DISS. ETH NO. 29148

MODULAR MAPPING AND LEARNING FOR ROBOTICS

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES OF ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

ANDREI CRAMARIUC

MSc. ETH Electrical Engineering and Information Technology

born on August 11, 1993
citizen of Romania and Finland

accepted on the recommendation of

Prof. Dr. Roland Siegwart, examiner
Prof. Dr. Juan Domingo Tardós, co-examiner
Prof. Dr. Igor Gilitschenski, co-examiner

2023

Autonomous Systems Lab
Department of Mechanical and Process Engineering
ETH Zurich
Switzerland

© 2023 Andrei Cramariuc. All rights reserved.

ABSTRACT

The ability to precisely localize a robot within the environment is a core capability for mobile robotics. Knowing a robot's precise pose permits various tasks, such as navigation in an environment, interaction between multiple agents, or mobile manipulation of an object. Various applications such as, for example, autonomous driving, delivery robots, augmented reality (AR), mobile manipulation, and robot inspection greatly benefit from an accurate and robust underlying Simultaneous Localization And Mapping (SLAM) system. Motivated by these widespread applications, the goal of this thesis is to improve the existing limits of SLAM. We focus on integrating deep learning into various components of SLAM, as we aim to integrate the advances in perceptual understanding deep learning has provided. We wish to draw inspiration from how humans perceive the environment to include more semantically meaningful landmarks in the mapping process. We believe deep learning will not only improve existing solutions but provide novel solutions to yet unsolved problems.

Different robots and environments have different optimal sensor configurations and perception requirements regarding SLAM. For example, micro aerial vehicles have limited payload and a different perspective than indoor wheeled robotic platforms. Accommodating these differences in software requires a flexible framework that can support multiple different modalities and types of sensors. However, many existing frameworks are very rigid and require significant changes when prototyping new products or testing novel research ideas. Therefore, in the first part of the thesis, we will focus on a new multi-modal, multi-robot, and modular SLAM framework. We show through experiments that state-of-the-art accuracy is possible, and we showcase the integration of various experimental modules.

In the second part of the thesis, we will focus on using deep learning to improve SLAM, first, by working on a new type of landmark and descriptor. We further explore the inclusion of semantics into the localization process of our proposed framework. Finally, we use learning to address maintaining accurate long-term calibration in SLAM, which is essential to the operation of all downstream tasks. We focus on the modular and meaningful integration of learning, as we believe this leads to robust robotic systems that remain understandable and easier to design.

ZUSAMMENFASSUNG

Die Fähigkeit zur präzisen Lokalisierung eines Roboters in seiner Umgebung spielt eine zentrale Rolle in der mobilen Robotik. Die Kenntnis der genauen Position und Orientierung eines Roboters ermöglicht verschiedene Aufgaben, wie die Navigation innerhalb seiner Umgebung, die Wechselwirkung zwischen mehreren Agenten oder den Umgang mit Objekten. Verschiedene Anwendungen wie z. B. autonomes Fahren, Lieferroboter, Augmented Reality (AR), das Manipulieren von Objekten mit mobilen Agenten und Inspektion mithilfe von Robotern profitieren in hohem Maße von einem genauen und robusten SLAM-System (Simultaneous Localization And Mapping). Motiviert durch diese weit verbreiteten Anwendungen ist das Ziel dieser Arbeit, bekannte Limitierungen von SLAM zu überwinden. Dazu konzentrieren wir uns auf die Integration von Deep Learning in verschiedene Komponenten von SLAM, da wir die Fortschritte im Bereich des Wahrnehmungsverständnisses, die durch Deep Learning erreicht wurden, nutzen wollen. Wir möchten uns davon inspirieren lassen, wie Menschen ihre Umgebung wahrnehmen, um mehr semantisch bedeutsame Orientierungspunkte in die Kartierung einzubeziehen. Wir glauben, dass Deep Learning nicht nur bestehende Lösungen verbessern, sondern auch neue Lösungen für bisher ungelöste Probleme bieten wird.

Verschiedene Roboter und Umgebungen haben unterschiedliche optimale Sensorkonfigurationen und Wahrnehmungsanforderungen in Zusammenhang mit SLAM. Mikro-Luftfahrzeuge haben beispielsweise eine begrenzte Nutzlast und eine andere Perspektive als Roboterplattformen mit Rädern in Innenräumen. Um diese Unterschiede in der Software zu berücksichtigen, ist eine flexible Plattform erforderlich, die unterschiedliche Modalitäten der Dateninterpretation und Sensortypen unterstützen kann. Viele bestehende Software-Plattformen sind jedoch sehr starr und erfordern erhebliche Änderungen, wenn es um die Entwicklung neuer Produkte oder die Erprobung neuer Forschungsideen geht. Daher konzentrieren wir uns im ersten Teil der Arbeit auf ein neues modulares SLAM-Framework, das für unterschiedliche Roboter und Zusammenstellungen geeignet ist. Wir zeigen anhand von Experimenten, dass eine aktuell bestmögliche Genauigkeit erreichbar ist, und wir präsentieren die Integration verschiedener experimenteller Module.

Im zweiten Teil der Arbeit konzentrieren wir uns auf die Verwendung von Deep Learning zur Verbesserung von SLAM, indem wir zunächst an einem neuen Typ von Orientierungspunkten und Deskriptoren arbeiten. Des Weiteren untersuchen wir die Einbeziehung von Semantik in den Lokalisierungsprozess des von uns vorgeschlagenen Frameworks. Schließlich nutzen wir maschinelles Lernen, um eine genaue Langzeitkalibrierung bei SLAM zu gewährleisten, die für alle davon abhängigen Aufgaben unerlässlich ist. Wir konzentrieren uns auf die modulare und sinnvolle Integration des maschinellen Lernens, da wir der Meinung sind, dass dies zu robusten Robotersystemen führt, die verständlich bleiben und einfacher zu entwerfen sind.

ACKNOWLEDGEMENTS

Many people have contributed directly and indirectly to the research presented in this thesis and I would like to use this opportunity to thank some of them, knowing full well that this list cannot be comprehensive. First and foremost, I would like to express my sincere gratitude to Prof. Dr. Roland Siegwart for offering me the opportunity to join his lab and work on exciting new research topics. He created a unique atmosphere of exploration, freedom, and collaboration among a group of bright and friendly people that form the Autonomous Systems Lab (ASL). Through his leadership and insights he created many opportunities for scientific research, and provided me with the possibility to make many new life-long connections and friendships.

A big thank you to Prof. Dr. Juan Domingo Tardós and Prof. Dr. Igor Gilitschenski, for agreeing to act as co-examiners on the committee of the doctoral defense. I would especially like to thank them for their critical comments and insightful questions that have helped improve this thesis.

I would furthermore like to thank Dr. Cesar Cadena, Prof. Dr. Igor Gilitschenski, and Juan Nieto for their insightful supervision during my PhD. Many explanations, rewriting, lengthy discussions, and ideas happened throughout the four years of my studies. They were there for both successes and rejections, for which I will always be grateful. I have learned much from them and their support and hope to carry these experiences forward in my career.

When thinking of the members of ASL, Renaud Dubé is one of the first to come to mind, as he introduced me to the lab, supervising my semester project. Later as my colleague I am sad we did not have more time to work on further successful projects. From the mapping team I would like to thank the old guard whose experience and discussions about maplab guided me very much at the beginning, namely Marius Fehr, Thomas Schneider, Marcin Dymczyk, and Hannes Sommer. I can not go by without thanking Florian Tschopp, who was my neighbor and close collaborator for more than three years. He especially taught me the value of student supervision, and I can not count the hours we spent in meetings. Similarly thanks to Lukas Bernreiter, with whom I enjoyed working on projects together, and always appreciated his knowledge of programming, statistics, and sci-fi. Thanks to Cornelius von Einem, who originally a student, and now a colleague I've had great times and worked on interesting projects. I learned much about robotic

manipulation when research interests aligned with Michel Breyer, and it was a pleasure always when he popped in for common coffee breaks and chit chat. Thanks also to Lukas Schmid, for the shared work on volumetric mapping, and his willingness to organize events and participate in board game nights. I would also like to mention the rest of the mapping team, with whom I'm happy to have cooperated on various projects and shared inspiring discussions Victor Reijgwart, Patrick Pfreundschuh, Hermann Blum, Lionel Ott, and Olov Andersson. From the rest of ASL I can't name everyone that in one way or another has influenced this work, but I want to mention by name Jen Jen Chung, Nicholas Lawrance, Daniel Dugas, Fadri Furrer, Tonci Novkovic, Julian Förster, Kenneth Blomqvist, Giuseppe Rizzi, Paula Wulkop, and Francesco Milano. Finally nothing at ASL would be possible without the technical expertise of Michael Riner and the administrative support provided by Luciana Borsatti and Cornelia Della Casa. It was a pleasure to be part of such a great lab as ASL, I will always remember fondly the times spent together at work and outside, and I hope it has lead to many friendships and collaborations that will last a lifetime.

In addition to ASL, many external institutions and people played a role in this thesis. I would like to thank SBB for their funding and support for data collection, provided during the difficult times of the pandemic. Particular thanks also to Bosch for their support, hosting and with all the great work done together with Stefan Benz and Dr. Tillmann Falck. Finally, a thanks to the lab of Prof. Dr. Margarita Chli and especially David Hug for their cooperation on event based projects.

During my PhD I had the pleasure of supervising many talented and hard-working students in their own studies. It was a pleasure to be involved in so many people's lives and careers. For their direct contributions to publications and this thesis, I would like to thank Aleksandar Petrov, Rohit Suri, Mayank Mittal, Andreas Bühler, Le Chen, Yunke Ao, Cornelius von Einem, Patrick Pfreundschuh, Jiapeng Zhong, Zheyu Ye, Benjamin Hahn, Stefan Lionar, Samuel Gull, Chunwei Xing, and Xinyu Sun.

Finally, I am deeply grateful to my family and friends without whom all of this would not have been possible. I would like to thank my mother for her continuous and unwavering love and support throughout everything. I would like to thank my friends, for all the adventures and good times we have shared, for all the support that I have received during hardships and for always being there for me. Last but not least, I am grateful for the love and support of Erika, my partner in life, and maybe some day in science. I look forward to the next step in life together·

Financial Support

The research leading to the results presented in this thesis has received funding from the ETH Mobility Initiative under the project PROMPT, and from Bosch Corporate Research.

CONTENTS

ABSTRACT i

ZUSAMMENFASSUNG iii

ACKNOWLEDGEMENTS v

PREFACE 1

1 INTRODUCTION 3
  1.1 Challenges 5
  1.2 Motivation and Objectives 6
  1.3 Approach 8
    1.3.1 Part A: Modular Mapping 9
    1.3.2 Part B: Modular Learning 9

2 CONTRIBUTION 13
  2.1 Part A: Modular Mapping 13
  2.2 Part B: Modular Learning 15
  2.3 List of Publications 21
    2.3.1 Publications Included in this Thesis 21
    2.3.2 Other Publications 21
  2.4 Conference and Workshop Dissemination 23
  2.5 Teaching and Student Supervision 24
  2.6 List of Open-source Software 28

3 CONCLUSION AND OUTLOOK 29
  3.1 Part A: Modular Mapping 29
    3.1.1 Discussion 30
    3.1.2 Future Outlook 31
  3.2 Part B: Modular Learning 31
    3.2.1 Discussion 32
    3.2.2 Future Outlook 33

A MODULAR MAPPING 35

PAPER I: MAPLAB 2.0 – A MODULAR AND MULTI-MODAL MAPPING FRAMEWORK 37
1 Introduction 38
2 Related Work 40
3 The `maplab 2.0` Framework 42
4 Use-Cases 47
5 Conclusion 54

B MODULAR LEARNING 55

PAPER II: SEGMAP: SEGMENT BASED MAPPING AND LOCALIZATION USING DATA-DRIVEN DESCRIPTORS 57
1 Introduction 58
2 Related Work 61
3 The SegMap Approach 64
4 The SegMap Descriptor 67
5 Experiments 70
6 Discussion and Future Work 84
7 Conclusion 86

PAPER III: SEMSEGMAP – 3D SEGMENT-BASED SEMANTIC LOCALIZATION 89
1 Introduction 91
2 Related Work 93
3 SemSegMap 94
4 Experiments 98
5 Conclusions 106

PAPER IV: LEARNING CAMERA MISCALIBRATION DETECTION 109
1 Introduction 110
2 Related Work 111
3 Methodology 114
4 Experiments and Discussion 117
5 Conclusion 123

BIBLIOGRAPHY 124

CURRICULUM VITAE 147

PREFACE

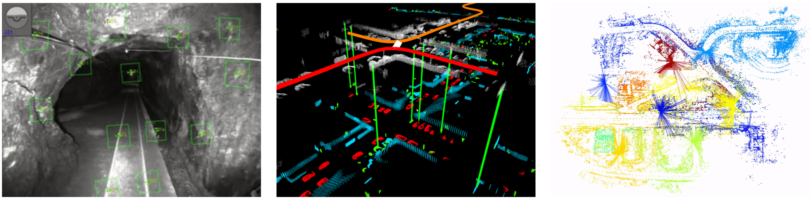
This is a cumulative doctoral thesis, which combines four first-author publications resulting from research carried out during the author's doctoral studies. The work was carried out under the supervision of Prof. Roland Siegwart at the Autonomous Systems Lab (ASL) at ETH Zürich. The thesis is divided into three prefacing chapters that put the publications into context, followed by the full publications in Parts A and B. Chapter 1 introduces the main motivations for the research topics and highlights the challenges this thesis aims to tackle. Furthermore, the desired goals and approaches taken to address the introduced challenges are discussed. Chapter 2 introduces the specific publications included in the thesis, puts them in context, and explains interrelations and their impact. Additionally, the chapter introduces additional contributions to the field of robotics, such as involvement in other publications not included as part of this thesis, teaching, student supervision, dissemination efforts, and contributions to open-source software. Finally, in Chapter 3, the findings of the thesis are summarized, and an outlook into potential future research directions is provided.

1

INTRODUCTION

An essential component of any robotic system is Simultaneous Localization And Mapping (SLAM). It allows robots to function independently in new environments and to localize and perform tasks in previously visited places. For example, SLAM is used in autonomous driving [1], mobile manipulation [2], household robotics [3], and augmented reality (AR) [4]. In these examples, SLAM can be used to navigate through the indoor or outdoor environment. However, navigation is only one of many use cases. Localization in a common map between multiple robotic platforms enables more complex interactions: autonomous cars can synchronize to reduce traffic jams, cleaning robots can divide work, or AR users can interact with each other in an application. With such varied applications, environments, and sensor combinations that a robotic platform can have, much research goes into perfecting SLAM solutions.

Over recent years, many tailored SLAM solutions have been successfully developed for specific environments or sensor configurations [7–15]. However, many challenges remain until SLAM is fully solved or generically deployed in any conditions. One issue with current solutions is their design rigidity and how tightly integrated various components are. SLAM can be



(a) Visual-inertial odometry in difficult conditions, in an underground mine, using ROVIO [5].

(b) LiDAR localization using *SegMap* [6], in an urban driving scenario.

(c) Globally consistent and optimized multi-session map in an disaster rescue training area.

Figure 1.1: Showcasing some of the components of SLAM in different environments.

canonically divided into multiple parts: odometry, localization, and global or local optimization. While variations exist, most SLAM systems contain the aforementioned components in one way or another. We showcase the operation of some of these components on different systems and in different environments in Figure 1.1. Many times in research, these parts are regarded as separate, and solutions are developed and proposed individually for each subproblem. For example, odometry methods [5, 16–18], and benchmarks exist for multiple environments, sensors, and scenarios [19–21]. New global localization methods, with a high focus on deep learning in recent years, are constantly being developed. In computer vision, visual feature extraction is a popular topic [22–24] and recently also deep learning-based feature trackers [25, 26]. Finally, multiple optimization backends exist [27–30], but their use is difficult as they require significant integration work for more complex problems and much fine-tuning.

While the SLAM problem can be regarded as highly modular, developing a fully-fledged framework requires extensive work. Integration requires not only communication between multiple components but also many technical considerations that need to be addressed, such as bookkeeping, data synchronization, map storage, visualizations, *etc*. This leads to many works where the SLAM framework is tightly integrated, and not much work has been put into making generic interfaces and components. As a result, while functional and sometimes highly accurate, these systems are hard to extend in future research and integrate into new platforms or products. The other phenomenon is that many modules are only tested individually in a bubble. While this specific testing provides a good way of directly comparing components, it does not fully tell how useful new features are in a real-world setting. This lack of a complete testing platform impacts the efficiency of testing new research or prototype components.

In particular, in robotics over the past decade, deep learning has changed how we approach perception problems. Convolutional Neural Networks (CNNs) have pushed the boundaries of state-of-the-art tasks such as object detection, place recognition, and robotic grasping. In addition, with the development of more efficient hardware acceleration and network architectures, deep learning is becoming capable of solving more complex problems. However, these intricate, deep learning-based solutions are sometimes hard to interpret. The question is how to easiest incorporate learning into robotic systems that need to function in unforeseen environments. For example, in SLAM, many deep learning solutions tackle localization, 3D reconstruction, semantic awareness, feature detection and tracking. However, many of these solutions often go untested in a full SLAM pipeline since their integration would take too much time and be very subjective based on the implementation. Therefore, a modular framework with easily replaceable or modifiable components is needed.

A modular approach to incorporating learning into robotics is also beneficial for interpretability and robust operation. For example, deep-learned solutions exist that, from a single image, can predict a full 6 degrees of freedom (DoF) pose. However, these solutions skip many safeguards present in hand-crafted, more human-engineered approaches. Moreover, the lack of interpretability makes it challenging to incorporate human intuition into learning-based components. The smaller the learning problem, the easier it is to control and include checks that prevent catastrophic failures. Robustness is essential everywhere, as components are highly linked in robotic systems. For example, navigation depends on state estimation and SLAM. Therefore small modular learning solutions are desirable, as well as a modular SLAM framework that can easily incorporate a variety of such modules.

1.1 CHALLENGES

The task of SLAM and integration of learning into mapping is challenging for several reasons. Some of the key challenges in these problems are listed below, especially focusing on the challenges we aim to address.

Different environments There is much variation in the environments we expect to use robots in, and they each present their own unique difficulties. From indoor environments (*e.g.* office buildings, factories, or hospitals), to outdoor environments (*e.g.* cities or forests), to special cases such as mines, construction sites, or disaster sites. As different robots and sensors are necessary or ideal for each case, we also need perception tools that can work flexibly and robustly in various configurations.

Multi-session mapping refers to the ability to manage maps taken in the same environments at different time instances. This involves not only the ability to merge data across temporal gaps but also to gain a benefit in doing so. Merging multiple mapping sessions together involves challenges and innovations not only in localization but also in data management. Irrelevant data needs to be thrown out, and much bookkeeping needs to be done to keep maps consistent.

Multi-robot mapping refers to multiple robots simultaneously exploring the same environment and communicating to create one global map. While it has some similar aspects to multi-session mapping, the challenge of multi-robot mapping involves more real-time challenges. Multiple robots need to be synchronized together, and online operation becomes a critical aspect of the SLAM framework. Additional challenges include when the robots have different sensor configurations.

Long-term mapping is the ability to continuously create consistent and accurate maps over long periods of time. Changes in the environment happen continuously: dynamic objects move and disappear, furniture gets moved around, or new buildings appear in the city. While tackling multi-session mapping allows for updated maps, continuous data collection is a time-consuming process. Maps with features that remain usable longer increase robustness and decrease operating costs.

Long-term sensor deployment One aspect of long-term robot deployment is ensuring that sensors remain operational. A critical aspect of any sensor setup is the quality of its calibration. Degraded calibrations quickly start affecting not only mapping performance but, for example, navigation and interactive tasks with the environment. Separating sensor failures from failures in the perception task can be difficult, and redundant sensors are not always available or easy to exploit.

Online operation is essential on robotic platforms with limited resources. As many systems, such as navigation, rely on mapping, online operation is needed to ensure they remain operational.

Compact maps go towards enabling large-scale mapping and also efficient multi-robot communication. Reducing the map size must be done to ensure long-term viability and robustness.

The aforementioned challenges are recurring themes in the works constituting this thesis. While some of the challenges have been addressed in other works, they are most often in limited environments or on specific robotic platforms. No general solutions exist yet. While some of the works in this thesis will focus on one challenge more than the other, the issues remain intertwined, and a perfect SLAM system would need to address them all.

1.2 MOTIVATION AND OBJECTIVES

The ability to autonomously map and simultaneously localize in an environment enables numerous robotic tasks and applications. These include autonomous driving, mobile manipulation, service robotics, AR applications, and many more. SLAM therefore receives much attention in the robotics community, and numerous proposed methods exist. Similarly, the integration of deep learning into robotics has achieved many successful results in the last decade, achieving leaps in perception that were not possible before. For example, object detection, feature extraction, segmentation, 3D reconstruction, and path planning have all greatly benefited from deep learning-based solutions. The next steps in merging learning and mapping in robotics are robustness and flexibility.

In SLAM factor graph-based approaches have emerged as one go-to solution due to their flexibility in easily including different types of constraints and their efficiency and sparsity. Factor graphs model states as nodes in a graph, which are constrained by observations that induce edges in the graph. This simple representation, combined with non-linear solvers, provides a very powerful representation of maps. One major challenge is the amount of work necessary to set up the framework that goes from raw data to an optimization problem formulation. Many small details, such as time synchronization, sensor calibration, data buffering, and storage, need to be addressed. These initial steps set up a large overhead to any new development in SLAM for robotics, as every new module requires an entire framework built around it. One main objective at the beginning of this thesis was to provide a framework that can support the easy integration, and testing, of new mapping modules into a flexible framework. Quick testing not only allows ideas and prototypes to advance quicker into products in the robotics industry but also benefits the research community. By allowing the testing of modules in a larger framework, comparability of results becomes easier across sub-topics and methods. Our goal is not only to develop a framework with generic interfaces and flexible usage but also one that can be extended. Easy extendibility is crucial as we can not foresee all future use cases. Additionally, our goal is not to focus on providing a new framework but on showing and developing features towards tackling the challenges mentioned in the previous chapter. We aim to show development towards multi-robot and multi-session SLAM and flexibility with sensors and robot configurations.

Furthermore, our goal is to include learning-based components into the previously mentioned SLAM framework. At the beginning of this thesis, there was a large trend towards learning-based descriptors and visual keypoints. In recent years deep learning has also enabled more semantically meaningful analysis of the environment. In this thesis, we aim to explore using deep learning to address some of the challenges mentioned in the previous section. More specifically, we aim to find and explore areas of robotics where we think deep learning-based solutions will have the largest impact. We wish to find components in robotic mapping systems where human intuition can be exploited only so far, and the next step towards progress is exploiting data-driven solutions. The expected outcome is to see performance improvements similar to the ones in object detection and semantic segmentation, where data-driven solutions have become the standard. Additionally, we wish to explore the possibility of finding new solutions to problems that were previously considered too hard or too expensive to solve.

The motivation behind this thesis is to address challenges in SLAM from two sides. First, through better engineering solutions, we wish to provide more versatile and adaptable SLAM frameworks. The goal is to have a platform with easily replaceable modules and support for many different types of robots and sensors. Second, through the use of novel data-driven deep learning frameworks, we wish to improve the performance of existing SLAM components. This is a more research-focused direction, where much effort is made toward developing completely new approaches to previous problems.

1.3 APPROACH

When examining how humans orient themselves in the environment, we notice the use of many high-level semantic queues, such as buildings, signs, unique objects, *etc*. Scene understanding is a critical part of our ability to efficiently map and navigate through an environment. Thus to enable robots to overcome some of the listed challenges in this thesis, we will work towards facilitating and integrating deep learning into various stages of SLAM. To accomplish our objectives, we group this thesis into two parts.

In Part A, we focus on creating a modular SLAM framework to facilitate integrating and testing new ideas. A major driving force behind this is the lack of good testing platforms that existed at the beginning of this thesis. The first part of the thesis addresses many technical challenges and provides solutions that address many of the limitations in SLAM mentioned in Section 1.1.

Part B explores the integration of deep learning into various components of SLAM. The idea is to replace small components that would benefit the most from data-driven approaches. By keeping the components limited, we maintain modules whose performance and interactions with other components can more easily be controlled. We address important challenges in descriptors, incorporating semantics into SLAM, and detecting miscalibration in long-term camera deployment.

1.3.1 *Part A: Modular Mapping*

We first address the problem of a unifying SLAM framework that allows the integration of different modules and has flexible sensor support. We focus on factor graph-based solutions as they are more generic than, for example, volumetric approaches [31, 32] where modifications to the underlying map structure are more complex. The basis of any factor graph-based SLAM approach is the ability to optimize and modify the graph-like map representation. While numerous non-linear optimization frameworks exist (*e.g.*, GTSAM [27], g2o [29], and Ceres [30]), much work is needed to implement the surrounding tools for SLAM. This involves, most notably, processing the sensor data into constraints, calculating initialization values for the different state variables, storing the map data, merging multiple maps, and handling inputs from multiple robots simultaneously. Other publicly available SLAM frameworks [12–15] simplify the required engineering work by having very tightly integrated and non-flexible components.

To overcome these difficulties, we propose a new SLAM framework that is flexible and multi-modal. We base our design on maplab [15], which is a strictly visual-inertial mapping framework. We fundamentally modify the original maplab to create a new framework, `maplab 2.0` [33], that better supports our goals. First, we remove the constraint of visual-inertial sensor data and enable the use of any type of odometry. Second, we enable the addition of numerous different modalities and constraints that can be provided externally. All of these can be added to the same factor graph problem and optimized jointly. Among the newly added constraints is also the use of multiple different types of visual feature points, in particular deep-learned ones that have shown to be more effective at localization in difficult conditions and across larger changes in time and viewpoint. Finally, we also enable online and multi-robot operation. All these changes together address many of the challenges listed in Section 1.1, and bring many of the existing solutions in SLAM together in one unified framework.

1.3.2 *Part B: Modular Learning*

At the start of this thesis, the integration of deep learning into mapping and SLAM was at its beginnings. One of the first areas of SLAM that saw very practical use of deep learning was keypoint detection and descriptor generation. These are both topics that had previously seen many handcrafted solutions proposed based on human intuition. The process of descriptor generation is well suited for data-driven learning approaches due to the availability of data, real or simulated, and the use of similarity losses where little human intervention was needed. In these topics, deep learning has proven far superior to previous solutions. Keypoint detection and description are normally followed by humanly meaningful fail-safes to ensure robustness (*e.g.*, random sample consensus (RANSAC), outlier rejection during triangulation, and robust cost functions in optimization). These safety checks enable the easy use of deep-learned methods, as mistakes due to unpredictable behavior or new environments can be filtered out. We show easy integration of deep-learned keypoints into a SLAM framework.

While visual keypoint descriptors have been largely studied, we use deep learning to find solutions to descriptor problems where applying human intuition was difficult. In particular, for more modern sensors, such as LiDAR, where processing 3D data requires a new paradigm in thinking when considering previous works on 2D image analysis. In *SegMap*, we propose a novel way of generating meaningful landmarks in LiDAR-based maps. Instead of focusing on local points or on entire scans, we take a middle ground where we model the environment using segments. While in our initial approach, these segments have no semantically clear meaning, they represent distinguishable parts of the environment, *e.g.* parts of cars, buildings, or vegetation. We use deep learning to find a much more powerful descriptor for these segments than handcrafted variants. We show that the descriptors alone can be used to recreate a 3D representation of the environment. We again incorporate this learning module into a larger SLAM framework to show its robustness and usability in real-world conditions and experiments.

