each selected sequence.

Sequence	SC++ [54]	GSC Height	GSC Huber
KITTI 02	57.8%	59.4%	**69.7%**
KITTI 05	81.2%	82.6%	**83.1%**
KITTI 00	80.8%	81.2%	**83.7%**
VBR Pincio	36.1%	42.3%	**49.3%**
VBR Colosseo	**18.4%**	16.2%	14.5%

5.2.3 Gaussian Scan Context experiments

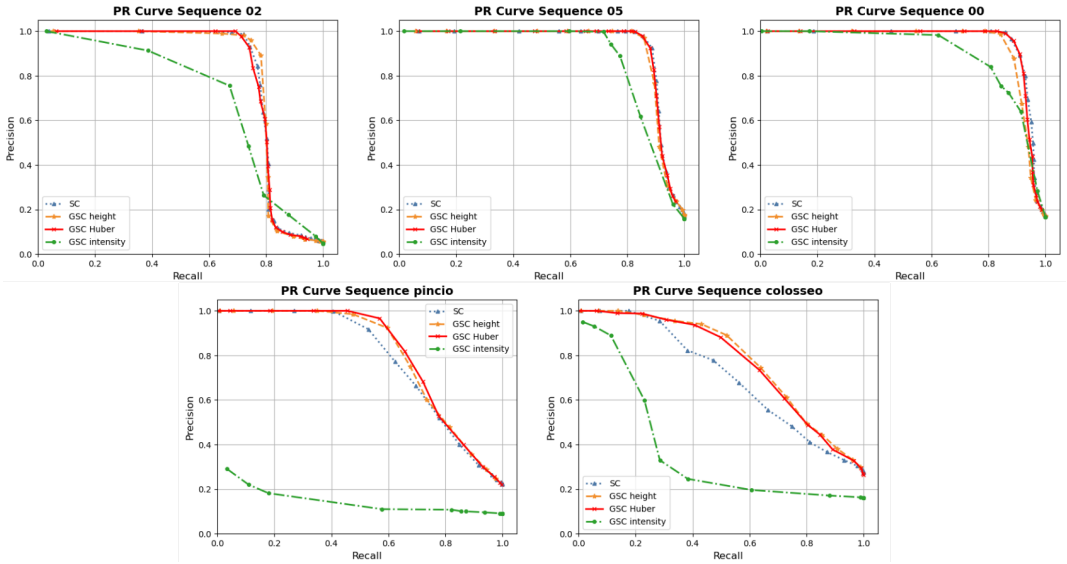


Figure 5.9: Precision–Recall curves obtained from the experimental results are presented. These graphs compare the SC++ baseline with three GSC variants. Each GSC variant computes the matrix entries using Eq. (4.28), but with different inputs: GSC height (orange) uses the z values, GSC Huber (red) applies Huber-weighted z values, and GSC intensity uses the i values.

To validate the performance of the GSC across diverse environments, two datasets are used for evaluation: the well-known KITTI dataset [38] and the VBR dataset [13], collected in Rome. Specifically, sequences `00`, `02`, and `05` from KITTI are selected, as well as `Pincio` and `Colosseo` from VBR, due to their high occurrence of loop closures.

The parameters for the Gaussian context are fixed at $\alpha = 2$ and $\delta = 1$. Both SC++ and GSC are evaluated on the keyframes, where each keyframe corresponds to a point cloud with a ground-truth pose expressed relative to a common map frame.

Evaluation metrics and conditions

To assess performance, Precision (P) and Recall (R) are computed for both algorithms across each sequence. Ground-truth loop closures were identified based on the Euclidean distance d_{ij} between the poses of two keyframes i and j. The distance threshold is set as $d_t = 5\,\mathrm{m}$, and the following conditions are defined to determine the correctness of a detected loop closure:

- A loop closure is detected between the current keyframe $\mathbf{k}_n$ and a previously observed keyframe $\mathbf{k}_j$, where $j \in \{1, \ldots, \mathrm{N} - \xi\}$. The parameter $\xi \in \mathbb{N}$ excludes the most recent ξ keyframes to prevent trivial and near-trivial matches.
- The Euclidean distance between the current keyframe $\mathbf{k}_n$ and the matched keyframe $\mathbf{k}_j$ is below the threshold, *i.e.*, $d_{nj} \leq d_t$, indicating that they correspond to the same physical location. This condition is meaningful only if condition (1) holds.
- There exists a ground-truth match $\mathbf{k}_p$ among the previous keyframes such that $d_{np} \leq d_t$. This ensures that a valid loop closure is possible for the current keyframe.

Using these conditions, each keyframe match is classified as: (i) True Positive (TP) (True Positive) if both conditions (1) and (2) are satisfied; (ii) False Positive (FP) (False Positive) when Condition (1) is satisfied, but condition (2) is not; (iii) False Negative (FN) (False Negative) if Condition (1) is not satisfied, but condition (3) is; and (iv) True Negative (TN) (True Negative) in all remaining cases.

This evaluation framework enables a rigorous and interpretable comparison between SC++ and GSC across diverse environmental conditions and sensor data characteristics.

Based on the classification results, P and R are computed as follows:

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad \mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{5.3}$$

These metrics were evaluated for each validation sequence by varying the Re-ID threshold τ over a range from 0 to 1. This ablation study enables the analysis of how the algorithms' performance evolves with respect to the threshold, offering insights into their sensitivity to parameter tuning.

The resulting Precision-Recall (PR) curves are presented in Fig. 5.9. These curves illustrate the robustness of each algorithm to variations in the threshold τ. In particular, the Area Under the Curve (AUC) of the PR curve serves as a measure of overall performance: the closer the AUC is to 1, the more consistently high-performing the

algorithm remains across threshold values. In the ideal case, where AUC = 1, the algorithm achieves perfect performance regardless of the selected threshold.

GSC variants and results

The performance of SC++ is compared with three variants of the proposed method:

- *GSC Height*: computes the standard mean and standard deviation of point heights, then uses Eq. (4.28) to populate the Gaussian context entries.
- *GSC Huber Height*: computes the mean and standard deviation of point heights using Huber weighting (as defined in Eq. (4.32) and Eq. (4.33)) and applies Eq. (4.28) to obtain the context entries.
- *GSC Intensity*: computes the mean and standard deviation of point intensity values and applies a modified version of Eq. (4.28) adapted for intensity instead of height.

In the LCD problem, the primary objective is to maximize both precision and recall. However, achieving 100% precision (*i.e.*, FP = 0) is particularly critical, as false positives can significantly degrade the accuracy of downstream SLAM optimization. Therefore, for each sequence, the maximum recall achieved is reported under the constraint of perfect precision. These results are summarized in Table 5.5. The GSC Intensity variant is excluded from this table, as it does not consistently achieve 100% precision and performs substantially worse than the other methods.

For computational performance, the methods are benchmarked on a machine equipped with an *Intel® Core™ i9-11950H* processor. On average, SC++ requires 54 ms per keyframe to generate the SC++ and execute the loop search, while the proposed GSC Huber variant requires 67 ms. This additional computational cost is justified by its improved robustness, and the overhead can be mitigated through optimization strategies, such as parallelized computation of means and covariances, which substantially reduce computation time for large point clouds.

Discussion

The results presented in Tab. 5.5 and Fig. 5.9 demonstrate that the proposed GSC variants consistently outperform the original SC++ [54] in most evaluated scenarios.

For the KITTI dataset, the PR curves of SC++ and GSC appear qualitatively similar across sequences, suggesting that both methods maintain robustness to variations

in the Re-ID threshold. However, a closer inspection of the maximum recall achieved at 100% precision (Tab. 5.5) reveals a more nuanced outcome: GSC, particularly the Huber-weighted height variant, significantly surpasses SC++. In sequence `02`, GSC Huber achieves nearly a 12% improvement in recall. This indicates that incorporating all points within each context bin and regularizing their statistical representation using the Huber weight produces more reliable and discriminative context descriptors. The Huber function's capacity to suppress the influence of outliers likely contributes to this substantial gain.

The advantages of the proposed approach are even more evident in the VBR, which contains more challenging and noisier real-world urban scenes. As shown in Fig. 5.9, the GSC variants exhibit markedly greater robustness to threshold variation than SC++. This suggests that statistical modeling, particularly with robust estimators such as the Huber weight, is highly effective in environments where LiDAR noise and dynamic elements are more prevalent. In the `Pincio` sequence, the proposed method delivers an improvement of over 13% in recall at perfect precision, underlining its strength in high-ambiguity contexts.

In contrast, in the `Colosseo` sequence, GSC performs slightly worse than SC++ (2.2% standard GSC and 3.9% less recall for Huber GSC). This is attributed to the scene's repetitive geometric and structural layout, which may hinder statistical estimation or reduce the distinctiveness of Gaussian descriptors. Nonetheless, the overall trend still favors GSC, particularly in terms of robustness and recall consistency across diverse environments.

Although the GSC Intensity variant does not match the performance of its height-based counterparts, it is included in the evaluation to explore concepts introduced in ISC [137]. ISC previously demonstrated gains over the original SC++ [55] by leveraging point intensity information. However, the results suggest that directly modeling intensity values within a Gaussian framework results in poor performance. This may be due to the inherently high variance and unreliability of intensity readings, which are influenced by factors such as surface reflectivity, angle of incidence, and lighting conditions. These factors cause significant fluctuations in intensity values within a single bin, undermining the effectiveness of statistical modeling.

Finally, in terms of computational efficiency, the proposed method incurs only a modest overhead compared to SC++. On the testing platform (*Intel® Core™ i9-11950H*), GSC with Huber weighting requires approximately 67 ms per keyframe, compared to 54 ms for SC++. The performance improvements justify the additional processing time and can be mitigated by efficiently parallelizing the mean and covariance

computations. Consequently, GSC can serve as a drop-in replacement for SC++ in SLAM systems requiring robust LCD, without compromising real-time performance.

5.3 Robot environmental contextual understanding and interaction experiments

This section reports the results obtained with the methods presented in Section 4.3. While Section 5.1 focuses on testing contextual awareness in Human-Robot collaborative scenarios, this section focuses on the robot's environmental understanding capabilities for recognizing which objects populate the surroundings and how it can interact with them.

5.3.1 Semantic mapping experiments

The experiments are performed both in simulation and using a real robot in a laboratory environment. The experimental setup is the same for both: some chosen objects are randomly positioned in the experiment area, and the robot, following a predefined path, maps the objects of interest it encounters. This strategy is chosen because the objective is to validate the artifacts mapping during an application, for example, during a patrol. In other application scenarios, *e.g.*, search and rescue, the proposed framework could run in parallel with an exploration algorithm, and the robot could trigger the exploration module whenever an object of interest is encountered to obtain a precise localization.

In the experiments, the data fusion approach is compared with mono-sensor applications (*i.e.*, using only an RGB-D camera or only the LiDAR) to demonstrate that data fusion significantly improves detection accuracy and decreases errors. For each environment setup, the experiments are repeated three times, once for each sensor configuration: only camera, only LiDAR, and both.

This work focuses only on semantic mapping and does not account for the robot localization, which is assumed to be given. Additional mapping errors resulting from localization are not considered in the final evaluation, even if they negatively affect the application. Moreover, it is important to note that quadrupedal robots' movements are jerky, which can affect the sensors.

The parameters min_c, acc_c, and max_c of Eq. (4.36) are set to 0.3, 4, and 6, respectively, based on the camera hardware information provided by the camera vendor (Intel Realsense).

The final validation performance is measured as the number of correctly identified objects divided by the total number of objects. Also, the number of correctly detected objects relative to the total number of detections is evaluated.

The object is considered *found* if the difference between the estimated and true positions is less than the true object radius and the associated class label is correct. The errors are categorized into duplicated objects, incorrect localization, and classification. Duplications occur when multiple artifacts are mapped to a single physical object. They could be caused by wrong artifact radius computation due to occlusions or distinct points of view (*i.e.*, detected from different perspectives, such as front and behind). Localization is considered incorrect if the artifact's estimated position is outside the real object's shape, and classification is erroneous if the artifact's class label is wrong.

For the simulation, the Whole-body Locomotion Framework [102] is used on a notebook with an *Intel® Core™ i9-11950H* processor and an *NVIDIA Geforce RTX 3080 Laptop* GPU. In the real scenario, a Unitree Go1[4] quadrupedal robot equipped with a RoboSense RS-Helios16 LiDAR[5], an *Intel® RealSense™ D455*[6] and three Nvidia Jetson[7] (two Jetson Nano 4GB and one Nvidia Xavier NX) are used for the evaluation. The experiments are performed using the instance segmentation algorithms Yolact++ [12] and YolactEdge [69] trained on the COCO [68] dataset.

Simulation experiments

Table 5.6: Detection results of the simulation and real experiments. Reprinted, with permission, from [109], © 2023 IEEE.

	Simulation			Real		
	Camera	*LiDAR*	*Fusion*	*Camera*	*LiDAR*	*Fusion*
Correct detection	386	391	**416**	86	81	**99**
Wrong localization	12	13	**7**	10	14	**2**
Duplication	19	24	**15**	10	14	**6**
Wrong classification	**0**	**0**	**0**	7	11	**6**
Total detections	417	428	433	113	120	113
Total objects	422			101		

The Gazebo simulator[8] is utilized to simulate the robot's behavior in two distinct

[4]Unitree Go1: `https://www.unitree.com/en/go1/`
[5]RoboSense RS-Helios16: `https://www.robosense.ai/en/rslidar/RS-Helios`
[6]*Intel® RealSense™ D455*: `https://www.intelrealsense.com/depth-camera-d455/`
[7]Nvidia Jetson: `https://www.nvidia.com/it-it/autonomous-machines/embedded-systems/`
[8]Gazebo simulator: `https://gazebosim.org/home`

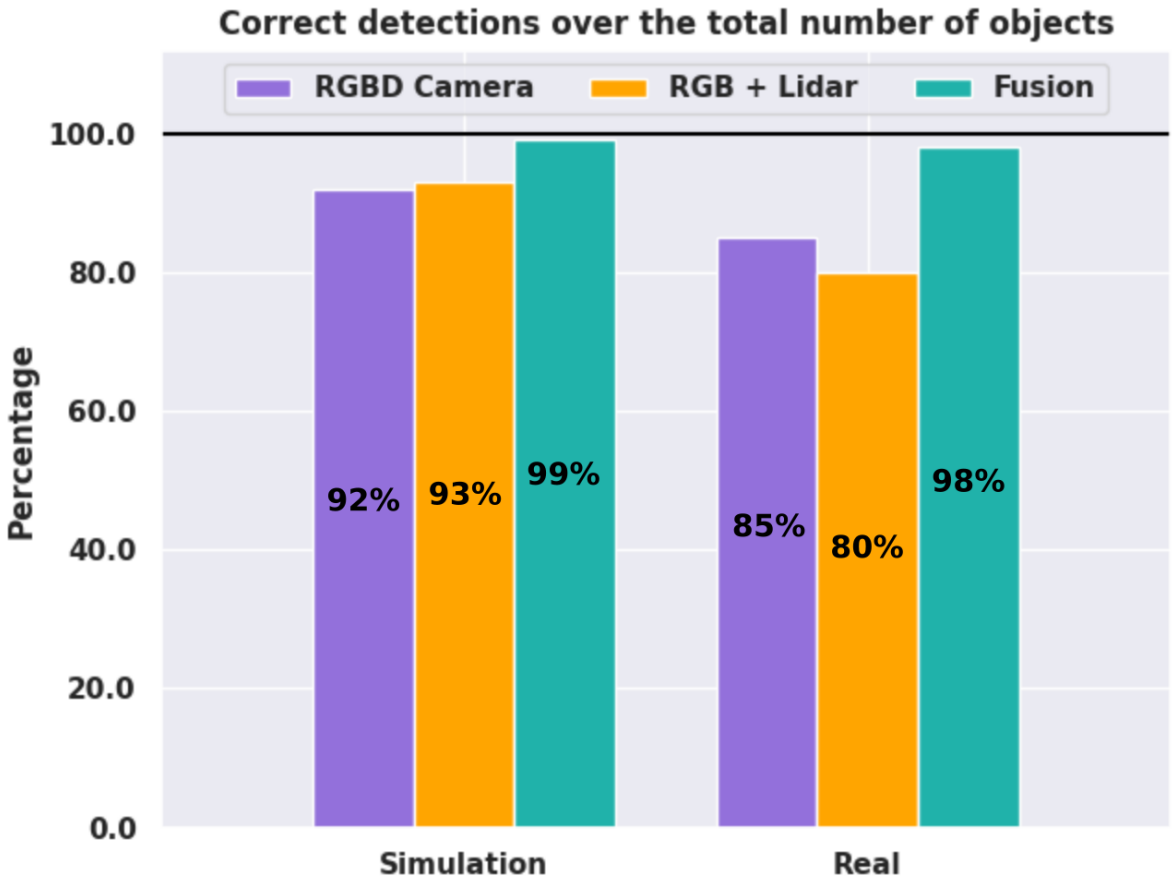


Figure 5.10: Percentage of the correctly mapped and labelled objects concerning the total number of objects on the scene. On the left are the simulation results; on the right are the real experiments. Each block contains three histograms representing the three *sensor configurations* used during the experiments: only RGB-D camera, RGB plus LiDAR, and RGB-D and LiDAR. Reprinted, with permission, from [109], © 2023 IEEE.

environments: the Office world[9] and the Maze world. In each environment, a predefined number of objects are randomly positioned at every iteration. The objects chosen for the simulation evaluation are *vase*, *couch*, *plant*, and *person*. Specifically, the Office world contains 5 vases, 12 couches, 6 plants, and 11 people, while the Maze world contains 15 vases, 13 couches, 12 plants, and 12 people. The robot's path is predetermined by a set of randomly chosen waypoints on the map. In total, 10 experiments were conducted for each *sensor configuration* (5 per environment, with different setups), yielding 30 experiments.