Deep learning has also shown great potential for semantic understanding. We wish to extend and improve our previously described *SegMap* approach to leverage semantic awareness. This is inspired by how we as humans use semantic queues from the environment and object awareness to navigate around an environment. We integrate semantic awareness into the *SegMap* descriptor generation process in two steps. First, we improve the segmentation process so that segments are now more semantically meaningful but not entirely limited to segment categories either. Second, we include semantic information in the network's input during training to achieve higher recall. We named the resulting SLAM framework *SemSegMap*. We also took the semantic localization process one step further and showcased a fully semantic loop-closure engine in `maplab 2.0`. There, inspired by the *SegMap* and *SemSegMap* localization methods, we showcase the easy integration of a fully object-based loop-closure method.

Finally, we explore the use of learning in a completely different facet of robotics, namely calibration. While obtaining very precise calibrations is a tedious and complicated process, it is also essential when good performance is required. Long-term deployment of robotic platforms requires not only a good starting calibration but also the ability to maintain it. Accidents or shocks can cause sensors to, for example, move, deform, or get scratched. All of the above might cause a sensor to become miscalibrated, *i.e.* the existing calibration would no longer model reality sufficiently well. While methods exist that can perform online calibration, these can be very expensive or rely on other secondary sensors. We develop a novel method that uses deep learning to detect when a sensor is miscalibrated. This allows us to know when to use an in-the-wild calibration method or to ask the user to take the sensor to be re-calibrated. In our approach, we focus on monocular cameras, as they are a very commonly used sensor, and 2D images are ideal for deep CNNs. Detecting if the intrinsic parameters of a monocular camera are sufficiently well calibrated also presents an interesting challenge, as such a task is difficult to parameterize using hand-crafted methods. Finally, our miscalibration module also fits well into our idiom of modular learning. We focus on one specific subtask that is part of a bigger pipeline and does not rely fully on learning. Even if our method were to erroneously predict a miscalibration, this would only consume resources to re-calibrate and not cause a full system failure.

2

CONTRIBUTION

In this chapter, we detail the novel contributions of each paper presented in this cumulative thesis. Additionally, we describe the context of the work at the time of publication and the motivations behind it. Finally, we discuss the interrelations between each paper and the other contributions in this thesis and our work's impact in the field of SLAM and robotics. We not only discuss interrelations between the publications that form this cumulative thesis but also to other publications to which the author contributed, and that are detailed in Section 2.3.2. We organized the publications into the two parts introduced in Chapter 1. First, we present our new modular SLAM framework, and then we move to the learning-based contributions to various SLAM components.

2.1 PART A: MODULAR MAPPING

PAPER I

Andrei Cramariuc*, Lukas Bernreiter*, Florian Tschopp*, Marius Fehr, Victor Reijgwart, Juan Nieto, Roland Siegwart, Cesar Cadena, "maplab 2.0 – A Modular and Multi-Modal Mapping Framework". In *IEEE Robotics and Automation Letters*, 2023.

Context

During and before this thesis, various SLAM frameworks have been published and open-sourced. However, most of them have been dedicated to a specific sensor configuration [7–15] or to a very specific purpose, such as a multi-robot SLAM or semantic mapping. Similarly, multiple works exist, many of the most recent ones employing deep learning, that explore improvements to various components of SLAM, such as keypoint detection and description, localization, outlier detection, *etc*. However, many of these works are tested in restricted experiments and not as part of a full pipeline. Similarly, adapting and re-purposing existing frameworks typically involves considerable engineering effort that does not directly contribute to research. Having a flexible pipeline that supports multiple different sensor configurations and has easily adaptable and replaceable modules would be of great use in many projects. It would not only benefit the unified comparison of novel research ideas in SLAM, but it would also allow fast prototyping for industrial purposes. As the goal of this thesis was to research ways in which learning could be leveraged to replace existing components in SLAM in a modular and controlled way, the need for a modular SLAM framework became apparent.

Contribution

To achieve this ambitious goal, we base ourselves on maplab [15], which was a mapping framework for purely visual-inertial mapping. We transformed the framework into a new multi-modal and multi-robot SLAM framework, named `maplab 2.0`, that allows the integration of an unparalleled amount of different data. The extension to multi-robot mapping was done through submapping and a centralized server that all robots communicate with. This not only allows for new developments in cooperative robotics but also enables online SLAM for the single robot scenario. The framework integrates all the multi-session and map management capabilities of its predecessor, thus enabling a highly customizable multi-session mapping experience. To showcase how `maplab 2.0` can be used, we perform experiments in which we integrate experimental 3D LiDAR keypoints and a semantic object-based loop closure engine. We also show that we can reach great accuracy by using state-of-the-art odometry and any number of visual features, including deep-learned keypoints. Our new proposed framework incrementally deals with many of the challenges we list in Section 1.1. Not only do we deal with many issues in SLAM, but we do so in one unified framework.

Interrelations and Dissemination

This work started with this thesis and was a motivation behind many of the other works done during the author's studies. The need for accurate calibrations, especially in the visual-inertial case, was apparent when working with `maplab 2.0`. It motivated the work into miscalibration detection to ensure accurate long-term operation (Paper IV and [34]), and the struggles in learning to accurately calibrate visual-inertial sensors served as a motivation for the works on automatic and efficient visual-inertial calibration using robotic arms [35, 36]. The success of segment-based mapping from Papers II and III was the inspiration behind a semantically even more meaningful loop closure method using objects. This work is continued in the constellation-based loop closure method proposed in [37].

The practical uses of the `maplab 2.0` framework were first showcased in our go and fetch demo [2] for indoor mobile manipulation. Later, most notably, the `maplab 2.0` framework was used by the winning team CERBERUS in the DARPA SubT challenge [38]. The software is made open-source[1] and has a significant userbase and usage throughout the community.

Personal Involvement

My personal contribution to the project involved implementing the multi-sensor and multi-feature integration. I also implemented the 3D features and performed the experiment shown in the paper. Additionally I did most of the work for the HITLI and EuRoC evaluations. I also contributed to the mapping node and multi-robot code. Additionally, I improved, fixed, and rewrote miscellaneous parts of the original maplab code, *e.g.* the optimization module.

2.2 PART B: MODULAR LEARNING

PAPER II

Renaud Dubé*, Andrei Cramariuc*, Daniel Dugas, Hannes Sommer, Marcin Dymczyk, Juan Nieto, Roland Siegwart, and Cesar Cadena, "SegMap: Segment based mapping and localization using data-driven descriptors". In *The International Journal of Robotics Research (IJRR)*, 2020.

Context

Much of the research into localization originally focused on local keypoint-based methods or on global descriptor methods. While highly precise, local methods suffer from a lack of descriptive power across large changes in time or viewpoint. Similarly, global methods have more context that they can use, but they lack precision as they normally only provide place recognition and not full 6 DoF localization. Our idea was to explore something in between, namely segments in 3D LiDAR maps. We define segments as larger chunks

---

[1] `https://github.com/ethz-asl/maplab/`

of the environment that are not necessarily semantically meaningful (*i.e.*, entire cars or buildings) but geometrically unique and easy to re-identify. These segments then represented landmarks, which could be stored and localized against. Originally finding handcrafted descriptors was difficult [39], as describing these chunks of the environment had little human intuition involved. However, this was an ideally suited task for deep learning-based data-driven methods.

Contribution

Our contribution was a segment-based SLAM pipeline for LiDAR that relies on deep learning for localization and visualization. The most significant part of our work was developing and training the novel 3D segment description network. We used GPS data and a convex hull matching technique to obtain ground truth information and showed on real-world datasets that our method outperformed baseline methods. A significant advantage of our segment-based method was the compactness of the resulting map, which also enabled extremely efficient inter-robot communication in our multi-robot experiment. Additionally, our CNN provided multiple other useful secondary functionalities. The first such functionality was the ability to 3D reconstruct the map from the compact segment descriptor representations. Second, we trained an additional network to semantically classify the descriptor into meaningful classes, showing that the network inherently learned semantically meaningful representations. Our *SegMap* framework also addressed numerous of the challenges from Section 1.1, among which are long-term mapping, compact map representations, and multi-robot mapping.

Interrelations and Dissemination

The work here strongly correlates with the topic of integrating learning into SLAM in a modular fashion. It inspired the need for the work in Paper I, which would have removed the need for much of the required groundwork. Additionally, this set up the work in Paper III, where we take semantic integration into the segmentation and description process one step further. We also take a similar approach in [37] but instead of segments use full objects. Both the need to automate the labeling of data and the presence of multiple dynamic objects in the dataset motivated the work in [40], where we automatically learn to segment and remove dynamic objects from LiDAR scans. Our work is open-sourced[2] and has received wide interest and use as a baseline in other publications [41, 42].

Personal Involvement

I solely developed and tested the learning part of the project, which included the descriptor generation, 3D reconstruction, and semantic classification modules. Additionally, I integrated the LiDAR odometry and contributed to the KITTI experiments.

PAPER III

Andrei Cramariuc*, Florian Tschopp*, Nikhilesh Alatur, Stefan Benz, Tillmann Falck, Marius Brühlmeier, Benjamin Hahn, Juan Nieto, and Roland Siegwart, "SemSegMap – 3D Segment-based Semantic Localization". In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

Context

Semantic awareness is a critical component of how humans perceive the environment, and we tend to use semantically meaningful landmarks to orient ourselves in the environment. This intuition on human behavior motivates the research into including semantically meaningful landmarks also into the standard SLAM pipeline [8, 43, 44]. Deep learning and data availability have provided very successful semantic segmentation networks on which the works in this publication rely. One weakness in *SegMap* was the unreliability of the segmentation process, which needed to be tuned depending on the type of environment. Additionally, the uniqueness of the resulting segments was hard to control, *e.g.* most walls looked the same, leading to poor quality descriptors.

Contribution

In this publication, we extend the previously mentioned *SegMap* framework into *SemSegMap*, by incorporating semantics awareness and color information into the framework. We include these new attributes not only in the description generation network but also in the segmentation process. We, therefore, extend both of the key components in the localization module. The proposed

---

[2] `https://github.com/ethz-asl/segmap/`

improvements increase the repeatability of the segmentation process and generate descriptors with better retrieval. As opposed to *SegMap*, we test the new framework both in simulation and on a different real-world dataset, showing in both cases the resulting improvement in localization performance. Most importantly, we show that *SemSegMap* can localize better across large temporal gaps, thus addressing the challenge of long-term mapping.

Interrelations and Dissemination

SemSegMap is a direct extension of our work in Paper II, improving the network by adding a new learning module. Even with the inclusion of semantics and color, we notice many segments that are difficult to describe due to lack of distinguishing features, *e.g.* many parts of walls, vegetation, or cars, can appear very similar. We believe that we must include even more environmental awareness to generate more uniqueness. In another subsequent work [37], we propose encoding spatial information on the configuration of multiple neighboring landmarks into the descriptor. *SemSegMap* is also open-sourced together with *SegMap*.

Personal Involvement

I implemented the new segmentation and learning pipeline in *SemSegMap*, which included a new loss, new inputs and a new network backbone. Additionally, contributed to the experiments and evaluations on the simulated data and NCLT.

PAPER IV

Andrei Cramariuc*, Aleksandar Petrov*, Rohit Suri, Mayank Mittal, Roland Siegwart, and Cesar Cadena, "Learning Camera Miscalibration Detection". In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

Context

The ability to maintain a stable calibration over long periods of time can be challenging, also depending on the environment and robotic platform. Shocks, scratches, mechanical stress, or user errors can easily change the calibration parameters of a sensor setup. The accuracy of the downstream tasks, *e.g.*, SLAM, is affected as they are directly tied together. Detecting a miscalibrated setup from other conditions, such as environmental degeneracy (*e.g.*, bad weather or lack of lighting), can be difficult. Particularly difficult are monocular camera setups where reliance on auxiliary sensors is impossible. Additionally, linking changes in camera intrinsic parameters (focal length, optical center, and distortion coefficients) to humanly intuitive effects can be challenging. We, therefore, see a good niche for a deep learning module that can detect a miscalibrated camera. This module can then, for example, be used to trigger a much more expensive online calibration or ask the user to take the robot in for maintenance.

Contribution

We propose a completely novel way of detecting when a monocular camera has been miscalibrated, *i.e.*, the existing calibration no longer matches the sensor's current physical state. We show on a real-world dataset how such a system can predict from a single image when a downstream task, in this case, monocular odometry, will perform poorly due to a miscalibrated sensor. To simplify the problem and increase robustness, instead of developing a CNN that can generalize to any camera and calibration, our solution is to individually specialize the CNN to each camera and calibration pairing. Therefore, the first challenge was to create an easy and efficient data generation pipeline. As it would be very time-consuming to generate multiple training sets with different calibrations for one camera, we proposed a strategy that uses real images but only needs one recording. The second challenge was developing a metric on which to train that would quantify miscalibration – for this, we proposed the *average pixel position difference* (APPD).

Interrelations and Dissemination

Detecting miscalibrations is an essential component in any SLAM system with long-term deployment goals. Especially since it is often an external module, integration can be relatively easy in many cases, making it a potentially beneficial addition to the work from Paper I. This publication was followed by a subsequent publication of the author that deals with stereo camera miscalibration detection [34]. Additionally, as mentioned above, the module can be used to request a re-calibration of the sensors, which is not a trivial process for an inexperienced user. This ties into our other work on learning optimal motions for calibration [35, 36]. Finally, the code for this work too was made publicly available for the benefit of the community[3].

---

[3] `https://github.com/ethz-asl/camera_miscalib_detection`

Personal Involvement

I am responsible for the original idea of training a per-camera specialized network to detect when a camera is no longer calibrated. Additionally, I also came up with the dataset generation strategy. I implemented the network training code and architecture.

2.3 LIST OF PUBLICATIONS

In the context of the author's doctoral studies the following publications were achieved.

2.3.1 *Publications Included in this Thesis*

1. A. Cramariuc*, A. Petrov*, R. Suri, M. Mittal, R. Siegwart, and C. Cadena, "Learning Camera Miscalibration Detection," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4997–5003.
2. R. Dubé*, A. Cramariuc*, D. Dugas, H. Sommer, M. Dymczyk, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: Segment-based mapping and localization using data-driven descriptors," *The International Journal of Robotics Research (IJRR)*, vol. 39, no. 2-3, pp. 339–355, 2020.
3. A. Cramariuc*, F. Tschopp*, N. Alatur, S. Benz, T. Falck, M. Brühlmeier, B. Hahn, J. Nieto, and R. Siegwart, "SemSegMap – 3D Segment-based Semantic Localization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1183–1190.
4. A. Cramariuc*, L. Bernreiter*, F. Tschopp*, M. Fehr, V. Reijgwart, J. Nieto, R. Siegwart, and C. Cadena, "maplab 2.0 – A Modular and Multi-Modal Mapping Framework," *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 2, pp. 520–527, 2023.

2.3.2 *Other Publications*

5. K. Blomqvist*, M. Breyer*, A. Cramariuc*, J. Förster*, M. Grinvald*, F. Tschopp*, J. J. Chung, L. Ott, J. Nieto, and R. Siegwart, "Go Fetch: Mobile Manipulation in Unstructured Environments," in *IEEE International Conference on Robotics and Automation (ICRA) Workshop on Perception, Action, Learning*, 2020.
6. A. Bühler, A. Gaidon, A. Cramariuc, R. Ambrus, G. Rosman, and W. Burgard, "Driving Through Ghosts: Behavioral Cloning with False Positives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5431–5437.
7. L. Chen*, Y. Ao*, F. Tschopp, A. Cramariuc, M. Breyer, J. J. Chung, R. Siegwart, and C. Cadena, "Learning Trajectories for Visual-Inertial System

Calibration via Model-based Heuristic Deep Reinforcement Learning," in *Conference on Robot Learning (CoRL)*, 2020.

8. S. Lionar*, L. Schmid*, C. Cadena, R. Siegwart, and A. Cramariuc, "NeuralBlox: Real-Time Neural Representation Fusion for Robust Volumetric Mapping," in *International Conference on 3D Vision (3DV)*, 2021, pp. 1279–1289.
9. P. Pfreundschuh, H. F. Hendrikx, V. Reijgwart, R. Dubé, R. Siegwart, and A. Cramariuc, "Dynamic Object Aware LiDAR SLAM based on Automatic Generation of Training Data," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 641–11 647.
10. F. Tschopp*, C. von Einem*, A. Cramariuc*, D. Hug, A. W. Palmer, R. Siegwart, M. Chli, and J. Nieto, "Hough^{2}Map – Iterative Event-Based Hough Transform for High-Speed Railway Mapping," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 2745–2752, 2021.
11. J. Zhong*, Z. Ye*, A. Cramariuc, F. Tschopp, J. J. Chung, R. Siegwart, and C. Cadena, "CalQNet-Detection of Calibration Quality for Life-Long Stereo Camera Setups," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 1312–1318.
12. Y. Ao*, L. Chen*, F. Tschopp, M. Breyer, R. Siegwart, and A. Cramariuc, "Unified Data Collection for Visual-Inertial Calibration via Deep Reinforcement Learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1646–1652.
13. C. Xing*, X. Sun*, A. Cramariuc, S. Gull, J. J. Chung, C. Cadena, R. Siegwart, and F. Tschopp, "Descriptellation: Deep Learned Constellation Descriptors for SLAM," *arXiv preprint arXiv:2203.00567*, 2022.

2.4 CONFERENCE AND WORKSHOP DISSEMINATION

The research conducted over the course of the doctoral studies was presented at multiple national and international conferences, workshops, and seminars, listed here.

1. Go Fetch: Mobile Manipulation in Unstructured Environments, presented at the *Workshop on Perception, Action, and Learning*, held at the International Conference on Robotics and Automation (ICRA), June 2020, Virtual.

2. Learning Camera Miscalibration Detection, presented at the *International Conference on Robotics and Automation (ICRA)*, June 2020, Virtual.

3. *SemSegMap* – 3D Segment-based Semantic Localization, presented at the *International Conference on Robotics and Automation (ICRA)*, June 2021, Virtual.

4. `maplab 2.0` – A Modular and Multi-Modal Mapping Framework, to be presented at the *International Conference on Robotics and Automation (ICRA)*, June 2023, London.

2.5 TEACHING AND STUDENT SUPERVISION

During the doctoral studies, a large emphasis was placed on teaching and supervising students. In particular, I was a teaching assistant for both the lab's courses "Autonomous Mobile Robots" in 2019 and "Perception and Learning for Robotics" in 2019, 2020 and 2021. I contributed to the organization of the Robotics Summer School in 2019, 2020, and 2022. In addition, over 45 students have been mentored in almost as many projects during the course of the doctoral studies. For projects that contributed to a publication, a citation is provided.

Master Thesis

Master student for 6 months full time (30 ECTS).

1. Jonas Aeschbacher (Spring 2019):
   "Semantic Indoor Localization using Deep Learned Object Descriptors"

2. Benjamin Hahn (Spring 2019):
   "3D Segment-based SLAM With Stereo Vision" [46]

3. Roman Ehrler (Autumn 2019):
   "Development of a Localization and Control Algorithm for an Agricultural Robot"

4. Juichung Kuo (Autumn 2019):
   "Enhance Visual-Inertial Mapping with Semantic Loop closure"

5. Andreas Buhler (Spring 2020):
   "Uncertainty-aware Monocular Depth Estimation for Robust Planning" [47]

6. Patrick Pfreundschuh (Spring 2020):
   "Dynamic Object Detection for Robust and Accurate LiDAR SLAM" [40]

7. Cornelius von Einem (Spring 2020):
   "Path-Constrained Localization for Rail vehicles"

8. Andrej Adzic (Autumn 2020):
   "Revisiting Deep Learned Visual Features for Robotics" (ETH Medal 2022 for outstanding Master Thesis)

9. Stefan Lionar (Spring 2021):
   "Neural Representations for Robotic Mapping" [48]

10. Philipp Egg (Spring 2022):
    "CAD modeling and global registration from indoor LiDAR scans"
11. Oliver Widler (Spring 2022):
    "PCES: Path-Constrained Event-based SLAM"
12. Matthias Brucker (Autumn 2022):
    "Using Patches for Self-Supervised Obstacle Detection on Railways"

Semester Thesis

Master student for 3-4 months part time (8 ECTS).

1. Juichung Kuo (Autumn 2018):
   "Semantic Landmarks for Maplab"
2. Rohit Suri (Autumn 2019):
   "L-Infinity SLAM: An Alternative to Bundle Adjustment"
3. Stefano d'Apolito (Autumn 2019):
   "Robust odometry for ground robots"
4. Petar Subotic (Autumn 2019):
   "LIKey: Lens-Invariant Keypoints"
5. Yassin Sdiri (Autumn 2019):
   "High-Speed SLAM for Rail Vehicles using a 1D Laser"
6. Cornelius von Einem (Autumn 2019):
   "High-Speed SLAM for Rail vehicles using Dynamic Vision Systems" [49]
7. Julius Erbach (Spring 2020):
   "Multi-View Object Detection"
8. Tiancheng Hu (Autumn 2020):
   "Object Instance Re-localization from Partial Observations"
9. Samuel Gull (Autumn 2020):
   "Descriptellation: Constellation Based Description for Object SLAM"
10. Tuna Turcan (Autumn 2020):
    "Deep-Learned Gravity Estimation for Robust Visual-Inertial SLAM"

11. Asil Örgen (Autumn 2020):
    "Estimating Uncertainty for Deep-Learned Visual Features"
12. Le Chen, and Yunke Ao (Spring 2021):
    "Learn to Calibrate Visual-Inertial Systems" [36]
13. Riccardo Rancan (Autumn 2021):
    "Map-based Anomaly Detection with Region of Interest & Laser"
14. Sourya Kovvali (Autumn 2021):
    "High-speed Mapping with DVS"
15. Zador Pataki (Autumn 2021):
    "How to Fuse?"
16. Oguzhan Ilter (Autumn 2021):
    "Multimodal Object Detection & Mapping"

Bachelor Thesis

Bachelor student for 3-4 months part time (15 ECTS).

1. Dominik Bornand (Spring 2019):
   "Non Linear Optimization Based Approach to Superquadrics for Object-Based SLAM"
2. Shuaixin Qi (Autumn 2019):
   "Artifact Detection in Challenging Conditions"
3. Marc Odermatt (Spring 2020):
   "Simultaneous Localization and Mapping on the TurtleBot3"
4. Andrea Balestra (Autumn 2020):
   "Autonomous Charging for Life-Long Autonomous Mapping"

"Perception and Learning for Robotics" course project

Master students for 3-4 months part time (4 ECTS).

1. Fadhil Ginting, and Hamza Javed (Spring 2019):
   "TextVLAD: Robust Visual Scene Representation for Place Recognition Using Text Landmarks"

2. Mayank Mittal, Aleksandar Petrov, and Rohit Suri (Spring 2019):
   "Using Semantics to Detect Miscalibration" [45]

3. Tuna Turcan, and Asil Örgen (Spring 2020):
   "DEEL-VIO: Deep End-to-End Learning Framework for Visual Inertial Odometry"

4. Le Chen, and Yunke Ao (Spring 2020):
   "Learning Optimal Trajectories for Visual-inertial System Calibration via Deep Reinforcement Learning" [35]

5. Soomin Lee, and Xiaoao Song (Spring 2020):
   "Intelligent Overfitting to Environments"

6. Jiapeng Zhong, and Zheyu Ye (Spring 2020):
   "Learn to Detect Sensor miscalibration" [34]

7. Chunwei Xing, and Xinyu Sun (Spring 2021):
   "Deep-Learned Constellation-based descriptors for Semantic SLAM"

"Studies on Mechatronics"

Bachelor student for 3-4 months part time (5 ECTS).

1. Pascal Gross (Autumn 2019):
   "Advances in SLAM"

2. Ajaykumar Unagar (Spring 2020):
   "Comparative study of learned Detectors and Descriptors"

Visiting Students and Summer Projects

1. Hyunjin Jung (Spring 2022):
   "Motion Estimation using Event-based Camera"

2.6 LIST OF OPEN-SOURCE SOFTWARE

Throughout the doctoral studies, I have actively developed and contributed to software packages used for various robotics applications. At the time of writing, the contributed software has 2500+ stars and 1000+ forks on github. The most notable open-source contributions include:

1. `Maplab 2.0`: A multi-modal and modular mapping framework that can work with multiple robots and in challenging environments [33].
   `https://github.com/ethz-asl/maplab`
2. `Segmap`: A segment-based LiDAR SLAM framework that uses deep learned descriptors for loop closure [6, 46].
   `https://github.com/ethz-asl/segmap`
3. `Camera Miscalibration Detection`: Deep learning-based method for detecting when the intrinsic calibration of a monocular camera has degraded [45].
   `https://github.com/ethz-asl/camera_miscalib_detection`
4. `Hough2Map`: Iterative Hough transform for event-based cameras and line-based mapping on railway vehicles using a double Hough transform [49].
   `https://github.com/ethz-asl/hough2map`
5. `NeuralBlox`: Incremental volumetric mapping for indoor scenes based on neural representations [48].
   `https://github.com/ethz-asl/neuralblox`
6. `Learn to Calibrate`: Reinforcement learning framework to teach a robot arm how to most efficiently calibrate a visual-inertial sensor [35, 36].
   `https://github.com/ethz-asl/learn-to-calibrate`
7. `Descriptellation`: Learning constellation based descriptors for long-term place recognition [37].
   `https://github.com/ethz-asl/descriptellation`

3

CONCLUSION AND OUTLOOK

In this chapter, we aim to summarize the overall findings following the topics of modular mapping and learning introduced in Chapter 1. While the presented contributions are promising, we also wish to add some critical discussion regarding potential improvements and limitations. Furthermore, based on our findings and experience gathered throughout this thesis, we wish to provide an outlook and discuss emerging opportunities for research from the contributions listed here.

3.1 PART A: MODULAR MAPPING

In the first part of the thesis, we focused on developing a large mapping framework that supports multiple sensors and robot types. We focused on the modular aspect of the framework, endeavoring to enable the use of as many different components that are also easily interchangeable. Additionally, using submapping we enabled online and multi-robot SLAM using a centralized server design. While all this implied significant amounts of engineering work, we simultaneously showcased novel applications in research that can be done using our framework.

From the challenges listed in Section 1.1 we incrementally addressed most of them directly or indirectly. Most notably, we showed that it is possible in one framework to handle very different environments (office buildings, parking lots, basements, construction sites, search and rescue areas, and other outdoor areas) and different types of robotic platforms (hand-held sensors, wheeled sensors, and drones) with many different sensor configurations. This allowed us to combine various external capabilities (*e.g.*, state-of-the-art LiDAR odometry, deep-learned visual features, and map sparsification) to reach better accuracy while maintaining compact maps.