The results of the simulation experiments are shown in the left panel of Fig. 5.10 as

[9]Clearpath robotics worlds: `https://github.com/clearpathrobotics/cpr_gazebo/tree/noetic-devel/cpr_office_gazebo`

the number of correctly detected objects. Specifically, across the three ordered *sensor configurations* (*i.e.*, only camera, only LiDAR, and both), the percentages of correctly localized and classified objects obtained were 92%, 93%, and 99%, respectively.

Moreover, the distribution of total detections is shown in the left column of Table 5.6 and in the top row of Fig. 5.11 for the simulation experiment. Among all the detections produced, considering the three *sensor configurations* in order, 92%, 91%, and 95% were correct, while the remaining 8%, 9%, and 5% were wrong.

The farthest object correctly detected during the camera-LiDAR sensor fusion experiments in simulation was at 15.47 m from the robot, while the nearest was at 1.23 m.

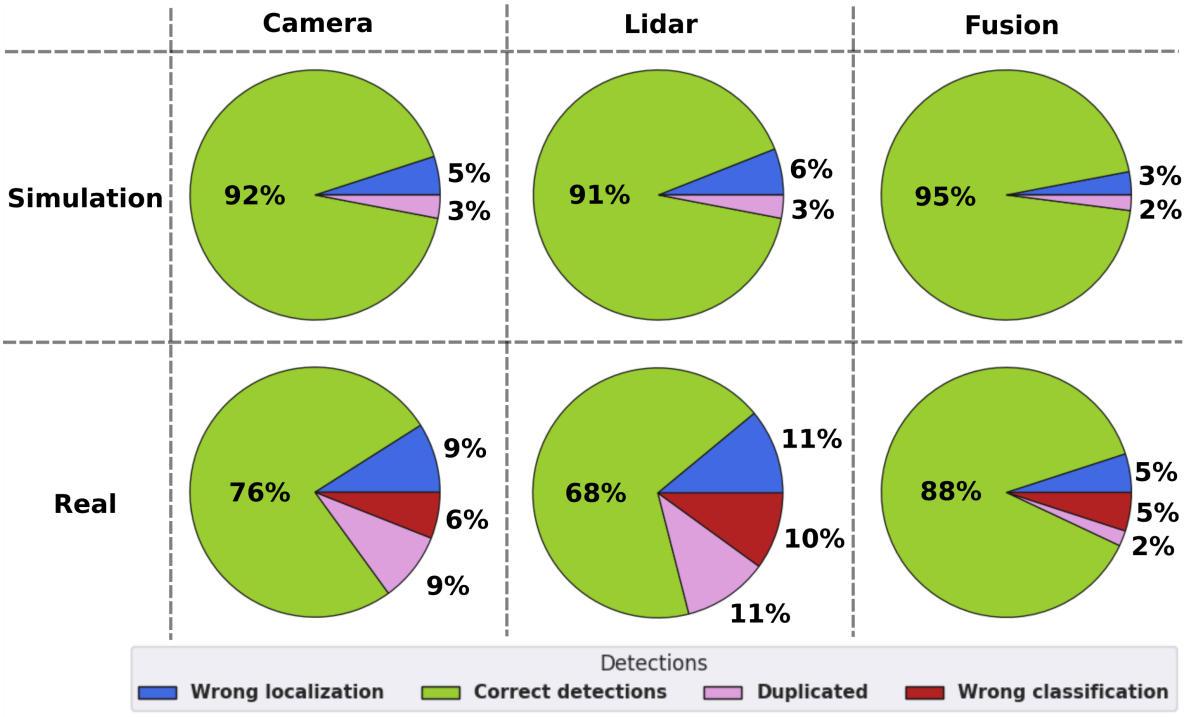


Figure 5.11: Distribution of correctly and wrongly detected artifacts among the total generated detections. The pie charts show the distribution of correctly detected artifacts (green), doubled objects (blue), wrongly localized (pink), and wrongly classified (red). The top row shows the simulation results, while the bottom row shows the real ones. For each row, experiments are divided into three columns depending on the *sensors configuration* used: only camera, only LiDAR, or both. Reprinted, with permission, from [109], © 2023 IEEE.

Laboratory experiments

The real-world experiments were conducted in a laboratory setting, with two scenarios: a one-room environment and a full-floor environment in which the robot could move

through corridors. In these environments, umbrellas, chairs, cabinets, backpacks, and TVs were arranged in varying numbers. For each *sensor configuration*, 6 experiments were conducted (3 per environment), for a total of 18. For each trial, the objects were randomly moved, and the illumination was changed, *i.e.*, switching off lights or closing shutters, to introduce environmental variability.

The results of the laboratory experiments are shown in the right panel of Fig. 5.10 as correctly detected objects. Specifically, considering the three *sensor configurations* in order (*i.e.*, only RGB-D camera, only RGB plus LiDAR, and both), the percentage of correctly localized and classified objects obtained was 85%, 80%, and 98%, respectively.

Moreover, the distribution of detections is shown in the right column of Table 5.6 and in the bottom part of Fig. 5.11 for the real experiment. Among all the detections produced, 76%, 68%, and 88% were correct, while the remaining 24%, 32%, and 12% of them were wrong.

The farthest object correctly detected during the camera-LiDAR sensor fusion experiments was at a distance of 10.37 m from the robot, while the nearest was at 0.98 m.

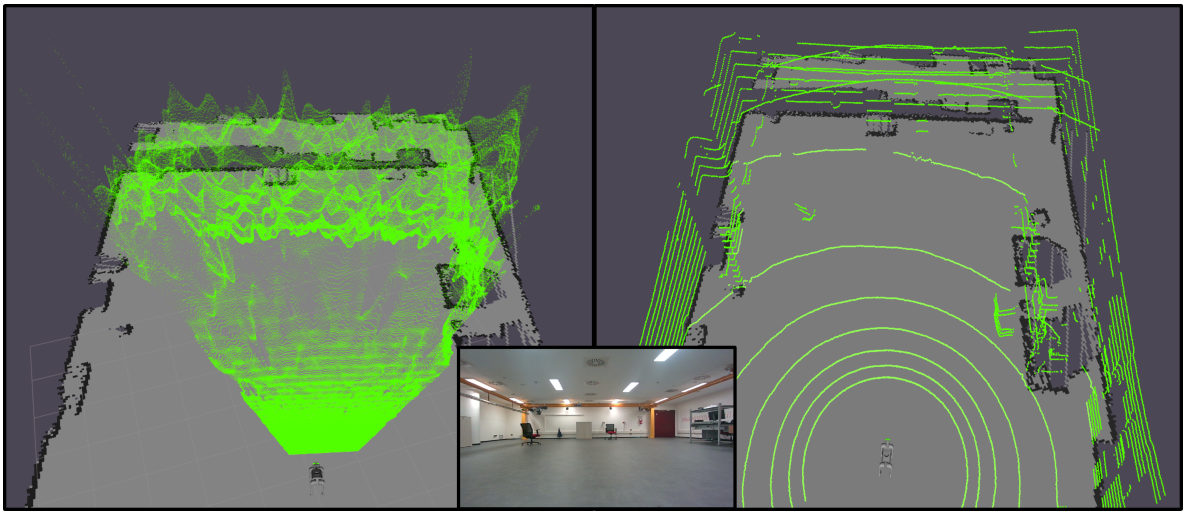


Figure 5.12: Qualitative comparison between RGB-D camera (left image) and LiDAR (right image) point cloud detections at an approximate distance of 10 m from the wall. At the bottom center is a representation of the scene taken with the robot camera at that instant. At large distances, camera data are noisier and less accurate than the LiDAR one. Still, at short distances, cameras provide a denser, more accurate point cloud, whereas LiDAR data are sparser. From this comparison, it can be deduced that a visual-LiDAR sensor fusion can enhance semantic mapping. Reprinted, with permission, from [109], © 2023 IEEE.

Using semantic mapping for autonomous loco-manipulation

A sample experiment demonstrating how to use such semantic maps has been performed in simulation using the setup described above. In the experiments, the robot is asked to bring an object to one of the people in the simulation (*e.g.*, bring a bottle to the person with id 1). The experiment is repeated for different world configurations in which people and the objects are randomly moved (see an example in the 6^{th} block of Fig. 4.17). The proposed pipeline always completed the task following the steps presented in Section 4.3.1.

Discussion

The first thing to point out is that the farthest distances of the detected objects were greater than 10 m both in simulation and in real experiments. This distance is chosen to show a qualitative comparison between the LiDAR and RGB-D measurements in Fig. 5.12. The figure qualitatively upholds the thesis that a LiDAR sensor, along with the camera, is necessary to improve semantic mapping and, in general, other detection algorithms in wide areas.

Moreover, the results obtained from the experiments clearly show that in this framework, the use of both sensors improves the robustness of the application and decreases the detection errors. These improvements are less evident in a simulation environment, where almost ideal sensors are used (*i.e.*, the noise representation is not as realistic as in Fig. 5.12), but they significantly impact real scenarios where sensor noise is more prevalent.

The LiDAR can map far obstacles precisely, while the camera introduces many errors at high distances. If only the camera is adopted, one solution to avoid erroneous depth measurements could be to not consider the depth information outside the accurate range guaranteed by the device specifications. However, by doing this, the robot could miss some artifacts if it does not get close enough to them.

The camera, by providing more information at near distances with respect to the LiDAR, yields more precise centroid computations because it typically has fewer outliers than the LiDAR. LiDAR outliers can be caused by wrong camera-LiDAR pose calibration and time synchronization, which are essential for these applications, especially when the robot moves fast. In contrast, with RGB-D cameras, the depth and the RGB images are synchronized in time and can be spatially superimposed almost exactly.

It is important to notice that wrong classification errors result from erroneous classifications in the pre-trained instance segmentation neural network, which can be caused

by illumination, reflections, or other environmental conditions. They are considered here because the image inference is a module of the proposed pipeline, but such errors can be decreased using more powerful neural networks.

Chapter 6

conclusion

The main remarks and demonstrations acquired during the PhD journey are reported in this concluding thesis section. As for the previous sections, the conclusive remarks are distributed in different sections. Section 6.1 reports the conclusions obtained from the Re-ID methods developed, validated, and exposed in this thesis. Environmental understanding conclusions are analyzed and reported in Section 6.2 for the geometric mapping and environment representations and in Section 6.3 for the semantic understanding that enhances the robotic knowledge, trying to mimic how human beings perceive the environment.

6.1 Re-identification for Human-Robot Collaboration

The works presented in Re-ID all have the same objective: to provide a robust Re-ID framework that can be used by a robot to perform HRC tasks. Robots can employ this capability to co-exist with different people in an environment, moving beyond their role as passive workers to actively understand the roles of the people around them and act accordingly.

The first work, the FollowMe [111], is a robust framework for following a target person by a mobile robot. It is mainly based on visual Re-ID and gesture detection. The experiments confirmed that visual human Re-ID is a strong feature to perform person-following tasks and suggested that this ability is crucial for other human-centered applications. Using a simple and non-invasive RGB-D camera, the robot can efficiently track a selected person, simplifying HRC. The FollowMe framework is easily customizable to a target person, avoiding mismatches or ID switches with other persons (*i.e.*, the distractors), even when two persons are similarly dressed.

Despite this, during experiments, some limitations were faced. In specific light conditions, *i.e.*, when a strong light is pointed towards the camera, the detection or Re-ID module lacks robustness. This limitation is also connected to the specific camera hardware. Also, Re-ID is strictly correlated to online calibration, *i.e.*, if the calibration is not properly finalized, the further recognition of the target could be less robust; the robot should see the person at different distances and in different configurations during calibration, simulating as much as possible the human motion made at inference time.

To solve some of these limitations, a framework for person Re-ID that can be adapted to different HRI scenarios, namely CARPE-ID [112], has been developed. This pipeline, similarly to FollowMe, consists of a detection and preliminary person-tracking algorithm, a feature extraction network for appearance representation, and a Re-ID structure based on statistical distance. In addition, an adaptable ideal target representation, threshold computation, and a custom version of the EMA with a damping factor (DEMA) are introduced. This setup enables tracking people even when they are partially or fully occluded or when their appearance changes during tracking (for instance, by wearing a sweatshirt). These are challenging situations that are difficult for traditional MOT algorithms to handle, as our experiments have shown.

It is worth noting that, even if there are improvements compared to the previous Re-ID framework, this work has some limitations. One of them is the issue of catastrophic forgetting, which means that the algorithm tends to forget the appearance of the target as time passes because it continuously adapts to the new information it gathers. This can be faced with an online method that uses continuous learning techniques. The feature extraction network can be trained using people detections extracted during CARPE-ID execution. This would specialize the feature extractor with the tracked person appearances, forcing the outputted feature to be similar when belonging to the target and different if associated with a distractor. Proximity to the camera poses another limitation of our algorithm. However, this is not usually a problem in HRC scenarios because the human is generally near the robot and the camera. Lastly, the algorithm's time performance may suffer when more than 20 people are present in the image. Nevertheless, it is worth noting that such scenarios are rare, especially considering the robot camera's point of view.

To improve CARPE-ID, the suggested introduction of continual learning techniques is crucial. This technique involves using online self-supervised training to specialize the feature extraction network on the target appearances and to differentiate more effectively from the appearances of other people in the vicinity who may cause distractions.

This can be accomplished with a human Re-ID and tracking approach that leverages

continual learning and smart image pool generation to fine-tune the feature extractor network specifically for the tracked target's characteristics, thus extending the foundation laid by the previous CARPE-ID framework. To maintain both accuracy and efficiency (in terms of time), a twin network for feature extraction is adopted, which is trained in parallel with the original network responsible for tracking. Our results demonstrate that this framework outperforms its baseline, effectively tracking the target throughout all videos. To support our findings, saliency maps are provided for enhanced interpretability. These achievements offer valuable insights to the robotics community, promoting a more conscious HRI.

However, it is important to note a minor limitation in this work. Given that the Soft Triplet loss function requires both negative and positive samples, a constraint is that it requires at least one distractor in the image for effective training. Conversely, tracking could become trivial if the target is always the only person in the image. During our evaluation, difficulties arose due to the scarcity of datasets that met our evaluation criteria, particularly those involving changes in target appearance and total occlusions. Consequently, an intriguing avenue for future research could involve curating new datasets or integrating existing ones to address these scenarios better while establishing robust real-world benchmarks for Re-ID and tracking performance, especially in HRC.

6.2 Geometric environmental perception

The geometric perception of the surroundings is one of the first tasks required by a robot to move freely and safely. To further explore contextual awareness for the robotic platform, three key areas of geometric perception were addressed: SLAM, sensor filtering, and robust loop closure.

To cope with this task, a LiDAR SLAM approach, LEO-SLAM [107], has been presented. It employs a submap-based keyframe, a multi-level alignment strategy, and a submap-based SC++ within an adaptive search area. This system differs from the SoTA due to its alignment method, which prioritizes maintaining map consistency between consecutive scans rather than optimizing only the final robot pose, as often occurs in pure LiDAR odometry systems. Our experiments demonstrated that the proposed framework outperforms existing SoTA methods in terms of ATE and RPE.

As with many other LiDAR-based approaches, our SLAM system struggles in low-feature environments, such as long corridors. As shown in our experiments, errors introduced during scan matching can be reduced through the integration of external

odometry information or IMU data. To address this, a multi-sensory approach combined with intelligent multi-odometry source filtering can be used to provide robust final odometry, thereby reducing drift errors.

LEO-SLAM provides a stable foundation for future developments aimed at enabling reliable environmental understanding and advancing towards autonomous, conscious robotics.

During the experiments, some environments with reflections and refractions were encountered. In those environments, the maps obtained with the SLAM were not exactly like the original ones. To solve this problem, two different implementations of a ground-aware intensity filtering method have been proposed. They are designed to remove points caused by erroneous reflections from semi-transparent or translucent surfaces, commonly found in indoor environments (*e.g.*, windows). Each implementation offers distinct advantages and drawbacks depending on the environment. However, the simpler solution is preferred due to its faster performance and satisfactory results. Our evaluation of SoTA LiDAR odometry systems confirmed that LiDAR-based systems significantly benefit from our simple yet effective ground-aware intensity filtering in environments with frequent erroneous reflections, enhancing both precision and robustness.

To conclude the geometrical representation study, robust LCD has also been improved with the GSC algorithm. It is an enhanced variant of SC++ [54], which integrates statistical analysis of point cloud data to improve LCD performance. Unlike the original method, which relies solely on the maximal-height point within each context bin, GSC models the complete distribution of points using Gaussian statistics. This enables a more robust and descriptive representation, particularly in environments affected by noise and outliers.

Our experimental validation across multiple datasets, including KITTI and VBR, demonstrated that GSC consistently outperforms the baseline in terms of recall while maintaining perfect precision in most cases. The incorporation of Huber-weighted statistics further enhances robustness by reducing the impact of outlier points.

6.3 Semantics for environmental understanding and interaction

Once personalized Re-ID for HRC and geometrical mapping for environmental understanding have been established, the next step is to improve the robot's contextual

awareness by introducing other semantics in the environment.