3.1.1 *Discussion*

We believe our framework is a good tool for the robotics community to test out new research and quickly prototype new products or ideas. However, flexibility and multipurposeness also come at the cost of increased complexity. The first source of complexity comes from settings that can be modified and relate to each individual component (odometry, keypoint detection, loop closure, *etc.*). While adding a new module might be easy, it will cause changes downstream and possibly require tuning parameters throughout the framework. This, in turn, requires experience and familiarity with much of the framework. A second source of complexity that requires expert knowledge is the multi-session mapping workflow, which allows for very easy tuning and gives the user much control and responsibility over the process. While in simple or standard conditions, this is robust and will work out of the box, new environments or new robotic platforms might require tuning and understanding many of the underlying components. This is in contrast to many more tightly integrated solutions which are also, therefore, more robust but also more limited in scope and in the number of new conditions they can handle.

While we rely on a factor graph as an underlying representation, we recognize some implementation limitations. The first relates to the scalability of the optimization process. Solutions such as iSAM2 [28] exist that can only partially update the factor graph and thus can deal with larger problems. In contrast, our use of Ceres [30] as a non-linear optimization framework forces full batch updates but allows for more flexibility in writing new constraints with the auto-differentiation function. Keyframing and map sparsification [50, 51] can alleviate the problem of scalability but also fail after a certain point as they will gradually drop more information until the map becomes too sparse to be useful.

Another issue with factor graphs is the need to provide confidence estimates for all measurements. Most often, these are arbitrarily chosen values since it is impossible to ascertain how reliable a measurement truly is. For example, a wheel encoder can not ascertain how much slippage occurs, or a keypoint detector can not estimate the re-projection error in pixels. While some relative confidence values between measurements from the same modality might be possible, comparability across sensors can become difficult. Incorporating multiple sensors can still require hand-tuning of measurement confidences, also depending on the type of environment.

3.1.2 *Future Outlook*

Hierarchical Approaches While our approach combines multiple modalities, not all modalities need to be used similarly. Drawing inspiration from how humans think in a hierarchical fashion, using different levels of semantic understanding, algorithms could learn to mimic this behavior. In particular, semantically more meaningful and oftentimes less accurate methods can be used as a filtering step for slower, more accurate methods [52]. This can be extended to multiple levels and to more complex decision processes (*e.g.*, exclude the GPS sensors when indoors). Such a hierarchical implementation would also reduce some of the difficulties in tuning when adding multiple types of measurements with varying degrees of precision into the same factor graph. Testing methods in such a hierarchical system would also require a framework that implements not only a modular design but also an easily configurable high-level control system with modifiable stages.

Uncertainty Estimation Another major issue, as we have highlighted in the discussion section, is the tuning of measurement uncertainties. The topic of uncertainty estimation is not unknown in the field of deep learning [53]. However, more work is needed to integrate and test these uncertainties in SLAM systems. Especially many deep learning-based modules and frameworks do not provide accurate or meaningful confidence values for their predictions. This complicates integration into factor graphs, as fixed hand-tuned values can fail in new conditions.

3.2 PART B: MODULAR LEARNING

In the second part of the thesis, we focused on developing and incorporating new learning-based additions to SLAM. Here our focus was on multiple different components that go into different parts of SLAM. We succeeded in showing state-of-the-art results in all our works, as well as novel approaches to problems. First, we presented a new type of deep-learned landmark for loop closure. Also, we showed the advantages of the CNN we developed in various secondary tasks, such as classification and reconstruction. Afterward, we dwelled deeper into semantics and incorporated more object awareness. Finally, we also showed a prototype for a fully object-based loop closure engine. These works primarily target the challenge of long-term mapping, seeking to increase robustness to viewpoint or temporal changes. Being high-level, the resulting maps are compact and sparse, enabling more efficient optimization and data transfer.

The second goal we achieved was related to maintaining long-term accurate calibrations in SLAM. By developing a new system for detecting when a system is miscalibrated, we enabled efficient use of resources where sensors are re-calibrated only when necessary. This work proposes some novel approaches to the problem, first of all regarding data acquisition and the target metric for learning. Secondly, we diverged from the general trend of CNNs that can generalize to all cases and proposed specializing a CNN to a specific camera calibration pair. This avoids the seemingly impossible task of generic miscalibration detection and simplifies it to a more intuitive task of detecting when perceptual changes occur. This work ties in also to other contributions in this thesis regarding miscalibration detection [34] and learning motions for visual-inertial calibration [35, 36].

3.2.1 *Discussion*

We believe learning is the future of perception in robotics, as more deep-learned approaches are emerging that can outperform humans. However, in robotics, safety is a crucial factor of any new system, which is a major reason learning-based approaches are difficult to integrate into finished products. Splitting problems into smaller subtasks allows for safety checks to be added in between and allows for more understandable and controllable behavior. One downside of this paradigm is that improvements to individual components do not directly translate to improvements in the entire system. For example, improving the descriptor recall in *SegMap* did not directly lead to an improvement in localization. Still, parameters for the search tree and RANSAC needed to be tuned to match the new module. Similarly, in `maplab 2.0`, changing to deep-learned descriptors required heavy changes in the configuration of the parameters of the entire system. While the modular approach allows for easy changes, modules can not always be replaced without considering ramifications further in the system. End-to-end systems require less tuning, as they are trained to directly produce the desired output from raw data. However, they are also more unpredictable, especially in new environments, and can often produce very erroneous results.

A problem with deep-learned approaches is the need for data, especially ground truth labels, for training. Descriptors have seen such popularity and so many approaches since self-supervised losses are easier to formulate. One area of deep learning we found particularly limiting to our approach was object detection and segmentation. While the methods themselves can have very good accuracy, a limiting factor was the labels provided in datasets. For example, the popular COCO dataset [54] for object detection has only a handful of household or outdoor objects that are presumed static. Of interest, for example, would be more permanent objects such as windows, doors, cupboards, shelves, and cabinets for indoors. Similarly, for outdoor scenes, the ability to segment out building or house instances would be very useful. We found that the lack of labeled data that interested us severely limited what we could do with existing CNNs.

3.2.2 *Future Outlook*

Performance The performance of deep-learned methods needs to be better taken into account during the design process and solutions focused on robotics applications. This is essential on smaller mobile platforms, such as micro aerial vehicles and AR devices, where the computational power is very limited. Especially for keypoint detection, state-of-the-art CNNs can run in real-time only on massive GPUs, and often for robotics applications, much smaller versions need to be trained or distilled. Similar efficiency issues are when looking at the descriptor sizes these networks generate that are impractically large (2MB per keypoint). By being more target-driven, *i.e.* designing with robotics in mind from the beginning, many deep learning-based solutions could be more easily deployable and more applicable to robotics. Similar issues exist between balancing performance with practical values.

Temporal Data One advantage of robotics is the availability of temporal data, *i.e.* instead of dealing with individual images or other sensor measurements, we have a continuous stream. However, oftentimes also, to simplify difficult problems, we tend to develop single-frame solutions. This leaves unexploited a vast amount of potential data. One issue is that temporal approaches, such as for example recurrent neural networks, are harder to train. For example, the descriptor network in *SegMap* and the miscalibration detection network could greatly benefit from added temporal information. The addition of temporal information would hopefully significantly reduce the chances of spurious miscalibration detection.

New Datasets As we mentioned in the previous section, one limiting factor was the labels in datasets. Namely, many objects of interest were difficult to detect. However, some of them were difficult also due to perceptual reasons. For example, cameras can not detect windows but see the environment behind them, which creates a conflict that classic CNN approaches can not resolve. Both an increase in datasets labeled for robotics purposes would greatly benefit research and the addition of multi-modality to datasets. For example, an object detection dataset with equally rich labels and annotations as COCO [54], but also with LiDAR, would be of great interest and would have ramifications in many areas of robotics, including mapping.

Text-based Mapping While we focus on including semantics in mapping, we have ignored one of the more important cues humans sometimes follow, namely text. We rely on text in difficult environments, such as repetitive office buildings or cities. A simple example of perceptual aliasing that happens is between similar floors in an office building or hotel, for example, where the text on doors would immediately resolve the ambiguity. Existing approaches have been shown only in very limited environments [55, 56]. The development and robustness of text-recognition tools in the wild still need to improve, but the usefulness of text in robotics is still a domain that remains largely unexplored.

Generalizing Miscalibration While our work focuses on detecting when monocular cameras are miscalibrated, it is rarely true that a robotic platform has only one sensor. A future tool for detecting miscalibration could hopefully leverage information from other sensors to detect when either one is miscalibrated. Especially inertial measurement units (IMUs) are cheap and can be mounted in parallel with any sensor to ensure a redundant modality. We hope that adding other sensors would simplify the problem allowing for a network that generalizes better and maybe does not need to be specialized for each camera.

Part A

MODULAR MAPPING

PAPER I

MAPLAB 2.0 – A MODULAR AND MULTI-MODAL MAPPING FRAMEWORK

Andrei Cramariuc*, Lukas Bernreiter*, Florian Tschopp*, Marius Fehr, Victor Reijgwart, Juan Nieto, Roland Siegwart, Cesar Cadena

*contributed equally

ABSTRACT

Integration of multiple sensor modalities and deep learning into Simultaneous Localization And Mapping (SLAM) systems are areas of significant interest in current research. Multi-modality is a stepping stone towards achieving robustness in challenging environments and interoperability of heterogeneous multi-robot systems with varying sensor setups. With `maplab 2.0`, we provide a versatile open-source platform that facilitates developing, testing, and integrating new modules and features into a fully-fledged SLAM system. Through extensive experiments, we show that `maplab 2.0`'s accuracy is comparable to the state-of-the-art on the HILTI 2021 benchmark. Additionally, we showcase the flexibility of our system with three use cases: i) large-scale (~10 km) multi-robot multi-session (23 missions) mapping, ii) integration of non-visual landmarks, and iii) incorporating a semantic object-based loop closure module into the mapping framework. The code is available open-source at `https://github.com/ethz-asl/maplab`.

Published in:
IEEE Robotics and Automation Letters, 2023
DOI: 10.1109/LRA.2022.3227865

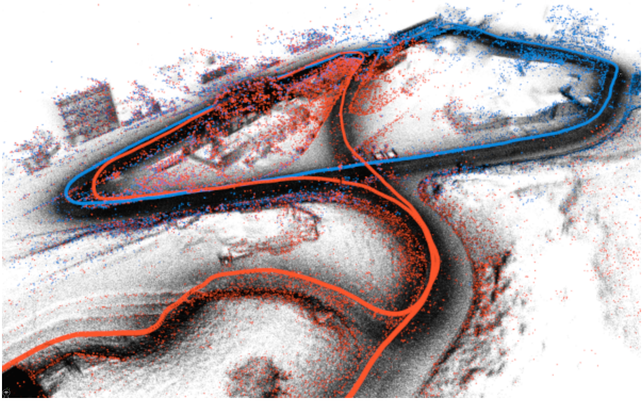


Figure 4.1: We propose `maplab 2.0`, a flexible and generic multi-robot, and multi-modal framework. `maplab 2.0` can seamlessly integrate multiple robots (colored paths), visual landmarks (colored points), and LiDAR scans (black points).

1 INTRODUCTION

Simultaneous Localization And Mapping (SLAM) is an essential component for various robotic applications, such as autonomous driving [1], mobile manipulation [2], and augmented/mixed reality. In these applications, the robotic platform needs to be aware of the surrounding environment and its location to perform the given task, be it driving autonomously to a particular destination or picking up and delivering an object. One step further is the ability to perform long-term mapping, which typically requires tools for processing and merging multiple maps enabling an even wider range of diverse applications and tasks.

Over recent years, many tailored SLAM solutions have been successfully developed for specific environments or sensor configurations [7–15]. However, many challenges remain until SLAM is fully solved or generically deployed in ubiquitous operating conditions. Recent efforts in fusing multiple modalities have gained significant traction due to the ability of multi-modal systems to compensate for weaknesses in individual sensors or methods. Thus, enabling a more robust robotic operation in degraded environments and even with full or partial sensor failures. Works combining many different sensors exist [7–9] and achieve remarkable performance. However, together with other open-source SLAM frameworks [12–15], these systems are tightly integrated. More specifically, they function only with specific sensor configurations, and the basic modules (*e.g.*, odometry, localization, or feature extraction) are highly entangled. Modifying those modules or incorporating new functionalities requires significant engineering work, adding major overheads to scientific research and the development of new products. Therefore, versatile systems that can seamlessly integrate various sensor setups and leverage multiple sensor modalities are desirable. Flexible support of multiple modalities is also the stepping stone for heterogeneous multi-robot systems, where different robots can be equipped with varying combinations of sensors, *e.g.*, due to platform constraints.

`maplab 2.0` provides an open-source platform for multi-session, multi-robot, and versatile multi-modal mapping. The original *maplab* [15] was an open-source toolbox for creating and managing exclusively visual-inertial maps. With `maplab 2.0`, we extend the original framework far beyond its initial scope by integrating multiple new modalities such as LiDAR, GPS receivers, wheel encoders, semantic objects, and more. These examples provide the templates for easy extension to further sensing modalities. `maplab 2.0` also offers interfaces for easy integration of external components, such as adding any number of different visual features or loop closure constraints. These features make our new platform ideally suited as a development and research tool for deep-learned keypoint detectors and loop closure engines that, until now, have mostly been tested separately [22]. Additionally, online collaborative SLAM is now possible in `maplab 2.0` due to the new submapping capabilities, enabling online building, optimization, and co-localization of one global map from multiple sources. This is made possible by our implementation of a new centralized server node that aggregates the data from multiple robots and can transmit the collaboratively built map back to the robots for increased performance [57]. We showcase the capabilities and performance of our system in multiple experiments and datasets, providing proof of concept implementations for non-visual keypoints, deep-learned descriptor integration, and a semantic object-based loop closure engine. Our contributions can be summarized as follows:

- We provide an open-source, multi-modal, and multi-robot mapping framework that allows integration and fusion of an unparalleled amount of different data compared to other existing methods.

- An online collaborative mapping system that utilizes submapping and a central server to compile and distribute globally consistent, feature-rich maps.

- Integration of interfaces for any number of custom feature points, descriptors, and loop closure. We showcase their flexibility in experiments featuring 3D LiDAR keypoints and semantic object-based loop closures.

2 RELATED WORK

Mapping can be defined as the challenge of creating environment representations and has seen a vast and diverse range of solutions over the past decades, with significant changes driven by new sensors and scenarios [58]. Multimodality has evolved beyond standard sensor fusion (*i.e.*, visual-inertial or stereo cameras) to include more complex combinations involving, for example, LiDARs and semantic information. Another notable topic is multi-robot mapping, where multiple robots simultaneously explore an environment and aim to create one globally consistent map. Multi-robot mapping differs from multi-session mapping, which involves collecting measurements of the same place at separate time intervals and enabling offline operations to and between sessions. While multi-robot frameworks can be used in a multi-session manner by sequentially processing data recordings, this is inefficient as it requires reprocessing all previous data any time a new recording is added due to their lack of map management tools. A comparison of significant SLAM frameworks and their features is presented in Table 4.1.

The first version of *maplab* [15] is a multi-session mapping framework designed for visual-inertial systems. Other comparable frameworks are ORB-SLAM3 [12] and RTAB-Map [7]. ORB-SLAM3 is an extension of its predecessor ORB-SLAM2 [59], adding support for an IMU and multi-session mapping capabilities. RTAB-Map integrates vision and depth measurements from a LiDAR or an RGB-D camera. An extension [60] to RTAB-Map supports a variety of handcrafted visual features and SuperPoint [22], but does not allow for easy integration of other descriptors. Both frameworks offer similar map creation and management features as *maplab*, with the addition of online loop closure and optimization during mapping. All three of the aforementioned frameworks are tightly integrated systems designed for a particular sensor configuration. On the contrary, we allow easy integration of different sensor setups, visual features, and support an arbitrary odometry input in `maplab 2.0`, which facilitates the use of heterogeneous robots and provides a new level of flexibility.

Kimera [8] is a multi-modal mapping framework that provides both local and global 3D meshes with semantic annotations and a global trajectory estimate based on visual-inertial SLAM. As opposed to `maplab 2.0`, Kimera

Table 4.1: Comparison of supported features in state-of-the-art mapping frameworks. (*Diff. sensors:* Maps with different sensor configurations can be combined; *Ext.*: External source; *LC.*: Loop closure; $*$: IMU biases have to be provided alongside the poses)

	Multi-Modal	**Multi-Robot**	**Multi-Session**	**Online**	**Diff. Sensors**	**Ext. Odometry**	**Ext. Features**	**Ext. LCs**	**GPS support**	**Semantic LCs**	**Open-Source**
RTAB-Map [7]	✓		✓	✓		✓					✓
ORB-SLAM3 [12]	✓		✓	✓							✓
LAMP 2.0 [10]	✓	✓		✓	✓	✓					✓
CVI-SLAM [11]		✓		✓							
COVINS [13]		✓		✓		$*$					✓
DOOR-SLAM [14]	✓	✓		✓		✓					✓
Kimera [8]	✓			✓		$*$					✓
Kimera-Multi [9]	✓	✓		✓		$*$					✓
maplab [15]			✓								✓
`maplab 2.0`	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

does not have multi-session capabilities, and the 3D reconstruction with its semantic annotations is not used to improve the accuracy of the SLAM estimates. In general, semantic information has the potential to significantly improve mapping [58] by being a catalyst for high-level scene understanding. However, previous works utilizing semantics for mapping [6, 46, 61–63] focus mainly on generating improved descriptors rather than leveraging them in a fully semantic SLAM system. In this work, we propose utilizing image descriptors and a simple semantic object representation, which allows us to optimize using well-known relative pose errors.

Kimera-Multi [9] is a direct extension of Kimera [8], which enables the multi-robot scenario using a fully distributed system but does not improve on the purely visual-inertial SLAM backend in the original Kimera. Our approach instead uses a centralized server that collects submaps, optimizes them, and creates one globally consistent map. A similarly centralized setting was also explored by COVINS [13]. However, COVINS is limited to the visual-inertial use case, while `maplab 2.0` can incorporate multiple sensor modalities and configurations. In a similar vein are also LAMP 2.0 [10], CVI-SLAM [11], and DOOR-SLAM [14], which offer collaborative mapping

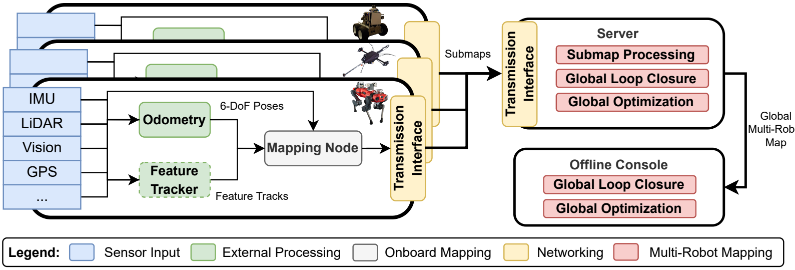


Figure 4.2: Overview of the `maplab 2.0` framework and its three main components, namely the mapping node, the centralized server, and the offline console. The mapping node runs on each robot and collects the sensor data into submaps which are passed to the centralized server that merges them into a globally consistent map. This map can then, at later stages, be refined or merged with other maps using the tools provided by the offline console. This figure showcases an example of one possible configuration of how `maplab 2.0` can be used. There are multiple other combinations as well as inputs and modules that can be added or excluded as described in Section 3.

between robots but are tightly integrated systems, limited to one sensor modality, and have little flexibility.

Although various other SLAM frameworks exist, they are mainly focused on specific sensor or robot-environment configurations, and changes to either one are usually difficult or impossible. To our knowledge of all the existing methods, `maplab 2.0` is the most flexible mapping and localization framework that not only supports a variety of sensors but can also be seamlessly adapted to new needs.

3 THE `MAPLAB 2.0` FRAMEWORK

The general structure of the `maplab 2.0` framework is presented in Figure 4.2. The entire framework can be divided into three main components: the mapping node, the mapping server, and the offline console interface. We begin with an overview of the underlying map structure in `maplab 2.0`, after which we discuss in more detail the main components.

3.1 *Map Structure*

We denote a *map* as a collection of one or more *missions*, where each mission is based on a single continuous mapping session. The underlying structure of a map is a factor graph consisting of vertices and edges that incorporate all the robot information and the measurements across different missions. The state of the robot at a certain point in time t is parameterized as a vertex (6 degrees of freedom (DoF) pose, velocity, IMU biases). Landmarks are also represented as vertices in the graph whose state is defined as a 3D position. The 3D landmarks can be used as an underlying representation for anything in the environment with a 3D position, *e.g.*, visual landmarks, 3D landmarks, or even semantic objects.

Constraints

Vertices are connected through different types of edges that impose constraints on their state variables based on observations (*e.g.*, keypoints, imu measurements, and loop closures). IMU edges contain the pre-integrated IMU measurements between connected vertices and therefore only connect temporally sequential vertices. Relative pose constraint edges impose a rigid 6 DoF transformation between two vertices and are used to represent either relative motion (*i.e.*, odometry) or loop closures across larger temporal gaps or missions. The edges are assigned a covariance to quantify the measurement noise, which is typically set to a predefined constant value. The covariance can be used to model the degrees of motion a sensor can observe, *e.g.* wheel odometry has infinite covariance for motion along the z-axis, as well as pitch and roll. We consider loop closure edges a special case of relative pose constraint edges. For increased robustness and to account for outliers, loop closure edges can be included as switchable constraints [64]. The optimizer can then discard edges from the graph if they conflict too much with the other constraints. Finally, edges connect a landmark to the poses from which it was observed and impose an error based on the difference between the estimated and observed landmark position.

During optimization, constraints can also be imposed directly on the internal states of chosen vertices. For example, absolute constraints enforce a global 3D position on a vertex with a given uncertainty and allow us to integrate GPS measurements or absolute fiducial marker observations. Additionally, fixing certain states enables the flexibility of choosing which parts of the problem are to be optimized.

Landmarks

The visual mapping module at the core of *maplab* [15] is still a part of `maplab 2.0`. It includes feature detection based on ORB [65], with binary descriptors from either BRISK [66] or FREAK [67]. Feature correspondences between consecutive frames are established based on descriptor matches, where for robustness, the matching window is restricted by integrated gyroscope measurements. These feature tracks are then triangulated into 3D landmarks.

Global localization and loop closure is done by taking individual frames and establishing a set of 2D-3D matches using the feature descriptors. A covisibility check is applied afterward to the matches to filter outliers. Then, with a P3P algorithm within a RANSAC scheme, the remaining matches are used to obtain a transformation with respect to the map's reference frame. This transformation can then be added to the factor graph as a loop closure edge. We also provide an alternative method that incorporates loop closures by merging the covisible landmarks and minimizing their reprojection error. This approach foregoes the difficulty of tuning explicit loop closure edge covariances but enforces softer constraints on the factor graph.

In `maplab 2.0`, we have added the possibility of concurrently including any number of different types of features into the map. To obtain feature tracks across consecutive frames, users can either use the included generic implementation of a Lucas–Kanade tracker [68] or supply the track information themselves. In addition, we expanded the matching engine to support floating point descriptors, enabling loop closure using the latest developed descriptors. Binary descriptors are matched, as in *maplab*, using an inverted multi-index [69], while floating point descriptors are matched using a Fast Library for Approximate Nearest Neighbors (FLANN) [70]. Matches are then treated similarly for the purpose of loop closures as previously described. Still, for tuning purposes, different feature types can have separate parameter sets to account for differences in quality and behavior.

`maplab 2.0` can also handle landmarks with 3D observations. These could originate from, for example, RGB-D cameras, where visual features also have an associated depth, or from features detected directly in a 3D point cloud. The significant difference is that the position of these landmarks is not triangulated using multi-view geometry but by averaging the 3D measurements. Similarly, the pose graph error term is not based on the reprojection error, but on the Euclidean distance between the observed 3D position and the 3D position of the landmark. The other significant difference is that loop closure is set up as a 3D to 3D RANSAC matching problem without the P3P algorithm.

3.2 *Mapping Node*

The mapping node runs onboard each robot and uses external input sources and the raw sensor data to create a map in the form of a multi-modal factor graph. A 6 DoF odometry input is used during map building to initialize the robot pose vertices for the underlying factor graph. The mapping node is agnostic to the odometry method and features a simple interface, thus enabling its easy use across various robots and sensor setups. This is in contrast to *maplab* [15], where only the built-in visual-inertial estimator ROVIOLI [5] was available – whereas `maplab 2.0` does not even require an IMU. However, if an IMU is available, inertial constraints are added to the map, and the state estimator can optionally also compute an initial estimate for the IMU biases. These bias estimates can then be used to improve the initialization of the global map optimization problem, which benefits its convergence speed and accuracy. If an IMU is present but not used by the state estimator, the bias estimation can also be done separately [71–73].

Substantial changes to the original *maplab* framework were also made such that other sensor modalities can be processed and integrated using custom internal components or easily configurable external interfaces. Most notably, `maplab 2.0` can incorporate any number of different 3D landmarks types at runtime. Furthermore, relative constraints (*e.g.*, odometry or external loop closures) and absolute 6 DoF constraints (*e.g.*, GPS or fiducial markers) can now seamlessly be added.

The raw camera images or the LiDAR point clouds can be attached to the map as resources that later modules can use at any time, for example, to compute additional loop-closures or detect objects. The resulting maps with all the included constraints can then be passed on to the mapping server for online processing or stored and loaded for later offline processing in the console.

3.3 *Mapping Server*

The mapping server is a new addition to `maplab 2.0`, enabling collaborative and online mapping. This method was successfully deployed in the DARPA Subterranean Challenge as part of the winning team's (CERBERUS) multi-robot mapping system [74]. The server node can run on a dedicated machine or one of the robots in parallel with the mapping node. The mapping nodes divide their maps into chunks, called submaps, at regular intervals. The submaps are immediately transmitted to the mapping server where they are preprocessed and concatenated to the corresponding previously transmitted submap from the same robot. Bookkeeping is done by duplicating the last vertex of each submap into the next submap when splitting. This also avoids discontinuities in edges and feature tracks. In parallel the server continuously loop closes maps from different robots into a globally consistent map. Notably, the server and console share the same code base, therefore any new functionalities can easily be integrated into either one.

Submap Preprocessing

The incoming submaps are not merged directly but rather first processed individually to ensure local accuracy. Specifically, a configurable set of operations is executed on each robot's submap, which includes local map optimization (full bundle adjustment over all sensor data and constraints), feature quality evaluation, and intra-map loop closure (visual and LiDAR depending on what is available). Since the submaps are processed independently of each other, the mapping server can efficiently process multiple maps concurrently. After completion of the preprocessing steps, each finished submap is concatenated to the previous submap from the same robot.

Multi-Robot Processing

The mapping server continuously operates on the global multi-robot map and executes a second set of configurable operations (loop closure, feature quality evaluation, bundle adjustment, visualizations, absolute constraint outlier rejection, *etc.*). Here, the loop closure algorithms (visual or LiDAR) try and place all the different robots into the same reference frame and correct drift. In contrast to the preprocessing step, the operations on the multi-robot map are always performed at a global scope, *e.g.*, loop closures are detected in an intra- and inter-robot approach, and the global optimization is done jointly over all robot maps. The collaboratively built global map can also be transmitted back to the robots to increase the accuracy of their onboard estimation [57]. The increased awareness of the environment not only benefits localization accuracy but also other tasks such as global path planning.