To accomplish this task, the Artifacts Mapping framework [109] has been presented. It utilizes multi-modal sensor fusion to address the semantic mapping problem, a rare setup in robotics applications. The LiDAR and RGB-D camera sensor readings are fused to achieve better accuracy for both near and far objects, as opposed to camera-only systems, which lose accuracy for distant objects, or LiDAR-only systems, which lack high-level texture understanding of the environment.

A UI application is proposed to interact with the artifacts map obtained during the mapping application. This application is useful for performing autonomous high-level decision-making tasks because it exposes the object's class and location to the robot and the user.

The experiments demonstrated that our application can correctly detect, localize, and map 98% of the objects present in the scene at different distances, providing a small number of detection errors and good localization accuracy. The comparisons with the single-sensor scenario (either camera or LiDAR) proved that sensor fusion is essential for wide areas and high-accuracy applications.

In addition, a semantic loco-manipulation framework for object retrieval has been introduced to prove the usefulness of semantic maps. This application uses high-level semantic scene understanding to enable a robot assistant to search for and bring objects that are not nearby and whose locations may be unknown to the user. It is a foundational platform for enhancing robotic assistance within industrial environments, where robots work alongside human operators and actively participate in the workplace community.

Several future developments for this framework include the generalization of the pose estimation to handle more complex situations and objects, making experiments on a real robot, and improving the reactivity of the BT, *e.g.*, if the robot loses the grasp of the object, the BT should restart autonomously from the *pose_ estimation* behavior.

6.4 Final remarks on robotic contextual awareness and future directions

In conclusion, the works presented in this thesis all move in the direction of full robotic autonomy. Robots require complete contextual awareness to live and work in a human-populated world. They need to have capabilities similar to those of a person to interact with them, and at the same time be compliant with the environment. Contextual

awareness is strictly connected to how a person perceives and interacts with the environment, and this leads to the development of how a robot should do it. This should not be strictly followed because robots can also surpass human capabilities, mimicking other animals or living beings' capabilities, but we can always take into account that robots will interact with humans, and the world we built and are building is mainly shaped for our objectives and capabilities.

To accomplish these tasks, future work in the robotic contextual awareness direction is needed. Improvements in the surrounding understanding are crucial for the robot control. Robots should be able to deeply comprehend the semantics in the environment, their relations, and how they can employ that information to accomplish the tasks that are required. The more the robots can understand their surroundings, the more tasks they can perform. Researchers are quickly moving in this direction as robotic perception and understanding lay the foundations for robotic autonomy.

Bibliography

[1] Abdessalem Achour, Hiba Al-Assaad, Yohan Dupuis, and Madeleine El Zaher. Collaborative mobile robotics for semantic mapping: A survey. *Applied Sciences*, 2022.

[2] Salman Afghani and Muhammad Ishfaq Javed. Follow me robot using infrared beacons. *Academic Research International*, 4, 2013.

[3] Rahaf Aljundi, Daniel Olmeda Reino, Nikolay Chumerin, and Richard E Turner. Continual novelty detection. In *Conference on lifelong learning agents*, pages 1004–1025. PMLR, 2022.

[4] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016.

[5] Relja Arandjelovic and Andrew Zisserman. All about vlad. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1578–1585, 2013.

[6] José Arce, Niclas Vödisch, Daniele Cattaneo, Wolfram Burgard, and Abhinav Valada. Padloc: Lidar-based deep loop closure detection and registration using panoptic attention. *IEEE Robotics and Automation Letters*, 8(3):1319–1326, 2023.

[7] Saba Arshad and Gon-Woo Kim. Role of deep learning in loop closure detection for visual and lidar slam: A survey. *Sensors*, 21(4):1243, 2021.

[8] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE robotics & automation magazine*, 13(3):108–117, 2006.

[9] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 2008.

[10] Julie Stephany Berrio, Mao Shan, Stewart Worrall, and Eduardo Nebot. Camera-lidar integration: Probabilistic sensor fusion for semantic mapping. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[11] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.

[12] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact++: Better real-time instance segmentation. *IEEE trans on pattern analysis and machine intel*, 2020.

[13] Leonardo Brizi, Emanuele Giacomini, Luca Di Giammarino, Simone Ferrari, Omar Salem, Lorenzo De Rebotti, and Giorgio Grisetti. Vbr: A vision benchmark in rome. *arXiv preprint arXiv:2404.11322*, 2024.

[14] Simon Bultmann, Jan Quenzel, and Sven Behnke. Real-time multi-modal semantic fusion on unmanned aerial vehicles. In *2021 European Conference on Mobile Robots (ECMR)*, 2021.

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

[16] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[17] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proc of the IEEE conf on computer vision and pattern recognition*, 2017.

[18] Kenny Chen, Brett T Lopez, Ali-akbar Agha-mohammadi, and Ankur Mehta. Direct lidar odometry: Fast localization with dense point clouds. *IEEE Robotics and Automation Letters*, 7(2):2000–2007, 2022.

[19] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguere, Jens Behley, and Cyrill Stachniss. Suma++: Efficient lidar-based semantic slam.

In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537. IEEE, 2019.

[20] Qing Cheng, Niclas Zeller, and Daniel Cremers. Vision-based large-scale 3d semantic mapping for autonomous driving applications. In *2022 International Conference on Robotics and Automation (ICRA)*, 2022.

[21] Javier Civera, Dorian Gálvez-López, Luis Riazuelo, Juan D Tardós, and Jose Maria Martinez Montiel. Towards semantic slam using a monocular camera. In *2011 IEEE/RSJ international conference on intelligent robots and systems*, 2011.

[22] Serhan Coşar and Nicola Bellotto. Human re-identification with a robot thermal camera using entropy-based sampling. *Journal of Intelligent & Robotic Systems*, 2020.

[23] Yunge Cui, Xieyuanli Chen, Yinlong Zhang, Jiahua Dong, Qingxiao Wu, and Feng Zhu. Bow3d: Bag of words for real-time loop closing in 3d lidar slam. *IEEE Robotics and Automation Letters*, 8(5):2828–2835, 2022.

[24] Zhenyu Cui, Jiahuan Zhou, Xun Wang, Manyu Zhu, and Yuxin Peng. Learning continual compatible representation for re-indexing free lifelong person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16614–16623, 2024.

[25] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International journal of robotics research*, 27(6):647–665, 2008.

[26] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE computer society conf on computer vision and pattern recognition (CVPR'05)*, volume 1, 2005.

[27] Quoc Khanh Dang and Young Soo Suh. Human-following robot using infrared camera. In *Int Conf on Control, Automation and Sys*, 2011.

[28] Edoardo Del Bianco, Davide Torielli, Federico Rollo, Damiano Gasperini, Arturo Laurenzi, Lorenzo Baccelliere, Luca Muratore, Marco Roveri, and Nikos G Tsagarakis. A high-force gripper with embedded multimodal sensing for powerful and perception driven grasping. In *2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*, pages 149–156. IEEE, 2024.

[29] Frank Dellaert. Factor graphs: Exploiting structure in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):141–166, 2021.

[30] Yunhao Du, Zhicheng Zhao, Yang Song, Yanyun Zhao, Fei Su, Tao Gong, and Hongying Meng. Strongsort: Make deepsort great again. *IEEE Transactions on Multimedia*, 2023.

[31] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[32] Markus Eisenbach, Alexander Vorndran, Sven Sorge, and Horst-Michael Gross. User recognition for guiding and following people with a mobile robot in a clinical environment. In *IEEE/RSJ Int Conf on Intel Robots and Sys (IROS)*, 2015.

[33] Hao-Shu Fang, Jiefeng Li, Hongyang Tang, Chao Xu, Haoyi Zhu, Yuliang Xiu, Yong-Lu Li, and Cewu Lu. Alphapose: Whole-body regional multi-person pose estimation and tracking in real-time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[34] Matteo Frosi and Matteo Matteucci. Art-slam: Accurate real-time 6dof lidar slam. *IEEE Robotics and Automation Letters*, 7(2):2692–2699, 2022.

[35] Calvin Galagain, Martyna Poreba, and François Goulette. Is semantic slam ready for embedded systems? a comparative survey. *arXiv preprint arXiv:2505.12384*, 2025.

[36] Yixiao Ge, Dapeng Chen, and Hongsheng Li. Mutual mean-teaching: Pseudo label refinery for unsupervised domain adaptation on person re-identification. *arXiv preprint arXiv:2001.01526*, 2020.

[37] Vasantha Geetha, Sanket Salvi, Gurdeep Saini, Naveen Yadav, and Rudra Pratap Singh Tomar. Follow me: A human following robot using wi-fi received signal strength indicator. In *ICT Systems and Sustainability: Proceedings of ICT4SD 2020, Volume 1*, pages 585–593. Springer, 2020.

[38] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[39] Margarita Grinvald, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Cesar Cadena, Roland Siegwart, and Juan Nieto. Volumetric instance-aware semantic mapping and 3d object discovery. *IEEE Robotics and Automation Letters*, 2019.

[40] Michael Grupp. evo: Python package for the evaluation of odometry and slam, 2017.

[41] Tiziano Guadagnino, Xieyuanli Chen, Matteo Sodano, Jens Behley, Giorgio Grisetti, and Cyrill Stachniss. Fast sparse lidar odometry using self-supervised feature selection on intensity images. *IEEE Robotics and Automation Letters*, 7(3):7597–7604, 2022.

[42] Saurabh Gupta, Tiziano Guadagnino, Benedikt Mersch, Niklas Trekel, Meher VR Malladi, and Cyrill Stachniss. Efficiently closing loops in lidar-based slam using point cloud density maps. *arXiv preprint arXiv:2501.07399*, 2025.

[43] Julian Hau, Simon Bultmann, and Sven Behnke. Object-level 3d semantic mapping using a network of smart edge sensors. *arXiv preprint arXiv:2211.11354*, 2022.

[44] Dongjiao He, Wei Xu, Nan Chen, Fanze Kong, Chongjian Yuan, and Fu Zhang. Point-lio: Robust high-bandwidth light detection and ranging inertial odometry. *Advanced Intelligent Systems*, 5(7):2200459, 2023.

[45] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proc of the IEEE int conf on computer vision*, 2017.

[46] Li He, Wen Li, Yisheng Guan, and Hong Zhang. Igicp: Intensity and geometry enhanced lidar odometry. *IEEE Transactions on Intelligent Vehicles*, 2023.

[47] He Huang, Xinyuan Yan, Junxing Yang, Yuming Cao, and Xin Zhang. Lidsor: A filter for removing rain and snow noise points from lidar point clouds in rainy and snowy weather. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 48:733–740, 2023.

[48] Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: A real-time spatial perception system for 3d scene graph construction and optimization. *Robotics: Science and Systems XVIII*, 2022.

[49] Li Hui, Liping Di, Huang Xianfeng, and Li Deren. Laser intensity used in classification of lidar point cloud data. In *IGARSS 2008-2008 IEEE International Geoscience and Remote Sensing Symposium*, volume 2, pages II–1140. IEEE, 2008.

[50] Sajid Javed, Martin Danelljan, Fahad Shahbaz Khan, Muhammad Haris Khan, Michael Felsberg, and Jiri Matas. Visual object tracking with discriminative filters and siamese networks: a survey and outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[51] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[52] Alireza G Kashani, Michael J Olsen, Christopher E Parrish, and Nicholas Wilson. A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration. *Sensors*, 15(11):28099–28128, 2015.

[53] Harpreet Kaur and Jyoti Rani. A review: Study of various techniques of hand gesture recognition. In *IEEE Int Conf on Power Electronics, Intel Control and Energy Sys (ICPEICES)*, 2016.

[54] Giseop Kim, Sunwook Choi, and Ayoung Kim. Scan context++: Structural place recognition robust to rotation and lateral variations in urban environments. *IEEE Transactions on Robotics*, 38(3):1856–1874, 2021.

[55] Giseop Kim and Ayoung Kim. Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Oct. 2018.

[56] Jonathan JY Kim, Martin Urschler, Patricia J Riddle, and Jörg S Wicker. Symbiolcd: Ensemble-based loop closure detection using cnn-extracted objects and visual bag-of-words. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5425–5425. IEEE, 2021.

[57] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[58] Kenji Koide. small_gicp: Efficient and parallel algorithms for point cloud registration. *Journal of Open Source Software*, 9(100):6948, 2024.

[59] Kenji Koide, Jun Miura, and Emanuele Menegatti. Monocular person tracking and identification with on-line deep feature selection for person following robots. *Robotics and Autonomous Systems*, 2020.

[60] Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 2015.

[61] Edoardo Lamon, Fabio Fusaro, Pietro Balatti, Wansoo Kim, and Arash Ajoudani. A visuo-haptic guidance interface for mobile collaborative robotic assistant (moca). In *IEEE/RSJ Int Conf on Intel Robots and Sys (IROS)*, 2020.

[62] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information fusion*, 2020.

[63] Haoran Li, Jingfeng Wu, and Vladimir Braverman. Fixed design analysis of regularization-based continual learning. In *Conference on lifelong learning agents*, pages 513–533. PMLR, 2023.

[64] Jing Li, Xin Zhang, Jiehao Li, Yanyu Liu, and Junzheng Wang. Building and optimization of 3d semantic map based on lidar and camera fusion. *Neurocomputing*, 2020.

[65] Lin Li, Xin Kong, Xiangrui Zhao, Wanlong Li, Feng Wen, Hongbo Zhang, and Yong Liu. Sa-loam: Semantic-aided lidar slam with loop closure. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7627–7634. IEEE, 2021.

[66] Yong Li, Jiexin An, Na He, Yanbo Li, Zhenyu Han, Zishan Chen, and Yaping Qu. A review of simultaneous localization and mapping algorithms based on lidar. *World Electric Vehicle Journal*, 16(2):56, 2025.

[67] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[68] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conf on computer vision*. Springer, 2014.

[69] Haotian Liu, Rafael A Rivera Soto, Fanyi Xiao, and Yong Jae Lee. Yolactedge: Real-time instance segmentation on the edge. In *IEEE Int Conf on Robotics and Automation (ICRA)*. IEEE, 2021.

[70] Hong Liu, Liang Hu, and Liqian Ma. Online rgb-d person re-identification based on metric model update. *CAAI Transactions on Intelligence Technology*, 2017.

[71] Kangcheng Liu and Muqing Cao. Dlc-slam: A robust lidar-slam system with learning-based denoising and loop closure. *IEEE/ASME Transactions on Mechatronics*, 28(5):2876–2884, 2023.

[72] W. Liu, Dragomir Anguelov, D. Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, 2015.

[73] Carlone Luca, Kim Ayoung, Barfoot Timothy, Cremers Daniel, and Dellaert Frank. Slam handbook, 2025.

[74] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial intelligence*, 2021.

[75] Steve Macenski and Ivona Jambrecic. Slam toolbox: Slam for the dynamic world. *Journal of Open Source Software*, 6(61):2783, 2021.

[76] Tomas Berriel Martins, Martin R Oswald, and Javier Civera. Open-vocabulary online semantic mapping for slam. *arXiv preprint arXiv:2411.15043*, 2024.

[77] John McCormac, Ronald Clark, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. Fusion++: Volumetric object-level slam. In *2018 international conference on 3D vision (3DV)*, 2018.

[78] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and automation (ICRA)*, 2017.

[79] Le Miao, Wen Liu, and Zhongliang Deng. A frontier review of semantic slam technologies applied to the open world. *Sensors*, 25(16):4994, 2025.

[80] Michael J Milford and Gordon F Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *2012 IEEE international conference on robotics and automation*, pages 1643–1649. IEEE, 2012.

[81] Rana Mostafa, Hoda Baraka, and AbdElMoniem Bayoumi. Lmot: Efficient lightweight detection and tracking in crowds. *IEEE Access*, 10, 2022.

[82] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 2017.

[83] Tayyab Naseer, Jürgen Sturm, and Daniel Cremers. Followme: Person following and gesture recognition with a quadrocopter. In *IEEE/RSJ Int Conf on Intel Robots and sys*, 2013.

[84] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, 2011.

[85] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *Proc of the European Conf on Computer Vision (ECCV)*, 2018.

[86] Yue Pan, Xingguang Zhong, Louis Wiesmann, Thorbjörn Posewsky, Jens Behley, and Cyrill Stachniss. Pin-slam: Lidar slam using a point-based implicit neural representation for achieving global map consistency. *IEEE Transactions on Robotics*, 2024.

[87] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 2019.

[88] Ji-Il Park, Jihyuk Park, and Kyung-Soo Kim. Fast and accurate desnowing algorithm for lidar point clouds. *IEEE Access*, 8:160202–160212, 2020.

[89] Cosimo Patruno, Roberto Marani, Grazia Cicirelli, Ettore Stella, and Tiziana D'Orazio. People re-identification using skeleton standard posture and color descriptors from rgb-d data. *Pattern Recognition*, 2019.

[90] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 815–824, 2023.

[91] Wei Peng, Jingchuan Wang, and Weidong Chen. Tracking control of human-following robot with sonar sensors. In *Int Conf on Intel Autonomous Sys*, 2016.

[92] Valentina Pericu, Federico Rollo, and Navvab Kashiri. A scalable robot-agnostic voice control framework for multi-robot systems. In *2025 IEEE 22nd International Conference on Informatics in Control, Automation and Robotics (ICINCO)*. IEEE, 2025.