3.4 *Offline Console*

The offline console was ported over from *maplab*, with old tools adapted to the new features regarding sensors and modalities. There are tools for further processing maps, such as batch optimization, merging maps from different sessions, outlier rejection, key-framing, map sparsification [50], *etc.* Loop closure using LiDAR is also now possible with a new module that includes an implementation of ICP [75] and G-ICP [76] but is not limited to these and can easily be extended. Transformations computed by the registration module are added as loop closure edges with switchable constraints (see Section 3.1). For each sensor and method combination we use a predefined fixed covariance, which is set separately for each translation and rotation component. The values are empirically chosen based on the sensor noise and the accuracy of the registration method. Dense reconstruction can also be done using the integrated Voxblox [31] plugin. The console additionally provides tools for resource management (manipulating or visualizing the attached point clouds, images, and semantic measurements) or exporting map data (poses, IMU biases, landmarks, *etc.*).

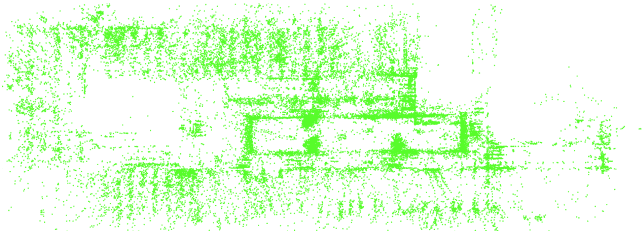
Finally, the console enables easy extensions through plugins that can run code offline and are independent of the map-building process. We used plugins, for example, to implement the LiDAR registration module mentioned above and a semantic loop closure module (see Section 4.4).

4 USE-CASES

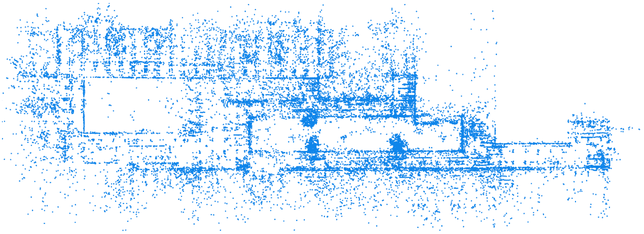
We conducted several experiments to evaluate our proposed framework thoroughly and to demonstrate its ease of use and high flexibility. Specifically, this section presents results on four datasets to showcase the new features and capabilities of `maplab 2.0`. First, we validate the performance and accuracy of our proposed framework on the public HILTI SLAM 2021 dataset [20] and compare it to well-known state-of-the-art approaches. Next, we demonstrate the real-world applicability of our proposed framework and showcase the large-scale multi-robot multi-session capabilities. Then, we show the versatility of the landmark system by incorporating 3D LiDAR features detected from projected point clouds. Finally, we showcase a semantic loop closure module on a custom indoor dataset. All datasets are collected with hardware time-synchronized sensor setups.

4.1 *Validation and Comparison*

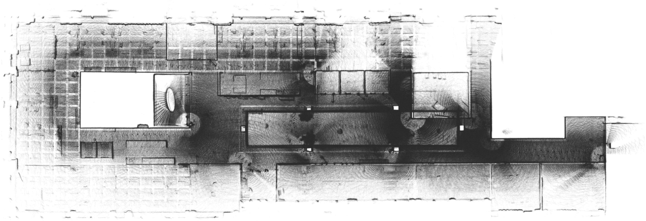
We use the HILTI SLAM Challenge 2021 dataset [20] to compare our proposed framework to state-of-the-art approaches. The dataset includes 12 recordings covering indoor office environments and challenging outdoor construction sites. In our experiments we exclude three sequences that are too small



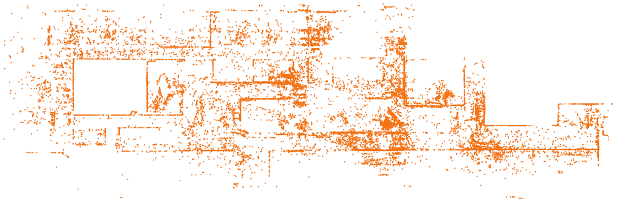
(a) BRISK descriptors with ORB detector.



(b) SuperPoint visual features with SuperGlue tracking.



(c) Accumulated point cloud reconstruction.



(d) SuperPoint on LiDAR images with SuperGlue tracking.

Figure 4.3: Visualizations of the features and sensor data in `maplab 2.0` on the Office Mitte sequence from the HILTI 2021 dataset, using OKVIS and global bundle adjustment over the features.

and do not present interesting challenges. Three pairs (Construction Site, Basement, and Campus) of the remaining nine sequences were taken in the same environment and can be co-localized to increase accuracy.

For `maplab 2.0`, we can use the five cameras, the ADIS IMU and the OS0-64 LiDAR provided in the dataset. We show three use-cases for `maplab 2.0` using three different odometry sources: ROVIO [5], OKVIS [16], and FAST-LIO2 [18]. Besides the standard BRISK [66] descriptors, we use the external interfaces from Section 3.1 to also include SuperPoint [22] features with SuperGlue [25] tracking, and SIFT features [77] with LK tracking [68]. To reduce map size and speed up the descriptor search, we compress the SuperPoint and SIFT features using principal component analysis (PCA) from 256 floating points to 32. Global loop closures are computed using all available features, and matched landmarks are merged. We also use ICP [75] from the LiDAR registration module (see Section 3.4) to refine our poses locally. Covariances for the loop closure edges are predefined empirically. Visualizations from sequence *Office Mitte* are presented in Figure 4.3a-c. It can be observed how SuperPoint features better follow the building structure compared to ORB.

Table 4.2 also shows the performance of the odometry sources alone, and other SLAM baselines (LVI-SAM [78], ORB-SLAM3 [12], RTAB-Map [7], and *maplab* [15]). *Maplab* and `maplab 2.0` are the only methods able to use all five cameras for loop closures. For ROVIO and OKVIS we only use the frontal camera or stereo pair for odometry. Among all methods that use vision `maplab 2.0` outperforms the baselines by a significant margin. FAST-LIO2, which uses only LiDAR-Inertial, is the best baseline, outperforming even LVI-SAM, which is a vision-LiDAR-inertial fusion based on the same principles. However, we show that we can also take the best performing method as odometry and further refine the result, especially improving significantly on the *Parking* sequence over FAST-LIO2. We also present a fusion of ROVIO and SIFT, demonstrating the versatility of `maplab 2.0` for fast incremental improvements, independently of better deep-learned visual features. Timings for all methods are also presented on a machine with an Intel i7-8700 and an Nvidia RTX 2080 GPU.

4.2 *Large-Scale Multi-Robot Multi-Session Mapping*

We demonstrate the applicability toward complex real-world scenarios by deploying our proposed framework in a large-scale training facility in Switzerland. The environment features urban-like streets with buildings and harsh

Table 4.2: Comparison of state-of-the-art methods in terms of the RMSE of the absolute position error (APE). SP + B represents SuperPoint and BRISK visual features. The icons represent the utilized sensors: monocular ▯, multi-camera ▥, LiDAR ⁙, and IMU ⅄. The total duration of the dataset is 52 minutes.

HILTI 2021 SLAM Dataset											
								`maplab 2.0`			
Sequence	**ORB SLAM3**	**LVI SAM**	**RTAB Map**	**maplab**	**ROVIO**	**OKVIS**	**FAST LIO2**	ROVIO + SIFT	OKVIS + SP + B	OKVIS + ICP	FAST-LIO2 + SP + B
	▥⅄	▥⁙⅄	▥⁙⅄	▥⅄	▯⅄	▥⅄	⁙⅄	▥⅄	▥⅄	▥⁙⅄	▥⁙⅄
Construction 1	1.55 m	0.13 m	0.36 m	0.16 m	0.98 m	1.17 m	**0.04 m**	0.14 m	0.08 m	0.08 m	**0.04 m**
Construction 2	2.77 m	0.33 m	0.67 m	0.57 m	1.50 m	2.13 m	**0.07 m**	0.34 m	0.19 m	0.19 m	**0.07 m**
IC Office	1.86 m	0.12 m	1.50 m	0.09 m	1.16 m	1.27 m	0.08 m	0.08 m	0.08 m	**0.07 m**	**0.07 m**
Office Mitte	1.70 m	0.24 m	0.94 m	3.18 m	0.86 m	1.15 m	0.12 m	0.27 m	0.18 m	0.15 m	**0.10 m**
Basement 3	1.55 m	0.10 m	0.38 m	0.09 m	3.05 m	1.01 m	**0.05 m**	0.09 m	0.09 m	0.08 m	**0.05 m**
Basement 4	1.71 m	0.13 m	0.38 m	0.11 m	2.90 m	1.23 m	**0.04 m**	0.11 m	0.10 m	0.09 m	**0.04 m**
Parking	5.49 m	4.43 m	7.82 m	0.39 m	6.13 m	3.36 m	5.00 m	0.31 m	**0.21 m**	**0.21 m**	**0.21 m**
Campus 1	1.93 m	0.12 m	0.93 m	0.60 m	5.10 m	2.41 m	**0.07 m**	0.38 m	0.19 m	0.17 m	**0.07 m**
Campus 2	2.24 m	0.14 m	0.79 m	0.47 m	2.02 m	2.23 m	0.09 m	0.28 m	0.20 m	0.18 m	**0.08 m**
Total Time	61 min	68 min	163 min	82 min	58 min	121 min	52 min	98 min	236 min	267 min	194 min

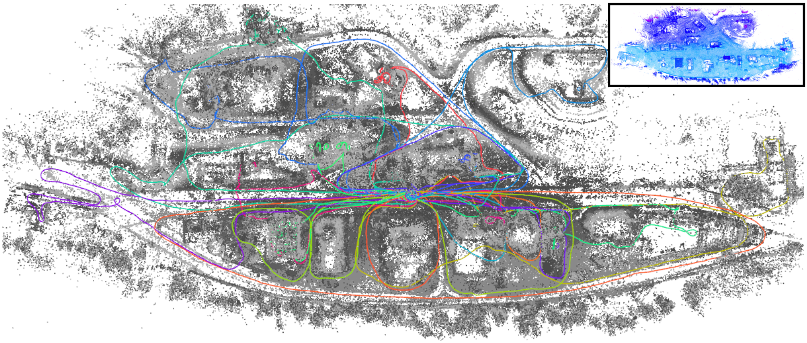
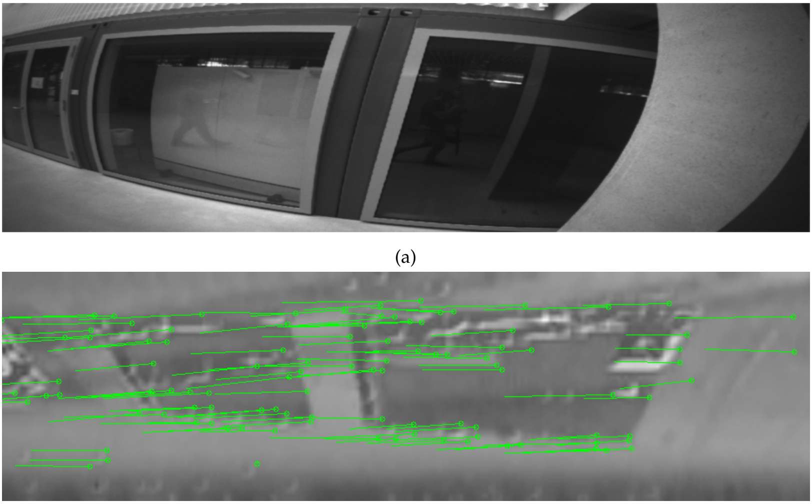


Figure 4.4: Visual mapping results of the global multi-robot map comprising 23 mapping runs. Individual colors denote robot trajectories and gray points denote triangulated BRISK landmarks in the multi-robot map. The top right image shows the LiDAR map by reprojecting the point clouds onto the optimized poses.

environments such as collapsed buildings with narrow spaces. For this experiment, we recorded 23 individual runs with a handheld device with five cameras and an Ouster OS0-128 comprising more than two hours of data over approximately 10 km and multiple indoors-outdoors transitions. Each run used OKVIS [16] for odometry. The first five maps were used to build a global multi-robot map using the mapping server, and the remaining maps were merged using multi-session mapping in the console. Consistency between all missions was enforced using global visual loop closures, and if available, additional absolute pose constraints from an RTK GPS. Moreover, individual trajectories were refined by performing intra- and inter-mission LiDAR registrations. Figure 4.4 shows the final multi-robot map.

To quantitatively evaluate the multi-robot server we test on the public EuRoC benchmark [21]. As the console and server use the same underlying mapping framework, excluding minor details such as operation ordering, the expected accuracy is the same. We run all 11 sequences simultaneously in a multi-robot experiment using the mapping server, with ROVIO and BRISK. Afterwards, we repeat the experiment by sequentially processing each mission using the mapping node and console and then merging them together. Both scenarios achieve an average RMSE APE of 0.043 m, but the parallelized mapping server only takes 3 min 27 s for everything, as opposed to 35 min 56 s for the sequential multi-session workflow. For both scenarios the timings include the odometry, optimization and map merging.



(a)

(b)

Figure 4.5: Tracking keypoints on LiDAR images. (a) shows a related camera image only for illustration purposes. (b) shows the LiDAR image (cropped for visualization $\sim 40°$) where the green circles and lines represent SuperPoint detections along with their tracked motion to the previous frame.

4.3 *Visual Tracking on Projected LiDAR Images*

To showcase the flexibility of the landmark system in `maplab 2.0`, we integrate 3D LiDAR keypoints[1]. We draw inspiration from the work of Streiff *et al.* [79] and project the LiDAR point cloud onto a 2D plane. We normalize the LiDAR range and intensity values using a logarithmic scale and merge the two channels using Mertens fusion [80]. Missing pixels from bad LiDAR returns are in-painted using neighboring values. An example image of the resulting 2D projection is shown in Figure 4.5, alongside a camera image from the same perspective showing the environment. We then treat the LiDAR image like a camera image and apply SuperPoint with SuperGlue to obtain point features and tracks, as shown in Figure 4.5. Since, for each feature observation, we have depth information from the LiDAR, we can more efficiently initialize and loop close these 3D LiDAR landmarks, as described in Section 3.1. Finally, we use these LiDAR keypoints to map out one of

---

[1] Please note that a similar workflow could be implemented for other modalities, *e.g.*, RGB-D cameras, radar or sonar imaging.

the sequences in the HILTI 2021 dataset and visualize the resulting map in Figure 4.3d. The LiDAR landmarks are mapped more accurately onto the structure than the visual keypoints, as seen from the straightness of the walls. However, they also suffer from outliers caused by noise in the LiDAR image from missing points or moving objects in the environment.

4.4 *Semantic-based Mapping*

This section showcases the extensibility and modular design of `maplab 2.0` by augmenting the map with semantic information and illustrating its potential application in a real-world scenario. Initially, semantic objects are detected in an image using Mask R-CNN [81], and for each detection, we use NetVLAD [82] to extract a descriptor on the masked instance segmentation. Instead of the built-in tracker, all detected objects are tracked using Deep SORT [83], which extends typical spatial data association metrics with an appearance term that can directly utilize the previously extracted object descriptor. Similar to visual landmarks, semantic objects are 3D landmarks in the `maplab 2.0` map but have an associated class label and can be used for, *e.g.*, semantic loop closure detection.

Finally, candidate semantic loop closures are found by directly comparing the object descriptors of the same class. First, a unique visibility filter is applied, *i.e.*, two landmarks observed in a single image cannot be matched. After geometrically verifying the candidates and clustering co-visible landmarks, a 6-DoF constraint between the two robot vertices closest to two matched landmark clusters is constructed using the relative coordinate transformation between two 3D landmark clusters, obtained through Horn's method [84]. Finally, the covariance of the corresponding factor-graph constraint is calculated using the method proposed by Manoj *et al.* [85].

We collected an indoor dataset in an office environment with multiple objects using an RGB-inertial sensor [86]. We observe an office table with objects on two occasions while leaving some time to accumulate drift (see Figure 4.6a and 4.6c). Figure 4.6b shows semantic landmark clusters and detected loop closure candidates. After adding the loop closure edges from the semantic objects to the full factor graph, the drift significantly reduces, and an improved map can be seen in Figure 4.6c.

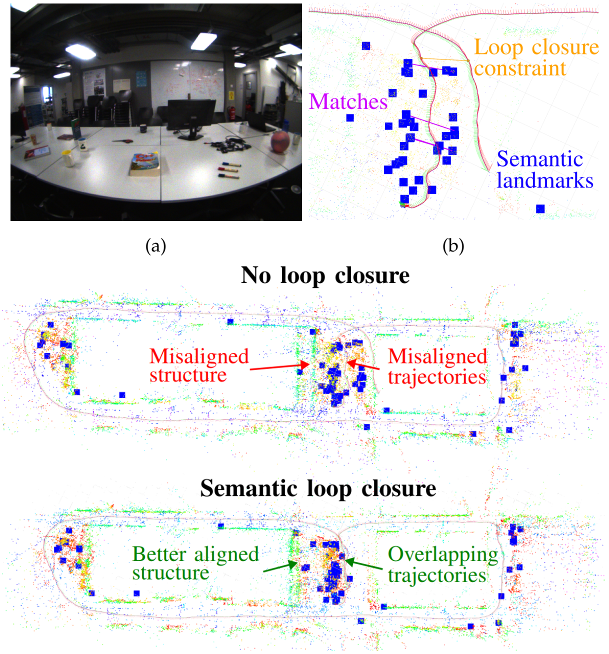


Figure 4.6: Semantic mapping pipeline. (a) Experimental setup: a table with multiple semantic objects. (b) Loop closure matches (magenta) between semantic landmarks (blue), resulting in a loop closure constraint (orange). (c) Visual-inertial-semantic map before and after semantic loop closure.

5 CONCLUSION

We presented a research platform for multi-modal and multi-robot mapping, supporting online and offline processing of the maps. We showcase state-of-the-art performance on a large-scale SLAM benchmark and multiple experimental use cases for `maplab 2.0`. Our proposed mapping framework's flexible and modular design facilitates research in various robotic applications and yields important implications in academia and industry. The code and tutorials to reproduce the experiments are available on the wiki of the repository.

Part B

MODULAR LEARNING

PAPER II

SEGMAP: SEGMENT BASED MAPPING AND LOCALIZATION USING DATA-DRIVEN DESCRIPTORS

Renaud Dubé*, Andrei Cramariuc*, Daniel Dugas, Hannes Sommer, Marcin Dymczyk, Juan Nieto, Roland Siegwart, and Cesar Cadena

*contributed equally

ABSTRACT

Precisely estimating a robot's pose in a prior, global map is a fundamental capability for mobile robotics, *e.g.* autonomous driving or exploration in disaster zones. This task, however, remains challenging in unstructured, dynamic environments, where local features are not discriminative enough and global scene descriptors only provide coarse information. We therefore present *SegMap*: a *map representation* solution for localization and mapping based on the extraction of segments in 3D point clouds. Working at the level of segments offers increased invariance to view-point and local structural changes, and facilitates real-time processing of large-scale 3D data. *SegMap* exploits a single compact data-driven descriptor for performing multiple tasks: global localization, 3D dense map reconstruction, and semantic information extraction. The performance of *SegMap* is evaluated in multiple urban driving and search and rescue experiments. We show that the learned *SegMap* descriptor has superior segment retrieval capabilities, compared to state-of-the-art handcrafted descriptors. In consequence, we achieve a higher localization accuracy and a 6% increase in recall over state-of-the-art. These segment-based localizations allow us to reduce the open-loop odometry drift by up to 50%. *SegMap* is open-source available along with easy to run demonstrations.

Published in:
The International Journal of Robotics Research (IJRR), 2020
DOI: 10.1177/0278364919863090

1 INTRODUCTION

Mapping and localization are fundamental competencies for mobile robotics and have been well-studied topics over the last couple of decades ([58]). Being able to map an environment and later localize within it unlocks a multitude of applications, that include autonomous driving, rescue robotics, service robotics, warehouse automation or automated goods delivery, to name a few. Robotic technologies undoubtedly have the potential to disrupt those applications within the next years. In order to allow for the successful deployment of autonomous robotic systems in such real-world environments, several challenges need to be overcome: mapping, localization and navigation in difficult conditions, for example crowded urban spaces, tight indoor areas or harsh natural environments. Reliable, prior-free global localization lies at the core of this challenge. Knowing the precise pose of a robot is necessary to guarantee reliable, robust and most importantly safe operation of mobile platforms and also allows for multi-agent collaborations.

The problem of mapping and global localization has been well covered by the research community. On the one hand, a large body of algorithms use cameras and visual cues to perform place recognition. Relying purely on appearance has, however, significant limitations. In spite of tremendous progress within this field, state-of-the-art algorithms still struggle with changing seasons, weather or even day-night variations ([87]). On the other hand, several approaches address the variability of appearance by relying instead on the 3D structure extracted from LiDAR data, which is expected to be more consistent across the aforementioned changes. Current LiDAR-based Simultaneous Localization And Mapping (SLAM) systems, however, mostly use the 3D structure for local odometry estimation and map tracking, but fail to perform global localization without any prior on the pose of the robot ([88]).

There exist several approaches that propose to use 3D point clouds for global place recognition. Some of them make use of various local features ([89, 90]), which permit to establish correspondences between a query scan and a map and subsequently estimate a 6-degrees of freedom (DoF) pose. The performance of those systems is limited, as local features are often not discriminative enough and not repeatable given the changes in the environment. Consequently, matching them is not always reliable and also incurs a large computational cost given the number of processed features. Another group of approaches relies on global descriptors of 3D LiDAR scans ([91]) that permit to find a correspondence in the map. Global descriptors, however, are view-point dependent, especially when designed for only rotational-invariance and not as translation-invariant. Furthermore, a global

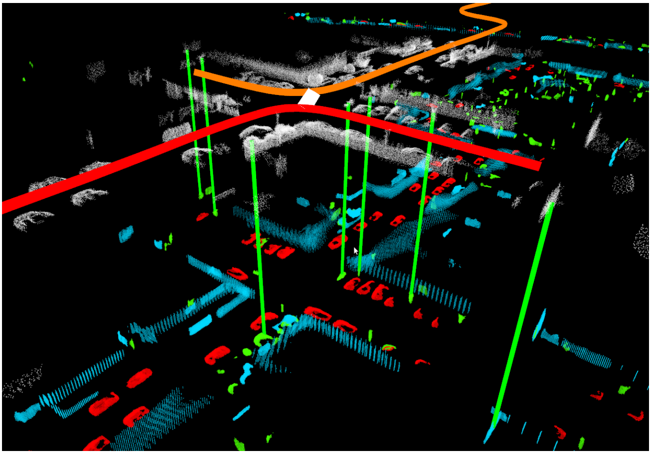


Figure 5.1: An illustration of the *SegMap* approach. The red and orange paths represent the trajectories of two robots driving simultaneously in opposite directions through an intersection. In white we show the local segments extracted from the robots' vicinity and characterized using our compact data-driven descriptor. Correspondences are then made with the target segments, resulting in a successful localization depicted with green vertical lines. A reconstruction of the target segments is illustrated below, where colors represent semantic information (cars in red, buildings in light blue, and others in green), all possible by leveraging the same compact representation. We take advantage of the semantic information by performing localization only against static objects, improving robustness against dynamic changes. Both the reconstruction and semantic classification are computed by leveraging the same descriptors used for global prior-free localization.

scan descriptor is more prone to failures under dynamic scenes *e.g.* parked cars, which can be important for reliable global localization in crowded, urban scenarios.

We therefore present *SegMap*[1]: a unified approach for *map representation* in the localization and mapping problem for 3D LiDAR point clouds. *SegMap* is formed on the basis of partitioning point clouds into sets of descriptive segments ([39]), as illustrated in Figure 5.2. The segment-based localization combines the advantages of global scan descriptors and local features – it offers reliable matching of segments and delivers accurate 6-DoF global localizations in real-time. The 3D segments are obtained using efficient region-growing techniques which are able to repeatedly form similar partitions of

---

[1] *SegMap* is open-source available along with easy to run demonstrations at `www.github.com/ethz-asl/segmap`. A video demonstration is available at `https://youtu.be/CMk4w4eRobg`

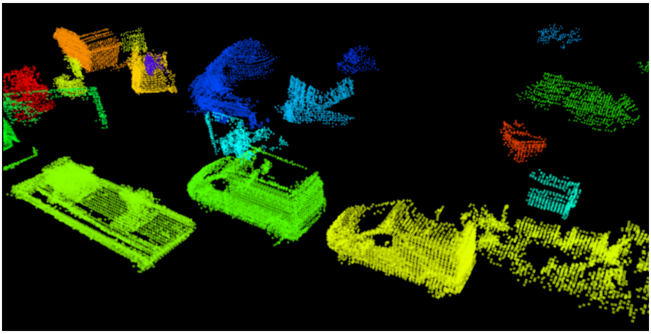


Figure 5.2: Exemplary segments extracted from 3D LiDAR data collected in a rural environment. These segments were extracted with an incremental Euclidean distance-based region-growing algorithm and represent, among others, vehicles, vegetation and parts of buildings ([92]).

the point clouds ([92]). This partitioning provides the means for compact, yet discriminative features to efficiently represent the environment. During localization global data associations are identified by segment descriptor retrieval, leveraging the repeatable and descriptive nature of segment-based features. This helps satisfy strict computational, memory and bandwidth constraints, and therefore makes the approach appropriate for real-time use in both multi-robot and long-term applications.

Previous work on segment-based localization considered hand-crafted features and provided only a sparse representation ([39, 93]). These features lack the ability to generalize to different environments and offer very limited insights into the underlying 3D structure. In this work, we overcome these shortcomings by introducing a novel data-driven segment descriptor which offers high retrieval performance, even under variations in view-point, and that generalizes well to unseen environments. Moreover, as segments typically represent meaningful and distinct elements that make up the environment, a scene can be effectively summarized by only a handful of descriptors. The resulting reconstructions, as depicted in Figure 5.1, can be built at no extra cost in descriptor computation or bandwidth usage. They can be used by robots for navigating around obstacles and visualized to improve situational awareness of remote operators. Moreover, we show that semantic labeling can be executed through classification in the descriptor space. This information can, for example, lead to increased robustness to changes in the environment by rejecting inherently dynamic classes.

To the best of our knowledge, this is the first work on robot localization that is able to leverage the extracted features for reconstructing environments in three dimensions and for retrieving semantic information. This reconstruction is, in our opinion, a very interesting capability for real-world, large-scale applications with limited memory and communication bandwidth. To summarize, this paper presents the following contributions:

- A data-driven 3D segment descriptor that improves localization performance.
- A novel technique for reconstructing the environment based on the same compact features used for localization.
- An extensive evaluation of the *SegMap* approach using real-world, multi-robot automotive and disaster scenario datasets.

In relation to the *Robotics: Science and System* conference paper ([94]), we make the following additional contributions:

- A comparison of the accuracy of our localization output with the results of recently published technique based on data-driven global 3D scan descriptors ([91]).
- An evaluation of trajectory estimates by combining our place recognition approach with a state-of-the-art 3D LiDAR-based SLAM technique ([17]).
- A triplet loss descriptor training technique and its comparison to the previously introduced classification-based approach.
- A particularly lightweight variant of our *SegMap* descriptor that can be deployed on platforms with limited computational resources.

The remainder of the paper is structured as follows: Section 2 provides an overview of the related work in the fields of localization and learning-based descriptors for 3D point clouds. The *SegMap* approach and our novel descriptor that enables reconstruction of the environment are detailed in Section 3 and Section 4. The method is evaluated in Section 5, and finally Sections 6 and 7 conclude with a short discussion and ideas on future works.