[93] Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Real-time progressive 3d semantic segmentation for indoor scenes. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[94] Sudeep Pillai and John Leonard. Monocular slam supported object recognition. *arXiv preprint arXiv:1506.01732*, 2015.

[95] Nattawat Pinrath and Nobuto Matsuhira. Simulation of a human following robot with object avoidance function. In *South East Asian Tech University Consortium (SEATUC)*, volume 1, 2018.

[96] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision (ECCV)*, 2020.

[97] Bonda Venna Pradeep, ES Rahul, and Rao R Bhavani. Follow me robot using bluetooth-based position estimation. In *Int Conf on Advances in Computing, Communications and Informatics (ICACCI)*, 2017.

[98] Nan Pu, Wei Chen, Yu Liu, Erwin M Bakker, and Michael S Lew. Lifelong person re-identification via adaptive knowledge accumulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7901–7910, 2021.

[99] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *International Conference on Computer Vision (ICCV)*, 2019.

[100] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating sys. In *ICRA workshop on open source software*, 2009.

[101] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[102] Gennaro Raiola, Michele Focchi, and Enrico Mingo Hoffman. Wolf: the whole-body locomotion framework for quadruped robots. *arXiv preprint arXiv:2205.06526*, 2022.

[103] Sonia Raychaudhuri and Angel X Chang. Semantic mapping in indoor embodied ai–a comprehensive survey and future directions. *arXiv preprint arXiv:2501.05750*, 2025.

[104] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[105] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc of the IEEE conf on computer vision and pattern recognition*, 2016.

[106] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 1995.

[107] Federico Rollo, Valentina Pericu, Marco Roveri, Arash Ajoudani, and Navvab Kashiri. Leo-slam: A multi-level scan matching approach with submap-based loop closure detection. In *2025 European Conference on Mobile Robots (ECMR)*, pages 1–7. IEEE, 2025.

[108] Federico Rollo, Gennaro Raiola, Nikolaos Tsagarakis, Marco Roveri, Enrico Mingo Hoffman, and Arash Ajoudani. Semantic-based loco-manipulation for human-robot collaboration in industrial environments. *arXiv preprint arXiv:2312.14487*, 2023.

[109] Federico Rollo, Gennaro Raiola, Andrea Zunino, Nikolaos Tsagarakis, and Arash Ajoudani. Artifacts mapping: Multi-modal semantic mapping for object detection and 3d localization. In *2023 European Conference on Mobile Robots (ECMR)*, pages 1–8. IEEE, 2023.

[110] Federico Rollo, Andrea Zunino, Arash Ajoudani, and Navvab Kashiri. Personalized re-identification through unsupervised continual learning and parallel training. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025.

[111] Federico Rollo, Andrea Zunino, Gennaro Raiola, Fabio Amadio, Arash Ajoudani, and Nikolaos Tsagarakis. Followme: a robust person following framework based on visual re-identification and gestures. In *2023 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*, pages 84–89. IEEE, 2023.

[112] Federico Rollo, Andrea Zunino, Nikolaos Tsagarakis, Enrico Mingo Hoffman, and Arash Ajoudani. Continuous adaptation in person re-identification for robotic assistance. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 425–431, 2024.

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

[114] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans. *arXiv preprint arXiv:2002.06289*, 2020.

[115] Martin Runz, Maud Buffier, and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2018.

[116] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[117] Radu Bogdan Rusu. Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24:345–348, 2010.

[118] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013.

[119] Christian Schlegel, Jörg Illmann, Heiko Jaberg, Matthias Schuster, and Robert Wörz. Vision based person tracking with a mobile robot. In *BMVC*, 1998.

[120] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *conf. on computer vision and pattern recognition (CVPR)*, 2015.

[121] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

[122] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *international conference on computer vision (ICCV)*, 2017.

[123] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[124] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 5135–5142. IEEE, 2020.

[125] Tixiao Shan, Brendan Englot, Carlo Ratti, and Daniela Rus. Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 5692–5698. IEEE, 2021.

[126] Mirai Shimoyama, Nobuto Matsuhira, and Kaoru Suzuki. Human characterization by a following robot using a depth sensor. In *IEEE/SICE Int Symp on Sys Integration (SII)*, 2017.

[127] Bing Shuai, Alessandro Bergamo, Uta Buechler, Andrew Berneshawi, Alyssa Boden, and Joe Tighe. Large scale real-world multi person tracking. In *European Conference on Computer Vision*. Springer, 2022.

[128] Zahra Soleimanitaleb and Mohammad Ali Keyvanrad. Single object tracking: A survey of methods, datasets, and evaluation metrics. *arXiv preprint arXiv:2201.13066*, 2022.

[129] Takafumi Sonoura, Hideichi Nakamoto, Manabu Nishiyama, Nobuto Matsuhira, Seiji Tokura, and Takashi Yoshimi. *Person following robot with vision-based and sensor fusion tracking algorithm*. INTECH Open Access Publisher, 2008.

[130] Daniel Stadler and Jurgen Beyerer. Improving multiple pedestrian tracking by track management and occlusion handling. In *Proc of the IEEE/CVF conf on computer vision and pattern recognition*, 2021.

[131] Niko Sünderhauf, Trung T Pham, Yasir Latif, Michael Milford, and Ian Reid. Meaningful maps with object-oriented semantic mapping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[132] Keisuke Tateno, Federico Tombari, and Nassir Navab. When 2.5 d is not enough: Simultaneous reconstruction, segmentation and recognition on dense slam. In *2016 IEEE international conference on robotics and automation (ICRA)*, 2016.

[133] Konstantinos A Tsintotas, Loukas Bampis, and Antonios Gasteratos. The revisiting problem in simultaneous localization and mapping: A survey on visual loop closure detection. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):19929–19953, 2022.

[134] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380, 1991.

[135] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. 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(2):1029–1036, 2023.

[136] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. Kiss-icp: In defense of point-to-point icp–simple,

accurate, and robust registration if done the right way. *IEEE Robotics and Automation Letters*, 8(2):1029–1036, 2023.

[137] H. Wang, C. Wang, and L. Xie. Intensity scan context: Coding intensity and geometry relations for loop closure detection. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2095–2101, 2020.

[138] Han Wang, Chen Wang, Chun-Lin Chen, and Lihua Xie. F-loam: Fast lidar odometry and mapping. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4390–4396. IEEE, 2021.

[139] Han Wang, Chen Wang, and Lihua Xie. Intensity-slam: Intensity assisted localization and mapping for large scale environment. *IEEE Robotics and Automation Letters*, 6(2):1715–1721, 2021.

[140] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024.

[141] Neng Wang, Xieyuanli Chen, Chenghao Shi, Zhiqiang Zheng, Hongshan Yu, and Huimin Lu. Sglc: Semantic graph-guided coarse-fine-refine full loop closing for lidar slam. *IEEE Robotics and Automation Letters*, 2024.

[142] Weiqi Wang, Xiong You, Lingyu Chen, Jiangpeng Tian, Fen Tang, and Lantian Zhang. A scalable and accurate de-snowing algorithm for lidar point clouds in winter. *Remote Sensing*, 14(6):1468, 2022.

[143] Yujiang Wang, Jie Shen, Stavros Petridis, and Maja Pantic. A real-time and unsupervised face re-identification system for human-robot interaction. *Pattern Recognition Letters*, 2019.

[144] Thomas Weber, Sergey Triputen, Michael Danner, Sascha Braun, Kristiaan Schreve, and Matthias Rätsch. Follow me: real-time in the wild person tracking application for autonomous robotics. In *Robot World Cup*, 2017.

[145] Longhui Wei, Shiliang Zhang, Wen Gao, and Qi Tian. Person transfer gan to bridge domain gap for person re-identification. In *Proc of the IEEE conf on computer vision and pattern recognition (CVPR)*, 2018.

[146] Thomas Whelan, Stefan Leutenegger, Renato Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: Science and Systems*, 2015.

[147] Linlin Xia, Jiashuo Cui, Ran Shen, Xun Xu, Yiping Gao, and Xinying Li. A survey of image semantics-based visual simultaneous localization and mapping: Application-oriented solutions to autonomous navigation of mobile robots. *International Journal of Advanced Robotic Systems*, 2020.

[148] Yu Xiang and Dieter Fox. Da-rnn: Semantic mapping with data associated recurrent neural networks. *arXiv preprint arXiv:1703.03098*, 2017.

[149] Wang Xiaoyu, LI Caihong, Song Li, Zhang Ning, and FU Hao. On adaptive monte carlo localization algorithm for the mobile robot based on ros. In *Chinese Control Conf (CCC)*. IEEE, 2018.

[150] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022.

[151] Hong Xuan, Abby Stylianou, and Robert Pless. Improved embeddings with easy positive triplet mining. In *Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[152] Dianyi Yang, Yu Gao, Xihan Wang, Yufeng Yue, Yi Yang, and Mengyin Fu. Opengs-slam: Open-set dense semantic slam with 3d gaussian splatting for object-level scene understanding. *arXiv preprint arXiv:2503.01646*, 2025.