2 RELATED WORK

This section first introduces state-of-the-art approaches to localization in 3D point clouds. Data driven techniques using 3D data which are relevant to the present work are then presented.

Localization in 3D point clouds Detecting loop-closures from 3D data has been tackled with different approaches. We have identified three main trends: (i) approaches based on local features, (ii) global descriptors and (iii) based on planes or objects.

A significant number of works propose to extract local features from keypoints and perform matching on the basis of these features. [95] extract keypoints directly from the point clouds and describe them with a 3D *Gestalt* descriptor. Keypoints then vote for their nearest neighbors in a *vote matrix* which is eventually thresholded for recognizing places. A similar approach has been used in [96]. Apart from such Gestalt descriptors, a number of alternative local feature descriptors exist, which can be used in similar frameworks. This includes features such as Fast Point Feature Histogram (FPFH) ([89]) and SHOT ([90]). Alternatively, [97] transform the local scans into bearing-angle images and extract Speeded Up Robust Features (SURFs) from these images. A strategy based on 3D spatial information is employed to order the scenes before matching the descriptors. A similar technique by [98] first transforms the local scans into a range image. Local features are extracted and compared to the ones stored in a database, employing the Euclidean distance for matching keypoints. This work is extended in [99] by using Normal-Aligned Radial Features (NARF) descriptors and a bag of words approach for matching.

Using global descriptors of the local point cloud for place recognition is also proposed in ([100–103]). [100] propose to describe each local point cloud with a 1D histogram of point heights, assuming that the sensor keeps a constant height above the ground. The histograms are then compared using the *Wasserstein* metric for recognizing places. [101] describe point clouds with rotation invariant features such as volume, nominal range, and range histogram. Distances are computed for feature vectors and cross-correlation for histogram features, and an AdaBoost classifier is trained to match places. Finally, Iterative Closest Point (ICP) is used for computing the relative pose between point clouds. In another approach, [102] split the cloud into overlapping grids and compute shape properties (spherical, linear, and several type of planar) of each cell and combine them into a matrix of surface shape histograms. Similar to other works, these descriptors are compared for recognizing places. Recently, [103] proposed to leverage LiDAR intensity information with a global point cloud descriptor. A two-stage approach is adopted such that, after retrieving places based on global descriptors retrieval, a local keypoint-based geometric verification step estimates localization transformations. The authors demonstrated that using intensity information can reduce the computational timings. However, the complete localization pipeline operates at a frequency one order of magnitude lower than most LiDAR sensor frequencies.

While local keypoint features often lack descriptive power, global descriptors can struggle with variations in view-point. Therefore other works have also proposed to use 3D shapes or objects for the place recognition task. [104], for example, propose to perform place recognition by detecting planes in 3D environments. The planes are accumulated in a graph and an interpretation tree is used to match sub-graphs. A final geometric consistency test is conducted over the planes in the matched sub-graphs. The work is extended in [105] to use the covariance of the plane parameters instead of the number of points in planes for matching. This strategy is only applied to small, indoor environments and assumes a plane model which is no longer valid in unstructured environments. A somewhat analogous, seminal work on object-based loop-closure detection in indoor environments using RGB-D cameras is presented by [106]. Although presenting interesting ideas, their work can only handle a small number of well segmented objects in small scale environments. Similarily, [107] proposed a novel SLAM solution in which semantic information and local geometric features are jointly incorporated into a probabilistic framework. Such semantic-based approaches have significant potential, for example robustness to stark changes in point of view, but require the presence of human-known objects in the scene.

We therefore aim for an approach which does not rely on assumptions about the environment being composed of simplistic geometric primitives such as planes, or a rich library of objects. This allows for a more general, scalable solution.

Learning with 3D point clouds In recent years, Convolutional Neural Networks (CNNs) have become the state-of-the-art-method for generating learning-based descriptors, due to their ability to find complex patterns in data ([108]). For 3D point clouds, methods based on CNNs achieve impressive performance in applications such as object detection ([109–116]), semantic segmentation ([111, 112, 115, 117–119]), and 3D object generation ([120]), and LiDAR-based local motion estimation ([121, 122]).

Recently, a handful of works proposing the use of CNNs for localization in 3D point clouds have been published. First, [123] proposes extracting data-driven 3D keypoint descriptors (3DMatch) which are robust to changes in view-point. Although impressive retrieval performance is demonstrated using an RGB-D sensor in indoor environments, it is not clear whether this method is applicable in real-time in large-scale outdoor environments. A different approach based on 3D CNNs was proposed in [124] for performing localization in semi-dense maps generated with visual data. Recently,

[91] introduced a semi-handcrafted global descriptor for performing place recognition and rely on an ICP step for estimating the 6-DoF localization transformations. This method will be used as a baseline solution in Section 5.8 when evaluating the precision of our localization transformations. [125] propose describing local subsets of points using a deep neural network autoencoder. The authors state, however, that the implementation has not been optimized for real-time operation and no timings have been provided. In contrast, our work presents a data-driven segment-based localization method that can operate in real-time and that enables map reconstruction and semantic extraction capabilities.

To achieve this reconstruction capability, the architecture of our descriptor was inspired by autoencoders in which an encoder network compresses the input to a small dimensional representation, and a decoder network attempts to decompress the representation back into the original input. The compressed representation can be used as a descriptor for performing 3D object classification ([126]). [126] also present successful results using variational autoencoders for reconstructing voxelized 3D data. Different configurations of encoding and decoding networks have also been proposed for achieving localization and for reconstructing and completing 3D shapes and environments ([125, 127–131]).

While autoencoders present the interesting opportunity of simultaneously accomplishing both compression and feature extraction tasks, optimal performance at both is not guaranteed. As will be shown in Section 5.4, these two tasks can have conflicting goals when robustness to changes in point of view is desired. In this work, we combine the advantages of the encoding-decoding architecture of autoencoders with a technique proposed by [132]. The authors address the face recognition problem by first training a CNN to classify people in a training set and afterwards use the second to last layer as a descriptor for new faces. Other alternative training techniques include for example the use of contrastive loss ([133]) or triplet loss ([134]), the latter one being evaluated in Section 5.4. We use the resulting segment descriptors in the context of SLAM to achieve better performance, as well as significantly compressed maps that can easily be stored, shared, and reconstructed.

3 THE SEGMAP APPROACH

This section presents our *SegMap* approach to localization and mapping in 3D point clouds. It is composed of five core modules: segment extraction, description, localization, map reconstruction, and semantics extraction. These modules are detailed in this section and together allow single and multi-robot systems to create a powerful unified representation which can conveniently be transferred.

Segmentation The stream of point clouds generated by a 3D sensor is first accumulated in a dynamic voxel grid[2]. Point cloud segments are then extracted in a section of radius R around the robot. In this work we consider two types of incremental segmentation algorithms ([92]). The first one starts by removing points corresponding to the ground plane, which acts as a separator for clustering together the remaining points based on their Euclidean distances. The second algorithm computes local normals and curvatures for each point and uses these to extract flat or planar-like surfaces. Both methods are used to incrementally grow segments by using only newly active voxels as seeds which are either added to existing segments, form new segments or merge existing segments together[3]. This results in a handful of local segments, which are individually associated to a set of past observations *i.e.* $S_i = \{s_1, s_2, \ldots, s_n\}$. Each observation $s_j \in S_i$ is a 3D point cloud representing a snapshot of the segment as points are added to it. Note that s_n represents the latest observation of a segment and is considered *complete* when no further measurements are collected, *e.g.* when the robot has moved away.

Description Compact features are then extracted from these 3D segment point clouds using the data-driven descriptor presented in Section 4. A global segment map is created online by accumulating the segment centroids and corresponding descriptors. In order for the global map to most accurately represent the latest state of the world, we only keep the descriptor associated with the last and most complete observation.

Localization In the next step, candidate correspondences are identified between global and local segments using k nearest neighbours (k-NN) in feature space. The approximate k nearest descriptors are retrieved through an efficient query in a kd-tree. Localization is finally performed by verifying the largest subset of candidate correspondences for geometrical consistency on the basis of the segment centroids. Specifically, the centroids of the corresponding local and global segments must have the same geometric configuration up to a small jitter in their position, to compensate for slight variations in segmentation. In the experiments presented in Section 5.9, this

---

[2] In our experiments, we consider two techniques for estimating the local motion by registering successive LiDAR scans: one which uses ICP and one based on LOAM ([17]).

[3] For more information on these segmentation algorithms, the reader is encouraged to consult our prior work ([92]).

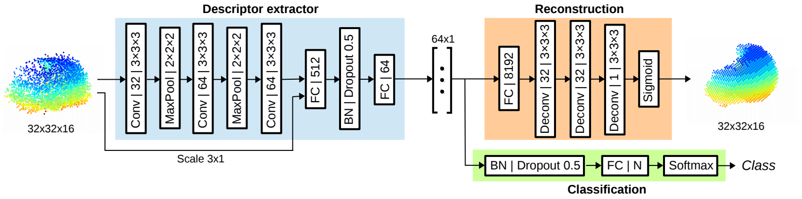


Figure 5.3: The descriptor extractor is composed of three convolutional and two fully connected layers. The 3D segments are compressed to a representation of dimension 64×1 which can be used for localization, map reconstruction and semantic extraction. Right of the descriptor we illustrate the classification and reconstruction layers which are used for training. In the diagram the convolutional (Conv), deconvolutional (Deconv), fully connected (FC) and batch normalization (BN) layers are abbreviated respectively. As parameters the Conv and Deconv layers have the number of filters and their sizes, FC layers have the number of nodes, max pool layers have the size of the pooling operation, and dropout layers have the ratio of values to drop. Unless otherwise specified, Rectified Linear Unit (ReLU) activation functions are used for all layers.

is achieved using an incremental recognition strategy which uses caching of correspondences for faster geometric verifications ([92]).

When a large enough geometrically consistent set of correspondence is identified, a 6-DoF transformation between the local and global maps is estimated. This transformation is fed to an incremental pose-graph SLAM solver which in turn estimates, in real-time, the trajectories of all robots ([135]).

Reconstruction Thanks to our autoencoder-like descriptor extractor architecture, the compressed representation can at any time be used to reconstruct an approximate map as illustrated in Figure 5.9. As the *SegMap* descriptor can conveniently be transmitted over wireless networks with limited bandwidth, any agent in the network can reconstruct and leverage this 3D information. More details on these reconstruction capabilities are given in Section 4.3.

Semantics The *SegMap* descriptor also contains semantically relevant information without the training process having enforced this property on the descriptor. This can, for example, be used to discern between static and dynamic objects in the environment to improve the robustness of the localization task. In this work we present an experiment where the network is able to distinguish between three different semantic classes: *vehicles*, *buildings*, and *others* (see Section 4.4).

4 THE SEGMAP DESCRIPTOR

In this section we present our main contribution: a data-driven descriptor for 3D segment point clouds which allows for localization, map reconstruction and semantic extraction. The descriptor extractor's architecture and the processing steps for inputting the point clouds to the network are introduced. We then describe our technique for training this descriptor to accomplish tasks of both segment retrieval and map reconstruction. We finally show how the descriptor can further be used to extract semantic information from the point cloud.

4.1 *Descriptor extractor architecture*

The architecture of the descriptor extractor is presented in Figure 5.3. Its input is a 3D binary voxel grid of fixed dimension $32 \times 32 \times 16$ which was determined empirically to offer a good balance between descriptiveness and the size of the network. The description part of the CNN is composed of three 3D convolutional layers with max pool layers placed in between and two fully connected layers. Unless otherwise specified, ReLU activation functions are used for all layers. The original scale of the input segment is passed as an additional parameter to the first fully connected layer to increase robustness to voxelization at different aspect ratios. The descriptor is obtained by taking the activations of the extractor's last fully connected layer. This architecture was selected by grid search over various configurations and parameters.

4.2 *Segment alignment and scaling*

A pre-processing stage is required in order to input the 3D segment point clouds for description. First, an alignment step is applied such that segments extracted from the same objects are similarly presented to the descriptor network. This is performed by applying a 2D Principal Components Analysis (PCA) of all points located within a segment. The segment is then rotated so that the x-axis of its frame of reference, from the robot's perspective, aligns with the eigenvector corresponding to the largest eigenvalue. We choose to solve the ambiguity in direction by rotating the segment so that the lower half section along the y-axis of its frame of reference contains the highest number of points. From the multiple alignment strategies we evaluated, the presented strategy worked best.

The network's input voxel grid is applied to the segment so that its center corresponds to the centroid of the aligned segment. By default the voxels have minimum side lengths of 0.1 m. These can individually be increased to exactly fit segments having one or more larger dimension than the grid. Whereas maintaining the aspect ratio while scaling can potentially offer better retrieval performance, this individual scaling with a minimum side length better avoids large errors caused by aliasing. We also found that this scaling method offers the best reconstruction performance, with only a minimal impact on the retrieval performance when the original scale of the segments is passed as a parameter to the network.

4.3 *Training the SegMap descriptor*

In order to achieve both a high retrieval performance and reconstruction capabilities, we propose a customized learning technique. The two desired objectives are imposed on the network by the *softmax cross entropy loss* L_c for retrieval and the reconstruction loss L_r. We propose to simultaneously apply both losses to the descriptor and to this end define a combined loss function L which merges the contributions of both objectives:

$$L = L_c + \alpha L_r \tag{5.1}$$

where the parameter α weighs the relative importance of the two losses. The value $\alpha = 200$ was empirically found to not significantly impact the performance of the combined network, as opposed to training separately with either of the losses. Weights are initialized based on Xavier's initialization method ([136]) and trained using the Adaptive Moment Estimation (ADAM) optimizer ([137]) with a learning rate of 10^{-4}. In comparison to Stochastic Gradient Descent (SGD), ADAM maintains separate learning rates for each network parameter, which facilitates training the network with two separate objectives simultaneously. Regularization is achieved using dropout ([138]) and batch normalization ([139]).

Classification loss L_c For training the descriptor to achieve better retrieval performance, we use a learning technique similar to the *N-ways classification problem* proposed by [132]. Specifically, we organize the training data into N classes where each class contains all observations of a segment or of multiple segments that belong to the same object or environment part. Note that these classes are solely used for training the descriptor and are not related to the semantics presented in Section 4.4. As seen in Fig 5.3, we then append a classification layer to the descriptor and teach the network to associate a score to each of the N predictors for each segment sample. These scores are compared to the true class labels using *softmax cross entropy loss*:

$$L_c = -\sum_{i=1}^{N} y_i \log \frac{e^{l_i}}{\sum_{k=1}^{N} e^{l_k}} \tag{5.2}$$

where y is the one hot encoded vector of the true class labels and l is the layer output.

Given a large number of classes and a small descriptor dimensionality, the network is forced to learn descriptors that better generalize and prevent overfitting to specific segment samples. Note that when deploying the system in a new environment the classification layer is removed, as its output is no longer relevant. The activations of the previous fully connected layer are then used as a descriptor for segment retrieval through k-NN.

Reconstruction loss L_r As depicted in Figure 5.3, map reconstruction is achieved by appending a decoder network and training it simultaneously with the descriptor extractor and classification layer. This decoder is composed of one fully connected and three deconvolutional layers with a final sigmoid output. Note that no weights are shared between the descriptor and the decoder networks. Furthermore, only the descriptor extraction needs to be run in real-time on the robotic platforms, whereas the decoding part can be executed any time a reconstruction is desired.

As proposed by [126], we use a specialized form of the *binary cross entropy loss*, which we denote by L_r:

$$\begin{split} L_r = -\sum_{x,y,z} (\gamma\, t_{xyz} \log(o_{xyz}) \\ + (1-\gamma)(1-t_{xyz}) \log(1-o_{xyz})) \end{split} \tag{5.3}$$

where t and o respectively represent the target segment and the network's output and γ is a hyperparameter which weighs the relative importance of false positives and false negatives. This parameter addresses the fact that only a minority of voxels are activated in the voxel grid. In our experiments, the voxel grids used for training were on average only 3% occupied and we found $\gamma = 0.9$ to yield good results.

4.4 *Knowledge transfer for semantic extraction*

As can be observed from Figure 5.1, segments extracted by the *SegMap* approach for localization and map reconstruction often represent objects or

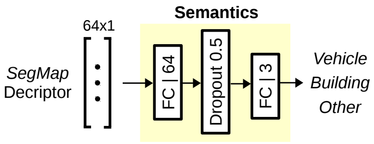


Figure 5.4: A simple fully connected network that can be appended to the *SegMap* descriptor (depicted in Figure 5.3) in order to extract semantic information. In our experiments, we train this network to distinguish between vehicles, buildings, and other objects.

parts of objects. It is therefore possible to assign semantic labels to these segments and use this information to improve the performance of the localization process. As depicted in Figure 5.4, we transfer the knowledge embedded in our compact descriptor by training a semantic extraction network on top of it. This last network is trained with labeled data using the *softmax cross entropy loss* and by freezing the weights of the descriptor network.

In this work, we choose to train this network to distinguish between three different semantic classes: *vehicles*, *buildings*, and *others*. Section 5.9 shows that this information can be used to increase the robustness of the localization algorithm to changes in the environment and to yield smaller map sizes. This is achieved by rejecting segments associated with potentially dynamic objects, such as vehicles, from the list of segment candidates.

4.5 *SegMini*

Finally we propose a lightweight version of the *SegMap* descriptor which is specifically tailored for resource-limited platforms. *SegMini* has the same architecture as *SegMap* (see Figure 5.3), with the exception that the number of filter in the convolutional layers and the size of the dense layers is halved. Without compromising much on the descriptor retrieval performance this model leads to a computational speedup of 2x for GPU and 6x for CPU (Section 5.3).

5 EXPERIMENTS

This section presents the experimental validation of our approach. We first present a procedure for generating training data and detail the performance of the *SegMap* descriptor for localization, reconstruction and semantics extrac-

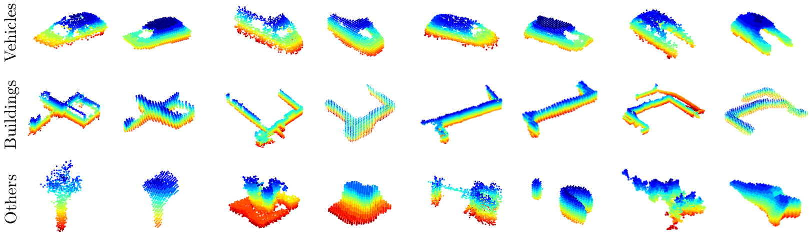


Figure 5.5: An illustration of the *SegMap* reconstruction capabilities. The segments are extracted from sequence 00 of the KITTI dataset and represent, from top to bottom respectively, vehicles, buildings, and other objects. For each segment pair, the reconstruction is shown to the right of the original. The network manages to accurately reconstruct the segments despite the high compression to only 64 values. Note that the voxelization effect is more visible on buildings as larger segments necessitate larger voxels to keep the input dimension fixed.

tion. We finally evaluate the complete *SegMap* solution in multiple real-world experiments.

5.1 *Experiment setup and implementation*

All experiments were performed on a system equipped with an Intel i7-6700K processor, and an NVIDIA GeForce GTX 980 Ti GPU. The CNN models were developed and executed in real-time using the TensorFlow library. The `libnabo` library is used for descriptor retrieval with fast k-NN search in low dimensional space ([140]). The incremental optimization back-end is based on the iSAM2 implementation from [28].

5.2 *Training data*

The *SegMap* descriptor is trained using real-world data from the KITTI odometry dataset ([141]). Sequences 05 and 06 are used for generating training and testing data, whereas sequence 00 is solely used for validation of the descriptor performance. In addition, end-to-end experiments are done using sequences 00 and 08, as they feature long tracks with multiple overlapping areas in the trajectories. For each sequence, segments are extracted using an incremental Euclidean distance-based region growing technique ([92]). This algorithm extracts point clouds representing parts of objects or buildings which are separated after removing the ground plane (see Figure 5.5). The training data is filtered by removing segments with too few observations, or training classes (as described in Section 4.3) with too few samples. In this manner, 3300, 1750, 810 and 2400 segments are respectively generated from sequences 00, 05, 06 and 08, with an average of 12 observations per segment over the whole dataset.

Data augmentation

To further increase robustness by reducing sensitivity to rotation and view-point changes in the descriptor extraction process, the dataset is augmented through various transformations at the beginning of each training epoch. Each segment is rotated at different angles to the alignment described in Section 4.2 to simulate different view-points. In order to simulate the effect of occlusion for each segment we remove all points which fall on one side of a randomly generated slicing plane that does not remove more than 50% of the points. Finally, random noise is simulated by randomly removing up to 10% of the points in the segment. Note that these two data augmentation steps are performed prior to voxelization.

Ground-truth generation

In the following step, we use GPS readings in order to identify ground truth correspondences between segments extracted in areas where the vehicle performed multiple visits. Only segment pairs with a maximum distance between their centroids of 3.0 m are considered. We compute the 3D convex hull of each segment observation s_1 and s_2 and create a correspondence when the following condition, inspired from the Jaccard index, holds:

$$\frac{\text{Volume}(\text{Conv}(s_1) \cap \text{Conv}(s_2))}{\text{Volume}(\text{Conv}(s_1) \cup \text{Conv}(s_2))} \geqslant p \tag{5.4}$$

In our experiments we found $p = 0.3$ to generate a sufficient number of correspondences while preventing false labelling. The procedure is performed on sequences 00, 05, and 06, generating 150, 260, and 320 ground truth correspondences respectively. We use two-thirds of the correspondences for augmenting the training data and one-third for creating validation samples. Finally, the ground-truth correspondences extracted from sequence 00 are used in Section 5.4 for evaluating the retrieval performance.

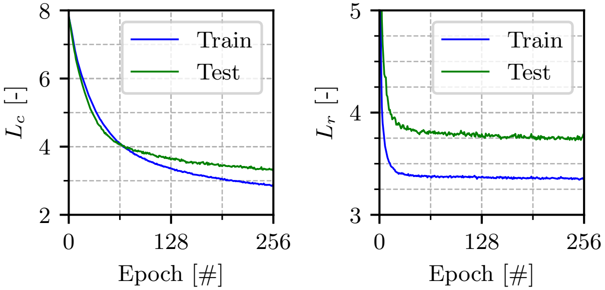


Figure 5.6: The classification loss L_c (left) and the reconstruction loss L_r (right) components of the total loss L, when training the descriptor extractor along with the reconstruction and classification networks. The depicted reconstruction loss has already been scaled by α.

5.3 *Training the models*

The descriptor extractor and the decoding part of the reconstruction network are trained using all segments extracted from drive 05 and 06. Training lasts three to four hours on the GPU and produces the classification and scaled reconstruction losses depicted in Figure 5.6. The total loss of the model is the sum of the two losses as describe in Section 4.3. We note that for classification the validation loss follows the training loss before converging towards a corresponding accuracy of 41% and 43% respectively. In other words, 41% of the validation samples were correctly assigned to one of the $N = 2500$ classes. This accuracy is expected given the large quantity of classes and the challenging task of discerning between multiple training samples with similar semantic meaning, but few distinctive features, *e.g.* flat walls. Note that we achieve a very similar classification loss L_c, when training with and without the L_r component of the combines loss L. On a GPU the *SegMap* descriptor takes on average 0.8 ms to compute, while the *SegMini* descriptor takes 0.3 ms. On the CPU the performance gain is more significant, as it takes 245 ms for a *SegMap* descriptor as opposed to only 41 ms for *SegMini*, which is a 6x improvement in efficiency.

5.4 *Descriptor retrieval performance*

We evaluate the retrieval performance of the *SegMap* descriptor against state-of-the-art methods as well as other networks trained with different secondary goals. First, our descriptor is compared with eigenvalue-based point cloud features ([142]). We also evaluate the effect of training only for the classification task (Classification) or of training only for the reconstruction one (Autoencoder). Additionally, we compare classification-based learning with a triplet loss solution ([143]), where during training, we enforce segments from the same sequence to have a minimal Euclidean distance. We use a per batch hard mining strategy and the best performing variant of triplet loss as proposed by [144]. We finally evaluate the *SegMini* model introduced in Section 4.5.

The retrieval performance of the aforementioned descriptors is depicted in Fig 5.7. The Receiver Operating Characteristic (ROC) curves are obtained by generating 45M labeled pairs of segment descriptors from sequence 00 of the KITTI odometry dataset ([141]). Using ground-truth correspondences, a positive sample is created for each possible segment observation pair. For each positive sample a thousand negative samples are generated by randomly sampling segment pairs whose centroids are further than 20 m apart. The positive to negative sample ratio is representative of our localization problem given that a map created from KITTI sequence 00 contains around a thousand segments. The ROC curves are finally obtained by varying the threshold applied on the L^2 norm between the two segment descriptors. We note that training with triplet loss offers the best ROC performance on these datasets, as it imposes the most consistent separation margin across all segments.

The ROC is not the best evaluation metric for this retrieval task, because it evaluates the quality of classification for a single threshold across all segments. As introduced in Section 3, correspondences are made between segments from the local and global maps by using k-NN retrieval in feature space. The varying parameter is the number of neighbours that is retrieved and not a threshold on the feature distances, which only matter in a relative fashion on a per query basis. In order to avoid false localizations, the aim is to reduce the number k of neighbours that need to be considered. Therefore, as a segment grows with time, it is critical that its descriptor converges as quickly as possible towards the descriptor of the corresponding segment in the target map, which in our case is extracted from the last and most *complete* observation (see Section 3). This behaviour is evaluated in Figure 5.8a which relates the number of neighbours which need to be considered to find the correct association, as a function of segment completeness. We note that the

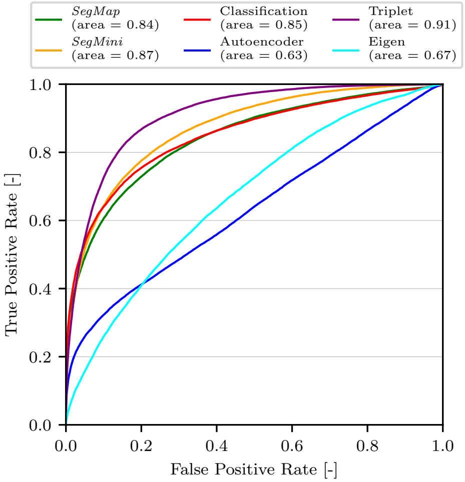
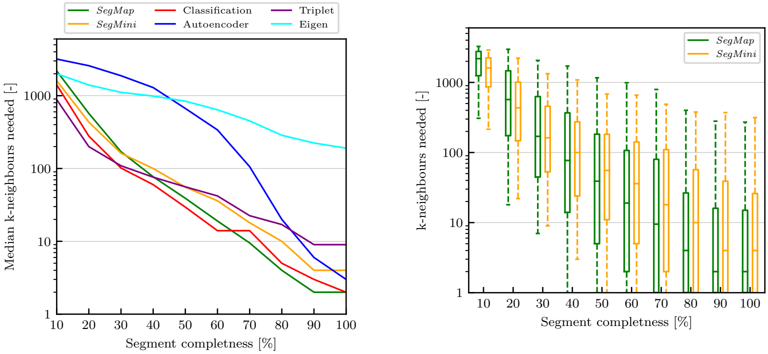


Figure 5.7: ROC curves for the descriptors considered in this work. This evaluation is performed using ground-truth correspondences extracted from sequence 00 of the KITTI odometry dataset ([141]). Note that the ROC is not an optimal measure of the quality of the retrieval performance, since it only considers a single threshold for all segment pairs and does not look at the relative ordering of matches on a per query basis.

SegMap descriptor offers competitive retrieval performance at every stage of the growing process. In practice this is important since it allows closing challenging loops such as the one presented in Figure 5.1.

Interestingly, the autoencoder has the worst performance at the early growing stages whereas good performance is observed at later stages. This is in accordance with the capacity of autoencoders to precisely describe the geometry of a segment, without explicitly aiming at gaining a robust representation in the presence of occlusions or changes in view-point. Although the triplet loss training method offers the best ROC performance, Figure 5.8a suggests that training with the secondry goal of classification yields considerably better results at the later stages of growing. The poor performance of the triplet loss method especially for very similar segments could be caused by the hard mining amplifying the noise in the dataset. After a certain point the ordering of matches becomes irrelevant, because the goal is to minimize the number of retrieved neighbours and retrieving too many is computationally unfeasible for later stages of the process. Therefore although the purely



(a) Median k-nearest neighbours needed for all methods as a function of segment completeness.

(b) More detailed plot of the k-nearest neighbours needed for the proposed methods as a function of segment completeness.

Figure 5.8: This figure presents how *quickly* descriptors extracted from incrementally grown segments contain relevant information that can be used for localization. The x-axis represents the completeness of a segment until all its measurements have been accumulated (here termed *complete*, see Section 3). In (a) the log-scaled y-axis represents the median of how many neighbours in the target map need to be considered in order to retrieve the correct target segment (the lower the better). Similarly (b) presents the same results in more detail for the proposed models. The *SegMap* descriptor offers over the majority of the growing process one order of magnitude better retrieval performance than the hand-crafted baseline descriptor.

classification-based model performs slightly better for very early observations of a segment, this gain in performance does not matter. The proposed *SegMap* descriptor achieves the best performance for very complete segments, where matches are most likely to happen, and maintains a comparable performance across very partial observations. A more detailed plot for the retrieval performance of the *SegMap* and *SegMini* is presented in Figure 5.8b, where also the variance in the retrieval accuracy is shown.

5.5 *Reconstruction performance*

In addition to offering high retrieval performance, the *SegMap* descriptor allows us to reconstruct 3D maps using the decoding CNN described in Section 4.3. Some examples of the resulting reconstructions are illustrated in Figure 5.5, for various objects captured during sequence 00 of the KITTI odometry dataset. Experiments done at a larger scale are presented in

	Descriptor size			
	16	32	64	128
Autoencoder	0.87	0.91	0.93	0.94
SegMap	0.86	0.89	0.91	0.92

Table 5.1: Average ratio of corresponding points within one voxel distance between original and reconstructed segments. Statistics for *SegMap* and the autoencoder baseline using different descriptor sizes.

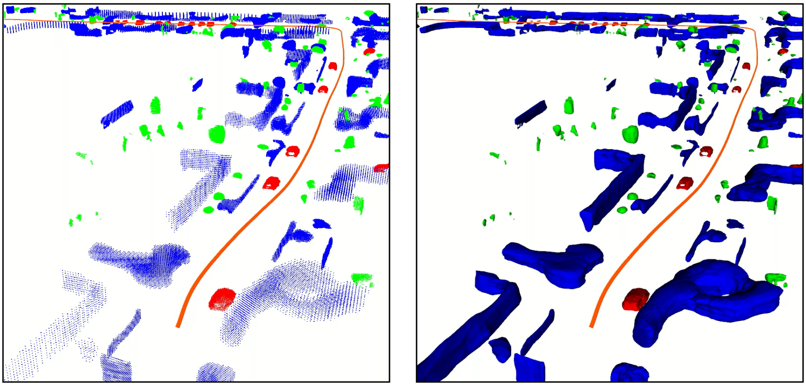


Figure 5.9: Visualization of segment reconstructions, as point clouds (left), and as surface meshes (right), generated from sequence 00 of the KITTI dataset. The quantization of point cloud reconstructions is most notable in the large wall segments (blue) visible in the background. Equivalent surface mesh representations do not suffer from this issue.

Figure 5.14, where buildings of a powerplant and a foundry are reconstructed by fusing data from multiple sensors.

Since most segments only sparsely model real-world surfaces, they occupy on average only 3% of the voxel grid. To obtain a visually relevant comparison metric, we calculate for both the original segment and its reconstruction the ratio of points having a corresponding point in the other segment, within a distance of one voxel. The tolerance of one voxel means that the shape of the original segment must be preserved while not focusing on reconstructing each individual point. Results calculated for different descriptor sizes are presented in Table 5.1, in comparison with the purely reconstruction focused baseline. The *SegMap* descriptor with a size of 64 has on average 91% correspondences between the points in the original and reconstructed segments, and is only slightly outperformed by the autoencoder baseline. Contrastingly, the significantly higher retrieval performance of the *SegMap* descriptor makes it a clear all-rounder choice for achieving both localization and map reconstruction.

Overall, the reconstructions are well recognizable despite the high compression ratio. In Figure 5.9, we note that the quantization error resulting from the voxelization step mostly affects larger segments that have been downscaled to fit into the voxel grid. To mitigate this problem, one can adopt a natural approach to representing this information in 3D space, which is to calculate the isosurface for a given probability threshold. This can be computed using the "marching cubes" algorithm, as presented by [145]. The result is a triangle-mesh surface, which can be used for intuitive visualization, as illustrated in Figure 5.9 and Figure 5.10.

5.6 *Semantic extraction performance*

For training the semantic extractor network (Figure 5.4), we manually labeled the last observation of all 1750 segments extracted from KITTI sequence 05. The labels are then propagated to each observation of a segment for a total of 20k labeled segment observations. We use 70% of the samples for training the network and 30% for validation. Given the low complexity of the semantic extraction network and the small amount of labeled samples, training takes only a few minutes. We achieve an accuracy of 89% and 85% on the training and validation data respectively. Note that our goal is not to improve over other semantic extraction methods ([112, 115]), but rather to illustrate that our compressed representation can additionally be used for discarding dynamic elements of the environment and for reducing the map size (Section 5.9).

5.7 *6-DoF pose retrieval performance*

In this section, we demonstrate how the advantageous properties of *SegMap*, particularly the descriptor retrieval performance, translate to state-of-the-art global localization results. We therefore compare our approach to a global localization method, LocNet ([91]). It uses rotation-invariant, data-driven descriptors that yield reliable matching of 3D LiDAR scans. LocNet retrieves a nearest neighbor database scan and returns its pose, its output is thus limited to the poses already present in the target map. Therefore, it works reliably in environments with well defined trajectories (e.g. roads), but fails to return a precise location within large traversable areas such as squares

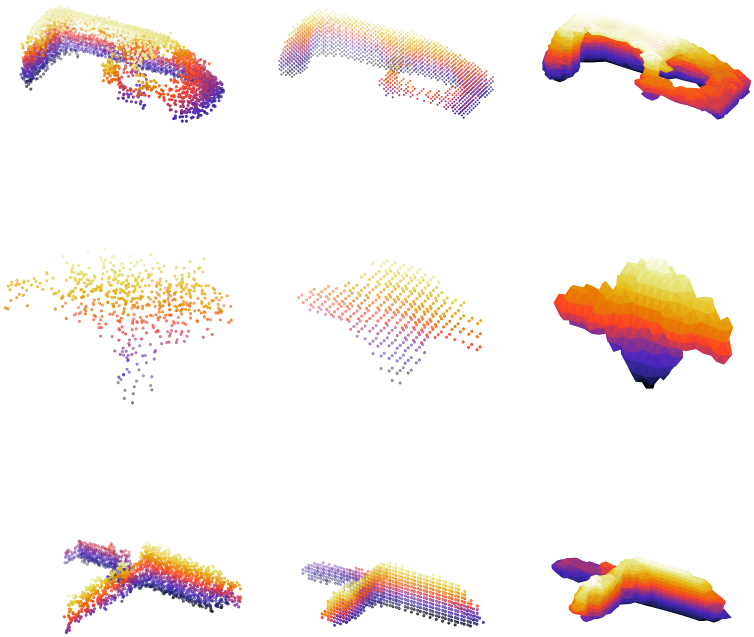


Figure 5.10: A visual comparison between (left) the original point cloud, (middle) the reconstruction point cloud, and (right) the reconstruction mesh, for 3 segments.

or hallways. In contrast, *SegMap* uses segment correspondences to estimate an accurate 6-DoF pose that includes orientation, which cannot be retrieved directly using the rotation-invariant LocNet descriptors.

Figure 5.11 presents the evaluation of both methods on the KITTI 00 odometry sequence (4541 scans). We use the first 3000 LiDAR scans and their ground-truth poses to create a map, against which we then localize using the last 1350 scans. *SegMap* demonstrates a superior performance both by successfully localizing about 6% more scans and by returning more accurate localized poses. To note is that from the query positions only 65% of them were taken within a distance of 50 m of the target map, therefore limiting the maximum possible saturation. We believe that robust matching of segments,

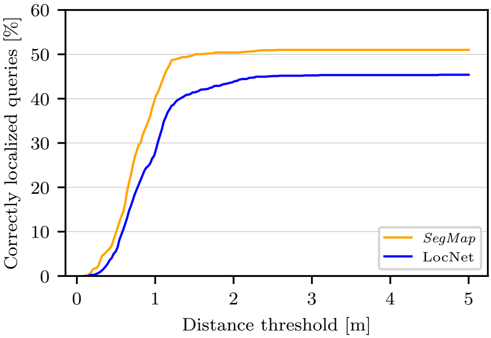
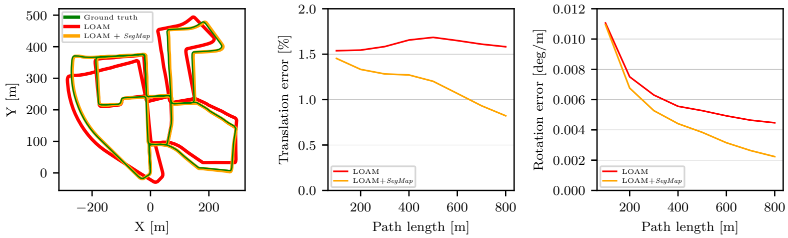


Figure 5.11: Cumulative distribution of position errors on KITTI 00 odometry sequence that compares *SegMap* with state-of-the-art data-driven LocNet approach presented in [91]. Our proposed method retrieves a full 6-DoF pose while LocNet uses global scan descriptors to obtain the nearest pose of the target map. *SegMap* retrieves poses for a larger number of scans and the returned estimates are more accurate. The results saturate at about 52% as not all query positions overlap with the target map, with only 65% of them being within a radius of 50 m from the map.

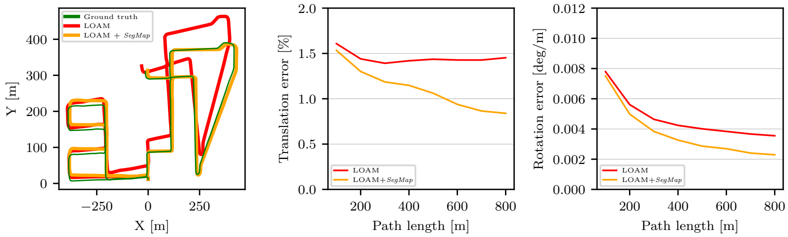
a principle of our method, helps to establish reliable correspondences with the target map, particularly for queries further away from the mapped areas. This state-of-the-art localization performance is further complemented by a compact map representation, with reconstruction and semantic labeling capabilities.

5.8 *A complete mapping and localization system*

So far, we have only evaluated *SegMap* as a stand-alone global localization system, demonstrating the performance of segment descriptors and the 6-DoF pose retrieval. Such global localization systems, however, are commonly used in conjunction with odometry and mapping algorithms. To prove the qualities of *SegMap* in such a scenario, we have combined it with a state-of-the-art LiDAR odometry and mapping system, LOAM ([17]). Our implementation is based on a publicly available version of LOAM and achieves similar odometry performance results on KITTI, as the ones reported by other works, such as [122]. We use a loosely coupled approach, where LOAM is used to undistort the scans and provide an odometry estimate between frames, in real-time. The scans from LOAM are used to build a local map from which segments



(a) KITTI odometry sequence 00.



(b) KITTI odometry sequence 08.

Figure 5.12: The trajectories for KITTI odometry sequences a) 00 and b) 08 for LOAM and the combination of LOAM and *SegMap*. In addition we show translation and rotation errors for the two approaches, using the standard KITTI evaluation method [141].

are extracted and attached to a pose-graph, together with the odometry measurements. Loop closures can then be added in real-time as constraints in the graph, to correct the drifting odometry. This results in a real-time LiDAR-only end-to-end pipeline that produces segment-based maps of the environment, with loop-closures.

In all experiments, we use a local map with a radius of 50 m around the robot. When performing segment retrieval we consider 64 neighbours and require a minimum of 7 correspondences, which are altogether geometrically consistent, to output a localization. These parameters were chosen empirically using the information presented in Figure 5.7 and 5.8 as a reference.

Our evaluations on KITTI sequences 00 and 08 (Figure 5.12) demonstrate that global localization results from *SegMap* help correct for the drift of the odometry estimates. The trajectories outputted by the system combining *SegMap* and LOAM, follow more precisely the ground-truth poses provided by the benchmark, compared to the open-loop solution. We also show how global localizations reduce both translational and rotational errors.

Particularly over longer paths *SegMap* is able to reduce the drift in the trajectory estimate by up to 2 times, considering both translation and rotation errors. For shorter paths, the drift only improves marginally or remains the same, as local errors are more dependent on the quality of the odometry estimate. We believe that our evaluation showcases not only the performance of *SegMap*, but also the general benefits stemming from global localization algorithms.

5.9 *Multi-robot experiments*

We evaluate the *SegMap* approach on three large-scale multi-robot experiments: one in an urban-driving environment and two in search and rescue scenarios. In both indoor and outdoor scenarios we use the same model which was trained on the KITTI sequences 05 and 06 as described in Section 5.3.

The experiments are run on a single machine, with a multi-thread approach to simulating a centralized system. One thread per robot accumulates the 3D measurements, extracting segments, and performing the descriptor extraction. The descriptors are transmitted to a separate thread which localizes the robots through descriptor retrieval and geometric verification, and runs the pose-graph optimization. In all experiments, sufficient global associations need to be made, in real-time, for co-registration of the trajectories and merging of the maps. Moreover in a centralized setup it might be crucial to limit the transmitted data over a wireless network with potentially limited bandwidth.

Multi-robot SLAM in urban scenario

In order to simulate a multi-robot setup, we split sequence 00 of the KITTI odometry dataset into five sequences, which are simultaneously played back on a single computer for a duration of 114 seconds. In this experiment, the semantic information extracted from the *SegMap* descriptors is used to reject segments classified as *vehicles* from the retrieval process.

With this setup, 113 global associations were discovered, allowing to link all the robot trajectories and create a common representation. We note that performing ICP between the associated point clouds would refine the localization transformation by, on average, only $0.13(6)$ m which is in the order of our voxelization resolution. However, this would require the original point cloud data to be kept in memory and transmitted to the central computer. Future work could consider refining the transformations by performing ICP on the reconstructions.

Table 5.2: Statistics resulting from the three experiments.

Statistic	**KITTI**	**Powerplant**	**Foundry**
Duration (s)	114	850	1086
Number of robots	5	3	2
Number of segmented local cloud	557	758	672
Average number of segments per cloud	42.9	37.0	45.4
Bandwidth for transmitting local clouds (kB/s)	4814.7	1269.2	738.1
Bandwidth for transmitting segments (kB/s)	2626.6	219.4	172.2
Bandwidth for transmitting descriptors (kB/s)	60.4	9.5	8.1
Final map size with the *SegMap* descriptor (kB)	386.2	181.3	121.2
Number of successful localizations	113	27	85

Localization and map reconstruction was performed at an average frequency of 10.5 Hz and segment description was responsible for 30% of the total runtime with an average duration of 28.4 ms per local cloud. A section of the target map which has been reconstructed from the descriptors is depicted in Figure 5.1.

Table 5.2 presents the results of this experiment. The required bandwidth is estimated by considering that each point is defined by three 32-bit floats and that 288 additional bits are required to link each descriptor to the trajectories. We only consider the *useful data* and ignore any transfer overhead. The final map of the KITTI sequence 00 contains 1341 segments out of which 284 were classified as vehicles. A map composed of all the raw segment point clouds would be 16.8 MB whereas using our descriptor it is reduced to only 386.2 kB. This compression ratio of 43.5x can be increased to 55.2x if one decides to remove vehicles from the map. This shows that our approach can be used for mapping much larger environments.

Multi-robot SLAM in disaster environments

For the two following experiments, we use data collected by Unmanned Ground Vehicles (UGVs) equipped with multiple motor encoders, an Xsens MTI-G inertial measurement unit (IMU) and a rotating 2D SICK LMS-151 LiDAR. First, three UGVs were deployed at the decommissioned Gustav Knepper powerplant: a large two-floors utility building measuring 100 m long by 25 m wide. The second mission took place at the Phoenix-West foundry in a semi-open building made of steel. A section measuring 100 m by 40 m was mapped using two UGVs. The buildings are shown in Fig 5.13.

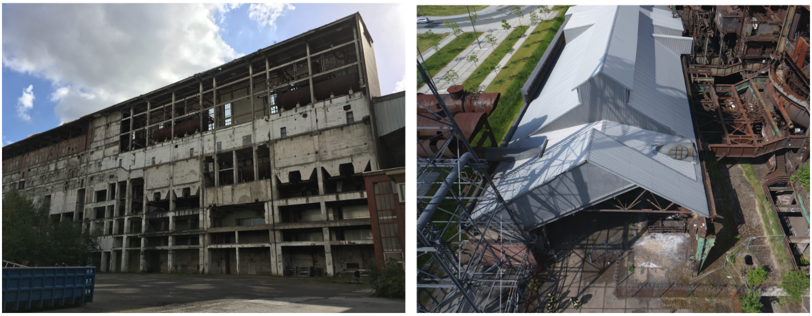


Figure 5.13: Buildings of the Gustav Knepper powerplant (left) and the Phoenix-West foundry (right).

For these two experiments, we used an incremental smoothness-based region growing algorithm which extracts plane-like segments ([92]). The resulting *SegMap* reconstructions are shown in Figure 5.14 and detailed statistics are presented in Table 5.2. Although these planar segments have a very different nature than the ones used for training the descriptor extractor, multiple localizations have been made in real-time so that consistent maps could be reconstructed in both experiments. Note that these search and rescue experiments were performed with sensors without full 360° field of view. Nevertheless, *SegMap* allowed robots to localize in areas visited in opposite directions.

6 DISCUSSION AND FUTURE WORK

While our proposed method works well in the demonstrated experiments it is limited by the ability to only observe the geometry of the surrounding structure. This can be problematic in some man-made environments, which are repetitive and can lead to perceptual aliasing, influencing both the descriptor and the geometric consistency verification. This could be addressed by detecting such aliasing instances and dealing with them explicitly, through for example an increase in the constraints of the geometric verification. On the other hand, featureless environments, such as for example flat fields or straight corridors, are equally challenging. As even LIDAR-based odometry methods struggle to maintain an accurate estimate of pose, these environments do not allow for reliable segment extraction. In these cases the map will drift until a more distinct section is reached that can be loop-closed, thus

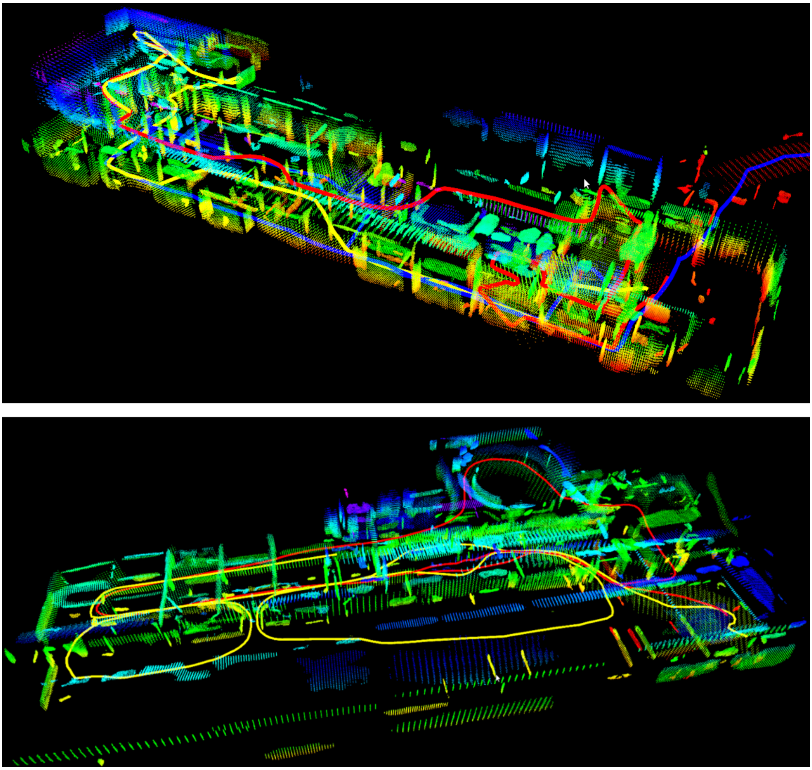


Figure 5.14: This figure illustrates a reconstruction of the buildings of the Gustav Knepper powerplant (top) and the Phoenix-West foundry (bottom). The point clouds are colored by height and the estimated robot trajectories are depicted with colored lines.

allowing for partial correction of the previously built pose-graph. In different environments the two segmentation algorithms will have varying performances, with the Euclidean distance based one working better in outdoor scenarios, while the curvature-based one is more suited for indoor scenarios. A future approach would be to run the two segmentation strategies in parallel, thus allowing them to compensate for each others short-comings and enabling robots to navigate in multiple types of environments during the same mission.

In order to address some of the aforementioned drawbacks, in future work we would like to extend the *SegMap* approach to different sensor modalities and different point cloud segmentation algorithms. For example, integrating information from camera images, such as color, into the descriptor learning could mitigate the lack of descriptiveness of features extracted from segments with little distinct geometric structure. In addition, color and semantic information from camera images could not only be used to improve the descriptor but also to enhance the robustness of the underlying segmentation process. Considering the real-time constraints of the system, to note with respect to future work are the additional computational expenses introduced by processing and combining more data modalities.

Furthermore, whereas the present work performs segment description in a discrete manner, it would be interesting to investigate incremental updates of learning-based descriptors that could make the description process more efficient, such as the voting scheme proposed by [109]. Instead of using a feed-forward network, one could also consider a structure that leverages temporal information in the form of recurrence in order to better describe segments based on their evolution in time. Moreover, it could be of interest to learn the usefulness of segments as a precursory step to localization, based on their distinctiveness and semantic attributes.

7 CONCLUSION

This paper presented *SegMap*: a segment-based approach for *map representation* in localization and mapping with 3D sensors. In essence, the robots' surroundings are decomposed into a set of segments, and each segment is represented by a distinctive, low dimensional learning-based descriptor. Data associations are identified by segment descriptor retrieval and matching, made possible by the repeatable and descriptive nature of segment-based features.

We have shown that the descriptive power of *SegMap* outperforms hand-crafted features as well as the evaluated data-driven baseline solutions. Our experiments indicate that *SegMap* offers competitive localization performance, in comparison to the state-of-the-art LocNet method. Additionally, we have combined our localization approach with LOAM, a LiDAR-based local motion estimator, and have demonstrated that the output of *SegMap* helps correct the drift of the open-loop odometry estimate. Finally, we have introduced *SegMini*: a light-weight version of our *SegMap* descriptor which can more easily be deployed on platforms with limited computational power.

In addition to enabling global localization, the *SegMap* descriptor allows us to reconstruct a map of the environment and to extract semantic information. The ability to reconstruct the environment while achieving a high compression rate is one of the main features of *SegMap*. This allows us to perform both SLAM and 3D reconstruction with LiDARs at large scale and with low communication bandwidth between the robots and a central computer. These capabilities have been demonstrated through multiple experiments with real-world data in urban driving and search and rescue scenarios. The reconstructed maps could allow performing navigation tasks such as, for instance, multi-robot global path planning or increasing situational awareness.

ACKNOWLEDGMENTS

This work was supported by the European Union's Seventh Framework Program for research, technological development and demonstration under the TRADR project No. FP7-ICT-609763, from the EU H2020 research project under grant agreement No 688652, the Swiss State Secretariat for Education, Research and Innovation (SERI) No 15.0284, and by the Swiss National Science Foundation through the National Center of Competence in Research Robotics (NCCR). The authors would like to thank Abel Gawel, Mark Pfeiffer, Mattia Gollub, Helen Oleynikova, Philipp Krüsi, Igor Gilitschenski and Elena Stumm for their valuable collaboration and support.

PAPER III

SEMSEGMAP – 3D SEGMENT-BASED SEMANTIC LOCALIZATION

Andrei Cramariuc*, Florian Tschopp*, Nikhilesh Alatur, Stefan Benz, Tillmann Falck, Marius Brühlmeier, Benjamin Hahn, Juan Nieto, and Roland Siegwart

*contributed equally

ABSTRACT

Localization is an essential task for mobile autonomous robotic systems that want to use pre-existing maps or create new ones in the context of SLAM. Today, many robotic platforms are equipped with high-accuracy 3D LiDAR sensors, which allow a geometric mapping, and cameras able to provide semantic cues of the environment. Segment-based mapping and localization have been applied with great success to 3D point-cloud data, while semantic understanding has been shown to improve localization performance in vision based systems. In this paper we combine both modalities in *SemSegMap*, extending *SegMap* into a segment based mapping framework able to also leverage color and semantic data from the environment to improve localization accuracy and robustness. In particular, we present new segmentation and descriptor extraction processes. The segmentation process benefits from additional distance information from color and semantic class consistency resulting in more repeatable segments and more overlap after re-visiting a place. For the descriptor, a tight fusion approach in a deep-learned descriptor extraction network is performed leading to a higher descriptiveness for landmark matching. We demonstrate the advantages of this fusion on multiple simulated and real-world datasets and compare its performance to various baselines. We show that we are able to find 50.9% more high-accuracy prior-less global localizations compared to *SegMap* on challenging datasets using very compact maps while also providing accurate full 6 DoF pose estimates in real-time.

Published in:
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021
DOI: 10.1109/IROS51168.2021.9636156

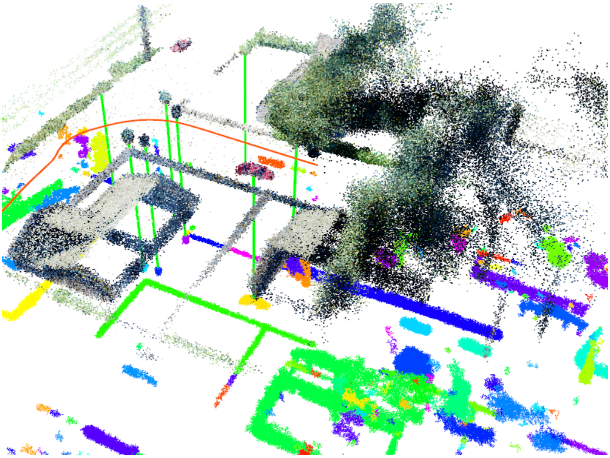


Figure 6.1: This image shows the *SemSegMap* pipeline in action. *SemSegMap* is able to perform segment-based semantic localization on point cloud data enriched with semantic and color information from a visual camera. The currently observed local map around the robot is shown as the colored point cloud on top of the global map depicted below, with each segment having a unique color. Green lines show matched segment correspondences leading to a localization while the orange line shows the robot trajectory.