[153] Liudi Yang, Ruben Mascaro, Ignacio Alzugaray, Sai Manoj Prakhya, Marco Karrer, Ziyuan Liu, and Margarita Chli. Lidar loop closure detection using semantic graphs with graph attention networks. *Journal of Intelligent & Robotic Systems*, 111(1):1–16, 2025.

[154] Hanjing Ye, Jieting Zhao, Yaling Pan, Weinan Cherr, Li He, and Hong Zhang. Robot person following under partial occlusion. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[155] Zhen Zeng, Yunwen Zhou, Odest Chadwicke Jenkins, and Karthik Desingh. Semantic mapping with simultaneous object detection and localization. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[156] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

[157] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.

[158] Ji Zhang and Sanjiv Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 2174–2181. IEEE, 2015.

[159] Ji Zhang, Sanjiv Singh, et al. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and systems*, volume 2, pages 1–9. Berkeley, CA, 2014.

[160] Yingying Zhang, Qiaoyong Zhong, Liang Ma, Di Xie, and Shiliang Pu. Learning incremental triplet margin for person re-identification. In *AAAI conf. on artificial intelligence*, 2019.

[161] Yucheng Zhang, Tian Wang, Kexin Liu, Baochang Zhang, and Lei Chen. Recent advances of single-object tracking methods: A brief survey. *Neurocomputing*, 455:1–11, 2021.

Appendix A

Pseudo Algorithms

Algorithm 1 CARPE-ID framework. Reprinted, with permission, from [112], © 2024 IEEE.

Input: $tracking\_id$
$\boldsymbol{\mu}, \boldsymbol{\sigma}, \mu_d, \sigma_d, \lambda_d \leftarrow initialize\_variables()$
while True **do**
  $\boldsymbol{I_{RGB}} \leftarrow cam.getRGB()$
  $dets \leftarrow mot.infer(\boldsymbol{I_{RGB}})$
  $feats \leftarrow reid.infer(dets)$
  $min\_dist \leftarrow MAX\_FLOAT\_NUM$
  $tracking\_feature \leftarrow Null$
  `// Re-identification`
  for $feature \in feats$ **do**
    if $tracking\_id = feat.id$ **then**
      $tracking\_feature \leftarrow feature$
      break
    else
      $d_{\boldsymbol{\mu},\boldsymbol{\sigma}} \leftarrow get\_distance(feature, \boldsymbol{\mu}, \boldsymbol{\sigma})$,
      if $d_{\boldsymbol{\mu},\boldsymbol{\sigma}} < min\_dist \ \ \& \ \ d_{\boldsymbol{\mu},\boldsymbol{\sigma}} < \lambda_d$ **then**
        $min\_dis \leftarrow d_{\boldsymbol{\mu},\boldsymbol{\sigma}}$
        $tracking\_feature \leftarrow feature$
        $tracking\_id \leftarrow feature.id$
      end if
    end if
  end for
  `// Target model update`
  if $tracking\_feature \neq Null$ **then**
    $var \leftarrow compute\_var(\boldsymbol{\mu}, tracking\_feature)$
    $\boldsymbol{\mu} \leftarrow DEMA(\boldsymbol{\mu}, tracking\_feature, \Delta_f)$
    $\boldsymbol{\sigma} \leftarrow DEMA(\boldsymbol{\sigma}, var, \Delta_f)$
    $var_d \leftarrow compute\_var(\mu_d, min\_dist)$
    $\mu_d \leftarrow DEMA(\mu_d, min\_dist, \Delta_{\lambda_d})$
    $\sigma_d \leftarrow DEMA(\sigma_d, var_d, \Delta_{\lambda_d})$
    $\lambda_d \leftarrow \mu_d + 2\sigma_d$
  end if
end while

Algorithm 2 LEO-SLAM algorithm. Reprinted, with permission, from [109], © 2023 IEEE.

Input: $\mathcal{P}$, ${}^{\mathrm{o}}\mathbf{H}_{\mathrm{b}} = \mathcal{I}_{4x4}$ *(optional)*
Output: ${}^{\mathrm{m}}\mathbf{H}_{\mathrm{o}}$

// Filtering
$\mathcal{P}_p \leftarrow preprocess(\mathcal{P})$
$\mathcal{P}_f \leftarrow filter(\mathcal{P}_p)$ // Used for map construction
${}^{\mathrm{m}}\mathbf{H}_{\mathrm{b}} \leftarrow {}^{\mathrm{m}}\mathbf{H}_{\mathrm{o}} * {}^{\mathrm{o}}\mathbf{H}_{\mathrm{b}}$
// Scan alignment
$\mathbf{H}^{ICP}_{\mathrm{s2s}} \leftarrow GICP(\mathcal{P}_p, \mathcal{P}_{prev}, {}^{\mathrm{m}}\mathbf{H}_{\mathrm{b}})$
$\mathbf{H}^{ICP}_{\mathrm{s2\mathbb{S}}} \leftarrow GICP(\mathcal{P}_p, \mathbb{S}, \mathbf{H}^{ICP}_{\mathrm{s2s}})$
${}^{\mathrm{m}}\mathbf{H}_{\mathrm{b}} \leftarrow \mathbf{H}^{ICP}_{\mathrm{s2\mathbb{S}}} * {}^{\mathrm{m}}\mathbf{H}_{\mathrm{b}}$
$d_m, d_\theta \leftarrow computeDistance({}^{\mathrm{m}}\mathbf{H}_{\mathrm{n_{c-1}}}, {}^{\mathrm{m}}\mathbf{H}_{\mathrm{b}})$
$\mathcal{S} \leftarrow addPcd(\mathcal{S}, \mathcal{P}_p)$
// Add new node
if $d_m \geq \Delta_m$ *or* $d_\theta \geq \Delta_\theta$ **then**
    // Submap alignment
    $\mathbf{H}^{ICP}_{\mathcal{S}2\mathbb{S}} \leftarrow GICP(\mathcal{S}, \mathbb{S}, {}^{\mathrm{m}}\mathbf{H}_{\mathrm{b}})$
    ${}^{\mathrm{m}}\mathbf{H}_{\mathrm{b}} \leftarrow \mathbf{H}^{ICP}_{\mathcal{S}2\mathbb{S}} * {}^{\mathrm{m}}\mathbf{H}_{\mathrm{b}}$
    $\mathrm{n_c} \leftarrow graph.addNode(\mathcal{S}, {}^{\mathrm{m}}\mathbf{H}_{\mathrm{b}})$
    // Loop Closure Detection
    $SC = computeScanContext(\mathcal{S})$
    $LC_{found}, LC_{ID} \leftarrow LCD(SC)$
    if LC_{found} **then**
        $\mathrm{n_l} \leftarrow graph.getNode(LC_{ID})$
        ${}^{\mathrm{n_c}}\mathbf{t}_{\mathrm{n_l}} \leftarrow {}^{\mathrm{m}}\mathbf{t}_{\mathrm{n_c}} - {}^{\mathrm{m}}\mathbf{t}_{\mathrm{n_l}}$
        $\mathbb{S}_c, \mathbb{S}_l \leftarrow computeSubmapsNeighbours()$
        $\mathbf{H}^{ICP}_{lc} \leftarrow GICP(\mathbb{S}_c, \mathbb{S}_l, {}^{\mathrm{n_c}}\mathbf{t}_{\mathrm{n_l}})$
        // Pose graph optimization
        if $GICP.hasConverged()$ **then**
            ${}^{\mathrm{n_c}}\mathbf{H}_{n_l} \leftarrow (\mathbf{H}^{ICP}_{lc} * {}^{\mathrm{m}}\mathbf{H}_{\mathrm{n_c}})^{-1} * {}^{\mathrm{m}}\mathbf{H}_{\mathrm{n_l}}$
            $graph.addLoop(\mathrm{n_c}, \mathrm{n_l}, {}^{\mathrm{n_c}}\mathbf{H}_{n_l})$
            $graph.isam2optimization()$
            $graph.update()$
            ${}^{\mathrm{m}}\mathbf{H}_{\mathrm{b}} \leftarrow graph.getLastNodePose()$
        end if
    end if
    $\mathbb{S} \leftarrow getPrevSubmaps()$
end if
$\mathcal{P}_{prev} \leftarrow transformPcd(\mathcal{P}_p, {}^{\mathrm{m}}\mathbf{H}_{\mathrm{b}})$
${}^{\mathrm{m}}\mathbf{H}_{\mathrm{o}} = {}^{\mathrm{m}}\mathbf{H}_{\mathrm{b}} * {}^{\mathrm{o}}\mathbf{H}_{\mathrm{b}}^{-1}$
return ${}^{\mathrm{m}}\mathbf{H}_{\mathrm{o}}$