1 INTRODUCTION

Mobile robots are continuously increasing their impact on our everyday life and becoming more and more viable not only in structured factories but also unstructured environments and in contact with humans [58]. One of the most crucial capabilities of mobile robots is the ability to know their position in the environment in order to navigate and fulfill their task. Positioning can be framed either in the context of localization in a known map or in the context of the Simultaneous Localization And Mapping (SLAM) problem where localizations and potential loop closures are needed to maintain a consistent map. A multitude of solutions to the positioning problem exist, mainly depending on the available sensor data, specific challenges of the environment and computational limitations of the robotic platform [58].

For standard indoor applications, visual localization based on hand-crafted keypoint descriptors [15, 146] has been demonstrated to achieve high accuracy and recall. However, outdoor applications typically pose challenges to those methods, namely large-scale environments, self similarity and vast appearance changes due to weather, daytime and seasonal conditions [147]. Vision based learning methods improve on viewpoint and appearance invariance [52, 82, 148] by enabling a more context aware description. In contrast, LiDAR based localization achieves illumination invariance using geometry to describe the environment [6, 149], however in turn missing rich information available from vision.

As a combined solution, in this paper, we introduce *SemSegMap*, a method that leverages the visual and semantic information available from cameras and fuses it with geometric information from a standard 3D LiDAR. As a basis for our localization framework we use *SegMap* [6], a LiDAR based SLAM pipeline that uses 3D segments of the environment as landmarks and allows for 6D pose retrieval from compact descriptors in large-scale maps. In contrast to *SegMap*, in *SemSegMap*, as outlined in Figure 6.1, first the point cloud (PC) gets enriched with color and semantic information using back-projection of semantically segmented RGB images. Further, the PC is segmented based on geometric, color and semantic information to create consistent and meaningful segments. We show in multiple experiments that as a result of this fusion, the segmentation process and the generated descriptors become more robust to viewpoint and appearance changes, thus enabling a more consistent re-localization of the robot.

Our contributions are as follow

- We show that integrating color and semantic information from a cameras into PCs improves both the segmentation and descriptor generation processes, leading to more consistent 6D localizations in a SLAM pipeline.

- We introduce a simulation based learning pipeline for training segment descriptors using ground truth associations, and show their transferability to real-world scenarios.

- We demonstrate the performance of *SemSegMap* in an extensive evaluation on simulation and real-world data outperforming various baselines.

- For the benefit of the community, we open-source the whole framework under a permissive license available at: `https://github.com/ethz-asl/segmap`.

2 RELATED WORK

The ability to localize is at the heart of the SLAM problem [58]. Two distinct problems addressed in localization are the localization of landmarks, which can then be used to calculate a precise 6D location in a global map, and place recognition, where only a rough neighborhood is estimated.

Vision-based place recognition methods such as NetVLAD [82] or DELF [150] have the advantage that they can incorporate a lot of contextual information and thus gain high robustness to viewpoint and illumination changes. Some recent techniques explicitly model the semantics of the scene to obtain higher robustness towards seasonal changes [61, 151, 152]. More precise visual keypoint-based place recognition methods are well studied [87] and of significant interest [153], but they present their own set of challenges with regards to scalability, viewpoint, and illumination changes. A compromise between precise keypoint localization and the ability to incorporate contextual and semantic information can be found in object-based localization frameworks [62, 154].

LiDAR based localization methods rely mainly on matching geometry and can be separated into various categories. Many PC registration methods, out of which Iterative Closest Point (ICP) [155] is the most well-known one, require a good pose prior and are not suited for global localization. While global registration methods exist that work beyond the local context [156, 157], they still require storing at least parts of the PC data. This can be partially mitigated by only extracting compact descriptors during map building and localization. Descriptors of single LiDAR scans [91, 158–160] only allow rough location estimates and not precise 6D poses that could be integrated into a SLAM pipeline. Descriptors of local keypoint neighborhoods [89, 161–163] in turn can be noisy and lack distinctiveness due to only having geometry data available. A combination can be achieved by performing a refinement step, *e.g.* using ICP, after the rough localization, but would again require storing and working with PC data [164, 165].

PC segment description and mapping based approaches were first proposed by Douillard *et al.* [166] and Nieto *et al.* [167] and then further extended into a full SLAM pipeline in *SegMap* by Dubé *et al.* [6]. *SegMap* leverages advantages from both local and global descriptors by looking at features in the environment that are large enough to be more robust and meaningful, while also maintaining the ability to produce accurate 6D poses. Extensions to the pipeline include different training methods for the descriptor, as well as fine tuning to different environments [42, 93].

Enriching PCs with semantic information, *e.g.* obtained by fusing camera and LiDAR data, has proven beneficial [43, 168–171]. Ratz *et al.* [172] propose not only to use just one scan instead of an accumulated PC but also to enrich the descriptor with appearance information from a camera using NetVLAD and a customized fusion layer in their network. This increases accuracy at the cost of a significant decrease in computational performance, due to the expensive NetVLAD backbone. Also, the segmentation procedure is still purely based on geometry and cannot incorporate any additional information from the camera. Schönberger *et al.* [131] propose a localization scheme based on semantically labeled 3D PCs able to localize in the presence of severe viewpoint and appearance changes. They extract a deep-learned descriptor for a subvolume of the semantic map and compare it to similarly extracted ones from a prior map. However, using such submaps only provides a rough place-recognition like localization and require further refinement steps to get 6D poses. Furthermore, by operating on comparably larger subvolumes and the need to check multiple hypothesis, the corresponding descriptor network is computationally expensive.

In contrast, *SemSegMap* introduces a framework that is both efficient enough to be run in real-time on a consumer CPU (excluding the semantic segmentation running on the GPU) and still able to leverage the rich information available from geometry, appearance and semantics. We perform a very early fusion of the two sensor modalities which allows also the segmentation to be based on all available information.

3 SEMSEGMAP

In this section we present the details of the proposed *SemSegMap* pipeline, as shown in Figure 6.2. The approach can be split into several key modules, out of which the new segmentation and descriptor explained in Sections 3.2 and 3.3, represent the core contributions of this paper.

3.1 *Semantic enrichment*

The inputs to the pipeline consist of a stream of color images and PCs. The color images are passed through a semantic segmentation network to obtain a semantic class for each pixel. Using the extrinsic calibration between the camera and the LiDAR as well as the intrinsic calibration parameters of the camera, the color and semantic class of each pixel is projected onto the PC. The result is a set of enriched points in the form $\mathbf{p} = \{x, y, z, h, s, v, c\}$, where x,

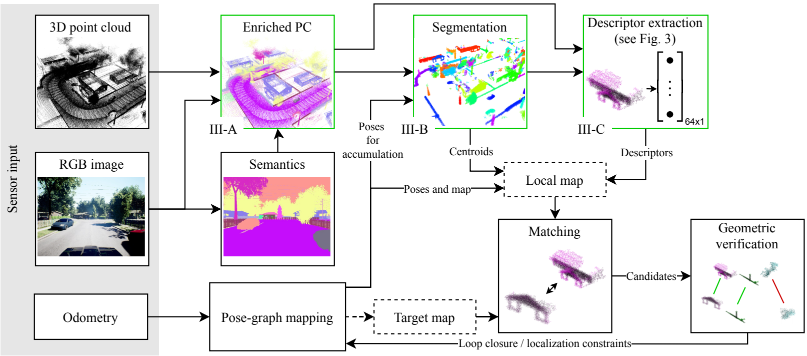


Figure 6.2: Overview of the *SemSegMap* pipeline. The whole pipeline can be run in *localization mode* with a target map loaded from the disk or in *loop closure mode* where the target map is provided by the current pose-graph. Green: Main modules changed from [6] with corresponding section numbers.

y, and z are the spatial coordinates, featuring also color values h, s, and v in HSV space (result visible in Figure 6.1), and semantic class labels c (example depicted in Figure 6.2: Enriched PC).

3.2 *Semantic segmentation*

To remove noise and achieve a more compact data representation, we accumulate the enriched PC data in a fixed size voxel grid. The voxel grid is a cylinder with a radius of R that dynamically follows the robot and is centered on it. For each voxel, the color information of multiple points is fused by using a running average over the incoming values to obtain the current value for the voxel. In contrast, the semantic class labels can not be averaged, and therefore, all values are stored and the semantic label of the voxel is determined by majority voting. Further filtering can be done by excluding points that belong to known dynamic classes *e.g.* humans and cars.

We use an incremental Euclidean segmentation that does not need to be rerun on the entire PC at each step, but is computed incrementally only on the newly active voxels, as detailed in [6]. A segment $\mathcal{S}$ is defined as a set of points, where for each point $\mathbf{p}_1 \in \mathcal{S}$ there exists at least one other point $\mathbf{p}_2 \in \mathcal{S}$, so that the distance between these two points is smaller than the segmentation distance $d_{segment}$.

To include the semantic and color information into the segmentation process, we modify the standard Euclidean distance function between two points $\mathbf{p}_1$ and $\mathbf{p}_2$ to be

$$\begin{aligned} d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 + \\ f_h^2(|h_1 - h_2|) + f_c^2(c_1, c_2). \end{aligned} \tag{6.1}$$

The function f_h is used to compute the distance between two colors by only comparing the difference in hue values in order to mitigate the appearance variance and is defined as

$$f_h(\Delta h) = \begin{cases} p_h & \min(\Delta h, 1 - \Delta h) > t_h \\ 0 & \text{otherwise} \end{cases}, \tag{6.2}$$

where p_h is a fixed penalty for when the color difference is above a certain chosen threshold t_h and where the hue values h are normalized to $[0, 1)$. In practice, there are two cases because the hue color space is cyclical. Similarly, the semantic class distance function is defined as

$$f_c(c_1, c_2) = \begin{cases} p_c & c_1 \neq c_2 \\ 0 & \text{otherwise} \end{cases}, \tag{6.3}$$

where p_c is a fixed penalty applied when the semantic classes do not match. Fixed penalties are necessary because the color space and semantic spaces are either not continuous or not numerically comparable to physical distances in space. A soft constraint on the segmentation can be imposed by choosing p_h and p_c smaller than $d_{segment}$, which means that two points that are sufficiently close in space can still be part of the same segment, even if they have a very different color or class.

During the segmentation process, at each step, the robot extracts a set of segments in the local map around itself. Those segments slowly accumulate points as more observations are made from different viewpoints. Similarly to how a keypoint is tracked, a segment will have multiple accumulated observations. Therefore, the final observation, just before it moves out of the local map neighborhood, will be the most complete one.

3.3 *Description*

For each segment observation, a learned descriptor is calculated and the local map is built by associating each descriptor to the corresponding segment

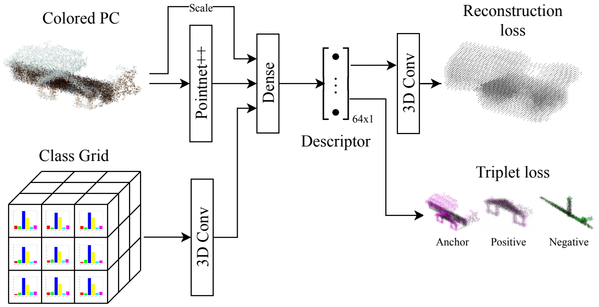


Figure 6.3: The descriptor extraction of the colored PC is based on a Pointnet++ [173] backbone while the semantics are fused in a coarse voxel grid of semantic class histograms. The network is trained using both a reconstruction loss and a triplet loss.

centroid point. For efficiency reasons, we only keep the descriptor of the last and most complete observation of each segment to create the target map for subsequent localizations or loop closures.

The new descriptor network, illustrated in Figure 6.3, uses the Pointnet++ architecture [173] that is based on hierarchical point set feature learning. The input to the Pointnet++ backbone is the colored PC segment that has been randomly subsampled to a fixed size of 2048 points. The class labels are instead accumulated into a very coarse $3 \times 3 \times 3$ spatial voxel grid, where each cell contains a normalized histogram of the class labels that fall inside that section of the PC segment. This very coarse description is enough because due to the semantic segmentation process the class labels inside most segments are relatively homogeneous. For the sake of computational efficiency this class representation is handled separately by a small 3D convolutional network, whose output is later concatenated with that of the Pointnet++ backbone. Finally, we also input into the network the scaling factor by which the point coordinates were normalized, which helps the network better discriminate between segments that do not match but are either visually or geometrically similar.

The descriptor is trained using both a triplet loss as well as a reconstruction loss from a convolutional decoder. The triplet loss is formulated as

$$L_{\text{triplet}} = \max(m + \sigma(\mathcal{A}, \mathcal{P}) \cdot D_{\mathcal{A},\mathcal{P}} - D_{\mathcal{A},\mathcal{N}}, 0) \tag{6.4}$$

where $m = 0.4$ is the margin, $D_{\mathcal{A},\mathcal{P}}$ is the Euclidean distance between an anchor segment $\mathcal{A}$ and a positive example $\mathcal{P}$, $D_{\mathcal{A},\mathcal{N}}$ is the Euclidean distance between the anchor and a negative example $\mathcal{N}$, and $\sigma(\mathcal{A},\mathcal{P}) = \frac{\mathbf{card}(\mathcal{P})}{\mathbf{card}(\mathcal{A})}$ is the ratio in number of points between anchor and positive example. The scaling $\sigma(\mathcal{A},\mathcal{P})$ is a heuristic that prevents the loss function from penalizing too much segment observations that should match, but due to segment incompleteness only share a small overlap and therefore in practice are hard to match. For the reconstruction loss we use a binary cross entropy loss applied to each voxel. This enables approximate reconstructions of the PC map from only the descriptor space for visualization and improves the descriptor quality of very similar looking segments. During training we augment the PCs in multiple ways, including both geometric variations such as random rotations, jitter, scale shifts, missing points or sections, and visual variations such as color shifts or erronous class labels.

3.4 *Localization and loop closure*

To perform localization or loop closure, candidate correspondences are identified between the locally built map of segments and the prior or global map, respectively. The candidates are identified using the descriptors of each locally visible segment and retrieving the k most similar descriptors from the global map. Finally, a geometric verification step is performed based on the centroids of the target and query match candidates to identify the 6 degrees of freedom (DoF) transformation that leads to the largest set of inliers using random sample consensus (RANSAC). The resulting transformation can either be used as a localization result when localizing from a previously built map, or as a loop closure constraint in a SLAM scenario. In the latter case, both the loop closure constraint and robot odometry constraints are placed into an online pose graph based on iSAM2 [28].

4 EXPERIMENTS

In this section, we describe our training procedure and present thorough evaluation of *SemSegMap* on both simulated and public real-world datasets, demonstrating a superior performance compared to different baselines on segmentation, descriptor quality, and localization accuracy and robustness.

4.1 *Datasets*

Datasets including visual as well as PC data, spanning large areas and covering different environmental conditions that include semantic annotations are

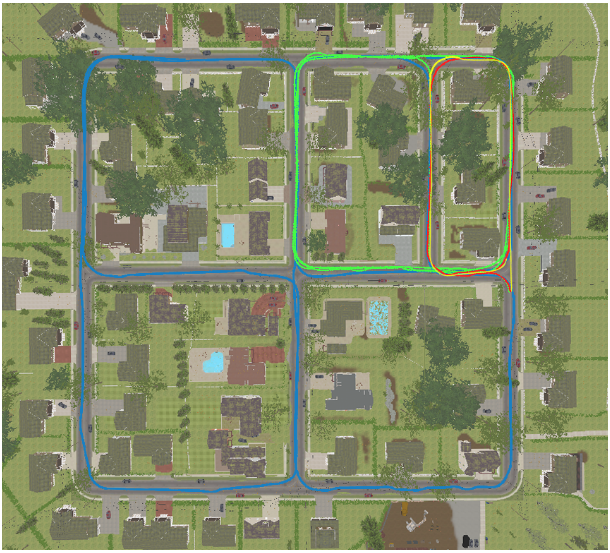
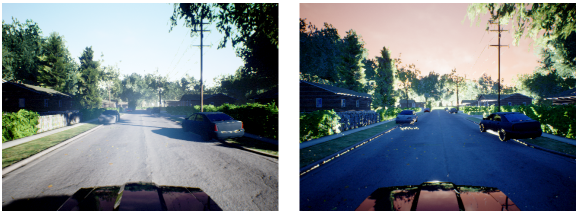


Figure 6.4: Bird's eye view of the simulation environment including the simulated trajectories (blue: `S0`, green: `S1`, red: `S2`, yellow: `S3`).

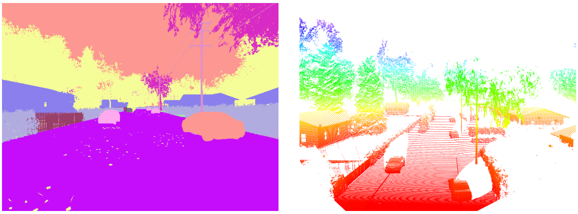
rare and hard to obtain. Therefore, for the data intensive step of descriptor training, we utilize simulated data to be able to quickly produce training data for *SemSegMap*. Furthermore, simulated datasets provide the opportunity to specifically evaluate our contribution in isolation of sensor noise, state estimation and calibration inaccuracies, and imperfect semantic segmentation. In addition, we demonstrate the transferablility of our approach to a challenging real-world dataset.

Simulation

Photo-realistic simulation is a popular tool for efficiently generating visually rich data with high quality ground truth annotations. Some of the most popular simulation tools in the robotics domain are Gazebo [174], CARLA [175], LGVSL [176] and AirSim [177]. While CARLA and LGVSL focus on autonomous driving scenarios, Gazebo does not provide photorealistic visual output. The datasets used in the following experiments were generated by



(a) RGB image: Sunny day (b) RGB image: Sunrise



(c) Semantic segmentation (d) 3D LiDAR PC

Figure 6.5: Overview of simulated sensor modalities and environmental conditions.

AirSim using the "Modular Neighborhood Pack"[1], a large residential environment. Each dataset consists of RGB image data, a semantic segmentation map for the image, LiDAR PCs as well as odometry information. The image data consists of three cameras, one on each side and a front facing one, spanning a total horizontal field of view (FoV) of $270\,^{\circ}$. The simulated LiDAR has a resolution of 1920×64 and the same $270\,^{\circ}$ FoV. All sensors are synchronized and operated at $5\,\mathrm{Hz}$.

An overview of the neighborhood, the simulated environmental conditions, as well as the simulated trajectories, is given in Figure 6.4 and 6.5, and Table 6.1, respectively.

Real world data

For demonstrating the applicability of our approach to real-world environments, we use the NCLT dataset [178]. The dataset provides raw sensor data from a LiDAR sensor, an omnidirectional camera, and ground truth position data, among others. As the images lack ground-truth semantic labelling, we used the author's implementation[2] from [179] to semantically segment

---

[1] `https://www.unrealengine.com/marketplace/en-US/product/modular-neighborhood-pack`

[2] `https://github.com/NVIDIA/semantic-segmentation`, accessed on 8th of November 2020

Table 6.1: Overview of simulated trajectories. The referenced trajectories as well as environmental conditions are shown in Figures 6.4 and 6.5, respectively.

#	Length	Conditions	Comment
`S0`	2071 m	daytime, sunny, clear	Covers whole map
`S1`	1462 m	daytime, sunny, clear	Medium size with overlap
`S2`	377 m	daytime, sunny, clear	Single small loop
`S3`	358 m	early sunrise, shadows	Opposite direction to `S2`

Table 6.2: Overview of the trajectories extracted from the NCLT dataset. Please refer to [178] for a detailed description of the dataset.

#	Date	Time	Conditions
`N0`	2012-04-29	4 min – 30 min	morning, sunny, foliage
`N1`	2012-05-26	0 min – 10 min	evening, sunny, foliage
`N2`	2012-11-04	0 min – 25 min	morning, cloudy
`N3`	2013-04-05	54 min – 69 min	afternoon, sunny, snow

the images from the five horizontal color cameras. The camera-to-LiDAR extrinsics, provided in the dataset, were used to enrich the PCs with color and semantic information by projection onto the image plane. As a last step, we removed the local ground plane from the enriched PC in order to make the subsequent segmentation process more robust. Because the dataset was recorded on a SegWay, which experiences a lot of back and forth pitching motion, the ground plane cannot simply be removed by setting a height threshold on the raw PC. Instead, we estimated the local ground plane based on the points in the enriched PC whose class labels correspond to *ground* and subsequently removed all points in close proximity to these points. A subset of the dataset featuring large map overlap and different environmental conditions was processed and used for our experiments as listed in Table 6.2.

4.2 *Segmentation*

To evaluate the segmentation method presented in Section 3.2 we use the simulation dataset run `S1` and the NCLT run `N0`. For the classic Euclidean segmentation in *SegMap* we use the default segmentation distance $d_{segment} = 0.2$. In *SemSegMap* we adjust this to $d_{segment} = 0.3$ and set the penalties and thresholds in the color and semantic distance functions to $t_h = 0.1$, $p_h = 0.05$ and $p_c = 0.15$.

To measure the segmentation quality, we first calculate the convex hull V_i of each segment S_i in the local map, and then compare it to the segment S_j with the convex hull V_j at the same global location in the target map. The quality of the segmentation is then given by the intersection over union (IOU) as

$$\mathrm{IoU}(V_i, V_j) = \frac{V_i \cap V_j}{V_i \cup V_j}. \tag{6.5}$$

In this way, we compare how repeatable and accurate the segmentation is when visiting the same place multiple times. Figure 6.6 shows the segmentation IOU results from run S1 in the simulated environment which features multiple intersecting loops in the same environment and N0 for the NCLT dataset. A low IOU indicates that the segment was not properly re-segmented when re-visiting the place, while a high IOU corresponds to a consistent re-segmentation. In the latter case the segments occupy the same volume in space and are therefore more likely to be matched based on their descriptors. We consider an $\mathrm{IoU} \geqslant 0.33$ to represent a good overlap while an $\mathrm{IoU} < 0.33$ represents an inconsistent segmentation. *SemSegMap* produced $-24.24\,\%$ and $-8.38\,\%$ less inconsistent segments with $\mathrm{IoU} < 0.33$ while obtaining $30.97\,\%$ and $25.03\,\%$ more consistent segments with an $\mathrm{IoU} \geqslant 0.33$ with respect to *SegMap* for the simulation and NCLT dataset, respectively.

4.3 *Descriptor quality*

To evaluate the impact of our adapted descriptor extraction network described in Section 3.3, we performed a similar experiment as described in [6, Section 5.4].

In order to use the segments as landmarks in a localization and loop-closure scenario as described in Section 3.4, we obtain candidate matches using k nearest neighbours (k-NN). For a more robust and efficient localization, the amount k of candidates necessary to retrieve the correct match should be as small as possible, even with partially observed segments which occur during live operation or different directions of travel. In Figure 6.7, we show a comparison of how high k needs to be in order to retrieve the correct match using *SemSegMap* and different state-of-the-art PC descriptors operating on the simulation dataset `S0` and the NCLT dataset `N0`. The *SemSegMap* network is trained solely on ground truth data obtained from `S1` and not retrained on the NCLT data.

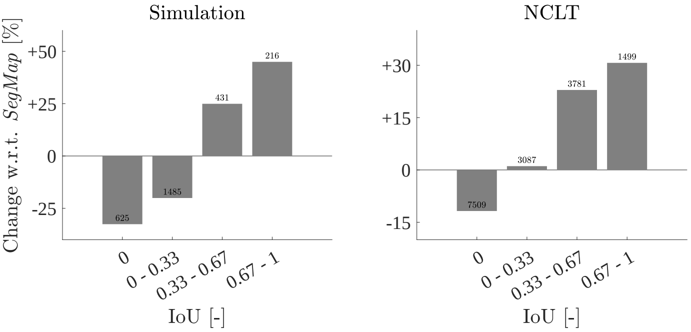


Figure 6.6: Change of the segmentation IOU for the simulation dataset `S1` (left) and NCLT `N0` (right) using *SemSegMap* with respect to *SegMap*. The bins represent the change in number of segments with a specific IOU range. The small numbers on top of the bars depict the absolute numbers of segments in that range produced by *SemSegMap*. While *SemSegMap* produces less segments with low IOU, which represent inconsistent segmentations, it is able to produce more consistent segments with a high overlap and IOU compared to *SegMap*.

We compare the quality of our descriptor to a learned segment descriptor [6] for the simulated data, and extended the evaluation for the NCLT dataset to the hand-crafted descriptor FPFH [89] and the learned local descriptor 3DSmoothNet [163]. For FPFH, a single segment descriptor was computed by selecting the segment centroid as the keypoint and choosing the descriptor radius to encompass all points of the respective segment [180]. In the case of 3DSmoothNet, simply extending the radius did not yield meaningful results. One reason could be that the descriptor was specifically designed and trained as a local descriptor. In order to deploy it within our segment-based framework, we randomly select points from the segment PC as keypoints, extract local descriptors for each keypoint and aggregate them to a global descriptor by following a Bag-of-Words (BoW) approach using k-means clustering. Specifically, we trained the k-means model on a subset of segments extracted from `N0`, where $k = 16$ yielded the best results.

Note from Figure 6.7 that *SemSegMap* outperforms all other descriptors in both datasets except for FPFH with incomplete segments with completeness $< 30\,\%$. This property of the *SemSegMap* descriptor is controlled by the σ term in the triplet loss formulation introduced in Equation 6.4 that biases the network towards better matching more complete segments. Removing the influence of this term causes better matching performance for harder matches, *i.e.* at low completeness thresholds, but at the same time reduces the

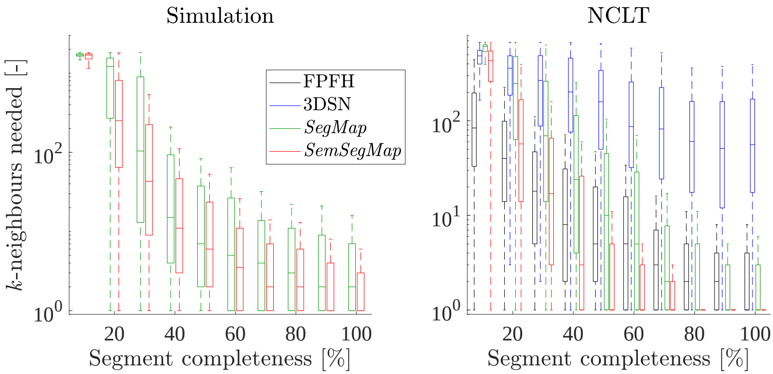


Figure 6.7: Comparison of descriptor quality on the simulation dataset `S0` (left) and the NCLT dataset `N0` (right).

matching performance of more complete segments as the descriptor attempts to perform more unlikely matches. However, in a typical SLAM application, while an early localization, *e.g.* with many incomplete segments is important, achieving more accurate and robust localizations during operation with complete segments is often of greater interest in a full smoothing and mapping framework. The descriptor transfers well to the more challenging real-world data from the NCLT dataset, where the performance difference is even more evident.

4.4 *Localization accuracy and robustness*

To assess the localization accuracy and robustness, we tested *SemSegMap*s ability to localize in a previously built target map. Therefore, we recorded the target map on trajectories `S0` and `N0` for the simulation and NCLT dataset, respectively. The map for `S0` contains 2006 segments and has a size of 0.51 MB while for `N0` there are 2588 segments in a 0.66 MB map. All other trajectories except for the training set `S1` are used for on-line localization against the target map as described in Section 3.4.

For the localization evaluation we retrieve a total of 16 neighbours using k-NN. For the geometric verification in our experiments the RANSAC was set to require a minimum of 6 and 7 inliers for the simulation and NCLT dataset, respectively, and allow a centroid distance of at most 0.4 m. For all the experiments for both *SegMap* and *SemSegMap* we keep these parameters

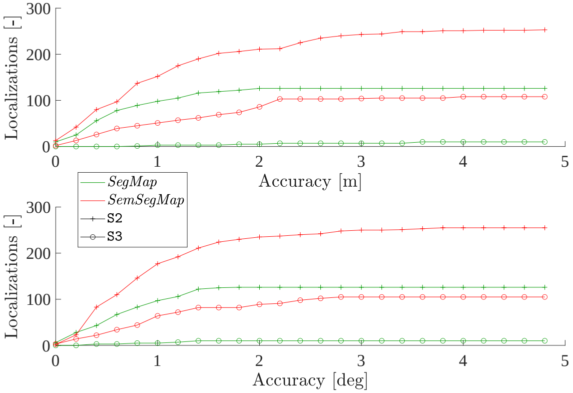


Figure 6.8: Cumulative successful localizations with a certain accuracy on the simulation datasets with a target map built from `S0`.

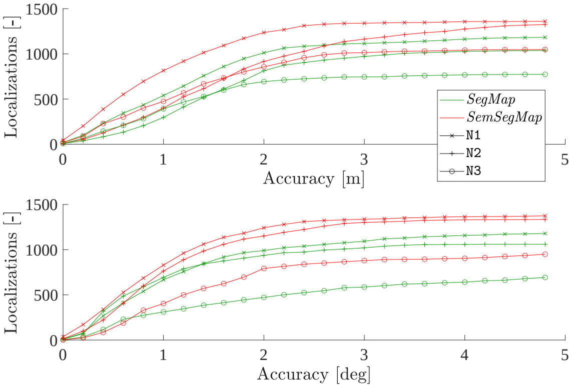


Figure 6.9: Cumulative successful localizations with a certain accuracy on the real-world datasets with a target map built from `N0`.

fixed and only change the segmentation and description process as outlined in Sections 3.1 and 3.3, to provide the fairest comparison. With these settings *SemSegMap* (excluding the semantic segmentation) runs on an Intel i7-8700 CPU with an average frequency of 6.13 Hz and 6.74 Hz on the simulation and NCLT dataset, respectively.

Table 6.3 and Figures 6.8 and 6.9 report the accuracy results of the estimated 6-DoF pose for the simulation and NCLT dataset, respectively. *SemSegMap* is

Table 6.3: Localization accuracy results overview ($n_{<\bullet \mathrm{m}}$ relative improvement of *SemSegMap* with respect to *SegMap* with a certain accuracy).

#	Number of localizations *SegMap*	*SemSegMap*	$n_{<1\,\mathrm{m}}$	$n_{<5\,\mathrm{m}}$
S2	126	258	53.93 %	100.79 %
S3	10	108	4400 %	980 %
N1	1201	1395	60.00 %	14.96 %
N2	1060	1337	44.17 %	27.55 %
N3	774	1056	41.70 %	35.58 %

able to consistently find more localizations throughout all the tested datasets. In simulation, less affected by odometry and sensor noise, *SemSegMap* is able to find a total of 102 % more high accuracy localizations (translation error of less than 1 m) with respect to *SegMap* and 165 % more accurate localizations (translation error of less than 5 m). Especially on trajectory `S3`, *SegMap* suffers from different appearance and viewpoint while *SemSegMap* is less affected and still able to produce many accurate localizations. Those results also transfer to the real-world dataset where more than 50.9 % high accuracy localization and 24.7 % more accurate localizations are found compared to *SegMap*.

5 CONCLUSIONS

In this paper, we introduced *SemSegMap*, an extension to *SegMap* that fuses both color and semantic information from an RGB camera with LiDAR data in real-time. In a real-world robotic application, the addition of cameras to a platform equipped with a LiDAR is typically easily possible due to the comparably low price of cameras and their cross-purpose use, especially when also performing semantic segmentation to improve scene understanding. We include this additional modality both to improve segmentation and descriptor quality, which we showed in a stimulated dataset with accurate ground truth and a challenging real-world dataset.

Using the described extensions, *SemSegMap* is able to outperform a geometric segmentation approach by producing less inconsistent segments and more highly overlapping segments when re-visiting a place. The tight fusion of the additional information in the descriptor also increases descriptor quality where *SemSegMap* not only outperforms the *SegMap* baseline but also other state-of-the-art PC descriptors like FPFH and 3DSmoothNet in terms of k-NN required to find the correct match. These improvements also propagate to the localization accuracy and robustness resulting in *SemSegMap* providing 102 % and 50.9 % more high accuracy localizations than *SegMap* for the simulated and the real-world dataset, respectively.

To further extend our framework, a combination of FPFH and *SemSegMap* descriptor based on the expected completeness of a segment could be used in order to benefit from both advantages.

PAPER IV

LEARNING CAMERA MISCALIBRATION DETECTION

Andrei Cramariuc*, Aleksandar Petrov*, Rohit Suri, Mayank Mittal, Roland Siegwart, and Cesar Cadena

*contributed equally

ABSTRACT

Self-diagnosis and self-repair are some of the key challenges in deploying robotic platforms for long-term real-world applications. One of the issues that can occur to a robot is miscalibration of its sensors due to aging, environmental transients, or external disturbances. Precise calibration lies at the core of a variety of applications, due to the need to accurately perceive the world. However, while a lot of work has focused on calibrating the sensors, not much has been done towards identifying when a sensor needs to be recalibrated. This paper focuses on a data-driven approach to learn the detection of miscalibration in vision sensors, specifically RGB cameras. Our contributions include a proposed miscalibration metric for RGB cameras and a novel semi-synthetic dataset generation pipeline based on this metric. Additionally, by training a deep convolutional neural network, we demonstrate the effectiveness of our pipeline to identify whether a recalibration of the camera's intrinsic parameters is required or not. The code is available at `http://github.com/ethz-asl/camera_miscalib_detection`.

Published in:
IEEE International Conference on Robotics and Automation (ICRA), 2020

DOI: 10.1109/ICRA40945.2020.9197378

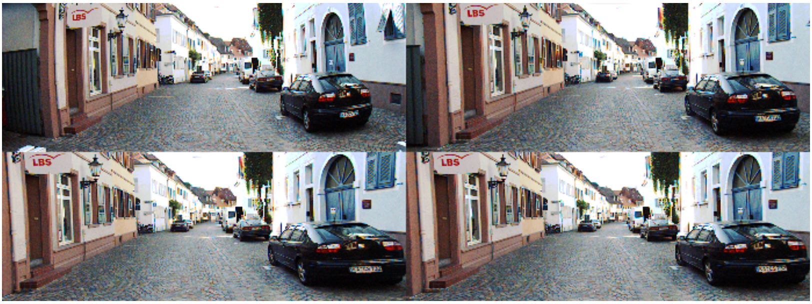


Figure 7.1: Illustration of the subtle differences that a miscalibration detection system needs to be sensitive to. **Top left:** Unrectified image; **Top right:** Correctly rectified image; **Bottom:** Two examples of incorrectly rectified images. The image is taken from the KITTI dataset [19].

1 INTRODUCTION

In robotics, errors in the estimation of the system's parameters can adversely affect the accuracy of algorithms for state estimation and the performance of feedback controllers. In order to avoid systematic errors due to incorrect parameter estimates, a common practice is to perform sophisticated calibration of the system by a human expert [181]. Once determined, calibration parameters are kept fixed during the operation cycle of the robot. However, this approach is not sustainable for a variety of real-world applications where robots need to operate in harsh environments for extended periods of time. The calibration parameters of the system are prone to change over time due to component wear, environmental transients such as temperature changes, or external disturbances like collisions. Additionally, it may be impractical to perform offline calibration regularly as a means to address this issue.

An alternative solution to offline calibration is online calibration techniques that are performed during the system's normal operation, such as those presented in [182, 183]. These techniques, though promising, are computationally expensive and have various limiting requirements, such as the type of required motion or the storage and processing of data to create a calibration dataset. Hence, instead of running these methods periodically and recalibrating the robotic platform, ideally, one would like to perform the calibration only when the system is detected to be miscalibrated. This objective is considered as a constituent of the fault detection and diagnosis for a robotic system [184].

Since sensors lie at the core of any autonomous system, it is critical to detect the sensor data faults for safety and stable performance [185]. However, unlike sensors for measuring attitude or temperature, calibration errors in vision sensors do not appear as an offset or as a drift in the sensor's readings. Although having hardware redundancy is a way to detect imperfections, it increases the cost and complexity of the system. Further, due to their complex nature, it is difficult to obtain a unified analytical solution for identifying miscalibration in vision sensors. Fortunately, common operating environments, both indoors and outdoors, contain regularities that can be exploited for this purpose, such as walls, furniture, street lamps, etc. We propose a data-driven approach that implicitly utilizes these regularities.

In this work, our goal is not to provide a neural network that detects miscalibrations for any camera. Instead, we propose a method in which a network is tuned for a specific camera to predict when an automatic recalibration is necessary for that camera. However, using a learning-based approach poses its own set of challenges. First, a large-scale dataset for training a network to detect miscalibration is not currently available in the public domain. Second, there is no standard metric for measuring the degree of miscalibration in the intrinsics of a camera. We address these challenges and provide the following key contributions:

- A novel dataset generation pipeline to create a large-scale dataset for camera miscalibration detection.

- A metric, *average pixel position difference*, for estimating the degree of miscalibration and analysis of how it correlates with performance in a monocular odometry task.

- A deep convolutional neural network (CNN) that predicts when the camera is miscalibrated even in previously unseen scenes.

2 RELATED WORK

In the last decade, a variety of approaches have been proposed for calibration of various types of range-based sensors, inertial sensors, and vision sensors, as well as the extrinsic calibration between them. In this section, we focus on literature related to sensor miscalibration, the estimation of intrinsic parameters of vision sensors, and fault detection in multi-sensor systems.

Accurate camera calibration is an important step for a multitude of 3D computer vision tasks. The existing calibration techniques can be broadly categorized as photogrammetric calibration and self-calibration. In photogrammetric calibration, the camera calibration is performed by observing a target of known geometry in 3D space. Over the years, various types of tags such as checkerboards [186, 187] and fiducial markers [188] have been proposed for this purpose. These approaches typically pose the calibration problem as a non-linear optimization problem to minimize a reprojection error and to estimate the most likely values of the camera parameters. However, the need to have an apparatus and a human expert in these techniques prevents them from being scalable or practical for robots deployed into the real world.

On the other hand, self-calibration, as introduced in [189], does not require a calibration object. Through a sequence of images, these methods estimate the intrinsic parameters that are consistent with the underlying projective reconstruction of the observed scene. Certain approaches use camera motion constraints, such as planar motion [190] or rotation of the camera [191], in conjunction with the 3D metric reconstruction of the scene to calibrate the camera's intrinsic parameters. Sturm [192] presents the concept of critical motion sequences for a camera with constant parameters for which there exists no unique solution for self-calibration. Wildenauer and Hanbury [193] detect orthogonal vanishing points in the scene to generate a hypothesis for focal length. However, the flexibility provided by self-calibration techniques comes at the price of computation expenses.

More recently, with the advent of deep learning [194], data-driven approaches have also been proposed to estimate the calibration parameters of the camera. Workman *et al.* [195] propose a CNN for estimating the focal length of an image. To train the network, they construct a dataset by combining images and camera models estimated using 1D structure from motion [196]. On the other hand, Lopez *et al.* [197] use separate regressors, which share a common pre-trained network architecture, to estimate tilt, roll, focal length, and radial distortion parameters from a single image. They use the SUN360 panorama dataset [198] to artificially generate the training images. However, the estimation of these parameters is highly dependent on finding the horizon in the image, an assumption that is highly environment dependent. Unlike the previous two approaches, which aim to estimate the camera calibration parameters, Yin *et al.* [199] propose an end-to-end multi-context deep network for removing distortions from single fish-eye camera images. They use a scene-parsing network to provide semantic cues during training and use an L_2 reconstruction loss for rectified image prediction.

Fault detection in multi-sensor systems can be done by correlating the information from multiple sensors and rejecting measurements that do not

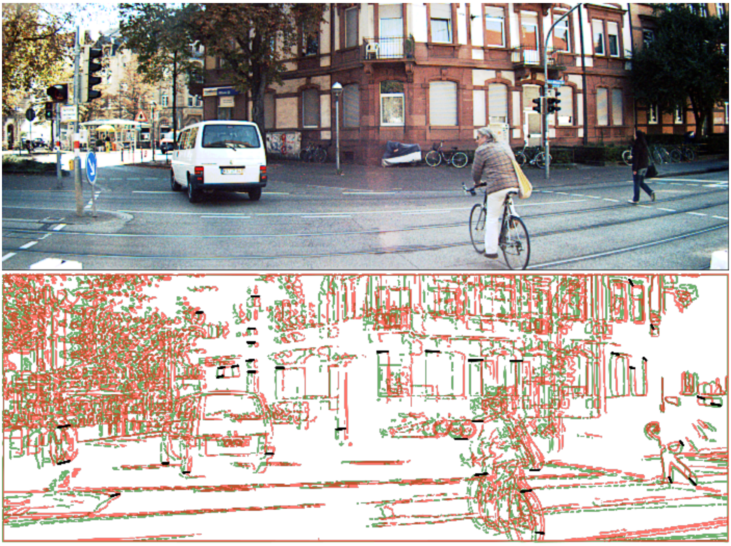


Figure 7.2: **Top:** An unrectified image from the KITTI dataset [19]. **Bottom:** For illustration purposes, canny edges detected from correctly (in green) and incorrectly (in red) rectified images are shown. A set of incorrect rectification parameters results in pixel projections from the raw image to be displaced relative to that with the correct parameters (indicated by the black segments). The mean of the L2-norms of these displacements over the image corresponds to the APPD.

match [200–204]. These methods are generally able to detect when a fault has occurred, however they rely on a redundant sensor setup. When the fault estimation is done indirectly, through an intermediary task such as localization performance, it can be ambiguous to decide whether the sensor is at fault or if the localization system failed.

While some of the above mentioned approaches deal with estimating the calibration parameters through either a geometry-based or a learning-based approach, our work is orthogonal and does not aim to replace them. We want to complement these methods by identifying when a camera needs to be recalibrated. Further, we do not want to rely on sensor redundancy since that increases the cost of the system. Thus, our objective is to detect miscalibration in a single camera. To the best of our knowledge, this is the first work proposing a deep learning approach to detect miscalibration of the intrinsic parameters for an RGB camera.

3 METHODOLOGY

In this section, we present our contributions in detail. The dataset generation pipeline is explained in Section 3.1. In Section 3.2, our novel metric for miscalibration is defined. The neural network and its training are detailed in Section 3.3.

3.1 *Dataset Generation*

A straightforward procedure to create a dataset for camera miscalibration detection is to manually vary the camera parameters by using different lenses while taking any one of the settings as the nominal one. However, this process is time-consuming and tedious since offline calibration would be required for every new setting. The procedure is also limited with respect to the generation of disturbances in the camera parameters. Some cameras have only one degree of freedom for calibration (the distance between the lens and the sensor), hence the calibration parameters cannot be varied independently. Due to these limitations, we propose an alternative solution to generate a semi-synthetic dataset by using a set of raw images and a set of correct calibration parameters for a given camera setup. The presented method is based on the idea that the visual effect obtained from rectifying an image from a miscalibrated sensor with its initial belief of the parameters is similar to the effect of rectifying an image from a calibrated sensor with parameters different from the correct ones.

In our semi-synthetic dataset generation pipeline, we consider the pinhole camera model with radial and tangential distortion [205]. We denote the set of true calibration parameters of the camera model as $\Theta = \{f_u, f_v, u_c, v_c, \mathbf{k}_r, \mathbf{k}_t\}$. Consider the raw camera image I, which is rectified using the parameters Θ to obtain the rectified image I'. The rectification map $M' = f(\Theta)$ used in this process relates each pixel in the rectified image to a position in the original image. Generally, not all the pixels in the rectified image have a corresponding position in the original one. Therefore, we define a validity mask, R, which is the largest rectangular region in the rectified image I' with only valid pixels, and that has the same aspect ratio as the original image I. The final sample image $\hat{I}$ is obtained by first cropping the valid mask region R of the image I' and then rescaling the result to the size of the original raw image I. An alternate way to express the rectification is by applying the rectification map $\hat{M}$ (which is obtained by cropping and rescaling M') on the raw image I to directly obtain the final sample image $\hat{I}$.

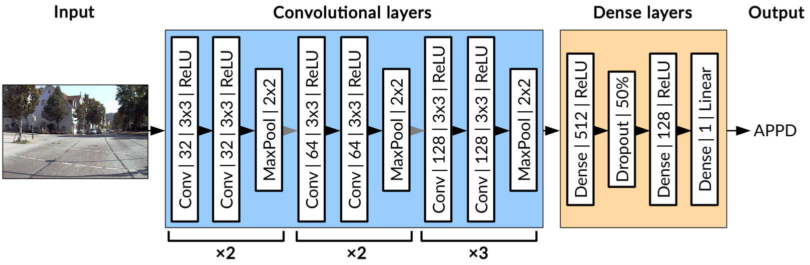


Figure 7.3: The network architecture used to run the experiments. All layers except the last one use Rectified Linear Unit (ReLU) activation functions.

In general, it is difficult to obtain the true calibration parameters of the camera. Thus, we use the values estimated using a calibration toolbox as the correct calibration parameters $\Theta^\star$ and denote the correct rectified image and rectification map as $\hat{I}^\star$ and $\hat{M}^\star$ respectively. To obtain samples of miscalibrated images, we perturb each intrinsic parameter independently to obtain Θ^m. This process allows generating arbitrarily many miscalibrated images, $\hat{I}^m$, and rectification maps, $\hat{M}^m$, by randomly perturbing the parameters. Thus, by collecting only a set of raw images with a correct calibration of the sensor, one can generate a large amount of data for detecting camera miscalibration. Even though we consider a pinhole camera model, this approach is also applicable to other camera models.

3.2 *Metric for Degree of Miscalibration*

As described in Section 3.1, image rectification is a transformation parameterized by the calibration parameters. Since these parameters are continuous, one can generate images arbitrarily close to a correctly rectified image by applying small perturbations to the true calibration parameters. Due to the non-linear effects and the strong correlations of these parameters on the rectification transformation, defining a meaningful distance metric to directly assess the quality of different randomly chosen calibrations is difficult. Moreover, as the rectification of an input image is typically only the first stage of a system, the degree of miscalibration should be considered in conjunction with the corresponding reduction in the overall system performance. Therefore, we propose an indirect approach using the *average pixel position difference (APPD)* as a scalar metric to measure the degree of camera miscalibration.

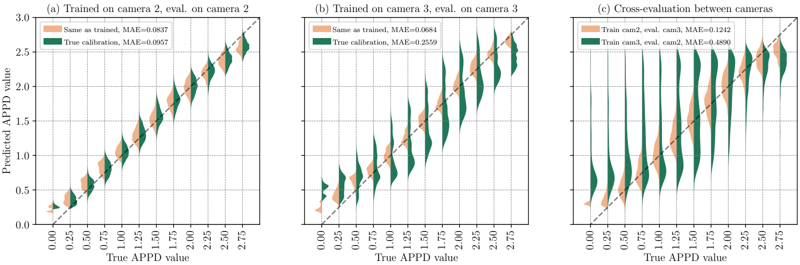


Figure 7.4: APPD prediction accuracy of the trained neural network models for the two RGB cameras from KITTI [19]. The plots show the distributions of networks predictions for given quantized APPD values. The dashed line designates perfect prediction.

Using the symbols introduced in Section 3.1, the numeric value for the APPD, denoted by δ, is calculated using the rectification maps $\hat{M}^\star$ and $\hat{M}^m$ obtained from using calibration parameters $\Theta^\star$ and Θ^m respectively. Since these maps are computed using different parameters, they relate the same pixel coordinate in their corresponding rectified images to a different image coordinate in the raw image. The Euclidean distance between these two positions is referred to as the *pixel position difference*. This is illustrated in Figure 7.2. The APPD is the mean value of these pixel position differences over the entire image, i.e.

$$\delta = \frac{1}{H \times W} \sum_{p \in I} \|\hat{M}^\star(p) - \hat{M}^m(p)\|_2,$$

where (H, W) is the size of the image I and p denotes the pixel coordinate (u, v). Even when normalized by the number of pixels, the value still depends on the resolution. Normalizing further by the diagonal makes it resolution-independent. That is why we report APPD values as a percentage of the image diagonal, i.e. they are divided by the diagonal and scaled by a factor of 100.

3.3 *Network Architecture and Training*

The architecture of the APPD prediction network is presented in Figure 7.3. The input to the network is the rectified image $\hat{I}^m$, and the output is the APPD metric. To prevent artifacts and loss of minute details due to image resizing, we use the input at full resolution.

For each camera and corresponding correct calibration, we train a separate network to deploy alongside the respective camera. This can be seen as an addition to the calibration procedure that, in a similar manner, is also pre-computed for each camera separately. The goal of the dataset generation process described in Section 3.1 is to reduce the amount and variety of data required to train the model. With the proposed method, it is sufficient to collect a single dataset, with correct calibration known and without any manual labeling.

During training, the perturbed parameters are sampled such that the calculated APPD values follow an approximately uniform distribution. Additionally, 1% of the samples are kept with the correct rectification, i.e. APPD value of zero. We use a mean squared error loss between the network predictions and the ground-truth labels for training. This loss is optimized by using the Adaptive Moment Estimation (ADAM) optimizer [137]. We initialize the network parameters by Xavier's initialization method [136] and use dropout [138] to avoid overfitting.

4 EXPERIMENTS AND DISCUSSION

The results from evaluating the trained models are discussed in Section 4.1. We present some generalization results for our approach in Section 4.2. The relationship between APPD and the intrinsic parameters is experimentally investigated in Section 4.3. Section 4.4 compares APPD and reprojection error.

4.1 *Detection of Miscalibrations with a Neural Network*

The KITTI dataset has two RGB cameras (cameras 2 and 3). We split the KITTI sequences from September 26, 2011 to obtain our training and validation sets. For testing, we use all sequences from the other four days. We vary the focal lengths from -5% to 20%, the optical center $\pm 5\%$, and the distortion coefficients $\pm 15\%$. The dataset provides different calibration files for every day, which were observed to be inconsistent. It is not known whether the cameras differed physically on different days, or if the differences in the calibrations arise from imperfections in the calibration procedure. Therefore, there is no single 'correct' calibration that can be used as a reference for calculating the true APPD value when evaluating prediction performance. Instead, we consider two cases: (i) taking the set of parameters corresponding

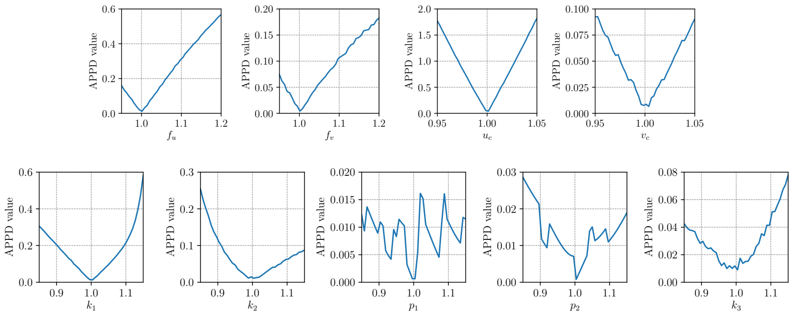


Figure 7.5: Plots showing the different effects of the camera's intrinsic parameters on the APPD, when one parameter is varied, and the rest are kept fixed. The x-axis is the multiplication factor applied to the reference parameter. The used reference calibration is from camera 2 for the KITTI sequences from date 26.09.2011. Note that the y-axis scales in the plots for f_u and f_v, as well as for u_c and v_c, are different. This is due to the image's aspect ratio.

to the day used for training, and (ii) using the set corresponding to the day on which the test image was actually recorded.

Figures 7.4a and 7.4b show the prediction quality of the trained networks for both camera 2 and camera 3, evaluated for the two cases described above. The mean absolute error (MAE) for each case is also reported. It can be seen that the models are able to generalize also to images and environments they have not seen before. While both networks are powerful in detecting miscalibration with respect to the reference set of parameters that they were trained with, the one for camera 2 performs significantly better.

This mismatch in performance is caused by the level of similarity between the sets of correct calibration parameters provided for each camera. The APPD ranges for the four test days, relative to the day used for training and validation are $[0.12, 0.93]$ and $[0.78, 2.37]$ for camera 2 and camera 3 respectively. It is likely that camera 3 might not have been well-calibrated either on the training day or on some of the other days. This result illustrates the importance of selecting a 'correct' calibration, with respect to which the training process must be defined.

As the two cameras are of the same make and brand, are positioned solely with a horizontal offset from one another, and operate in the same environment, the transferability of the model trained on the data from one camera to the other was also evaluated. The corresponding results are shown in Figure 7.4c. Indeed, the model trained on camera 2 generalizes well to camera 3. Figure 7.4c also shows that the reverse generalization does not hold, which stresses the importance of the choice of reference calibration and is another indication that camera 3 might have been slightly less consistently calibrated.

Figure 7.4 further demonstrates that the trained models experience bias in the extremely low and extremely high APPD values. This is a limitation of both training in a regression setting and of the miscalibration sampling procedure, which provides very few miscalibrations with APPD values close to 0. Instead, if one targets a specific performance metric, which can be related to APPD (see Section 4.4), then they can determine a threshold value and rephrase the problem into a binary classification setting.

While the presented neural network architecture is simple and further performance improvement may be possible, the above results indicate that a CNN can indeed be trained to be sensitive to miscalibration artifacts. One should note that the data does not explicitly designate the regularities which are not robust to the perturbation effects arising from disturbing a camera setup, but the model has discovered these regularities on its own. Moreover, even though motion distortion and blur are not explicitly addressed by the analysis, they are represented in both the training and test sets (as the images are obtained from a moving vehicle), and therefore the results account for them as well.

4.2 *Generalization to new Environments and Cameras*

The KITTI dataset is limited in the variation of its scenes (recordings only in the city of Karlsruhe, Germany), and in the position of the camera sensors (both oriented forward with only a horizontal offset between them). In order to study the potential further generalization capabilities of the proposed method, we trained the same model on some of the scenes recorded in Boston from the forward camera of the nuScenes dataset [206]. Only the scenes recorded during the day were considered.

The performance of the model was evaluated on the other scenes of the same camera in Boston, as well as on the forward camera in Singapore, and the forward-left and forward-right cameras in Boston. The results can be seen in Figure 7.6 and show that the accuracy is comparable in the four cases. The sets of intrinsic calibration parameters for the four cameras considered are almost the same and hence much closer in terms of APPD than the ones from the KITTI dataset (less than 10^{-5}). This consistency between the different

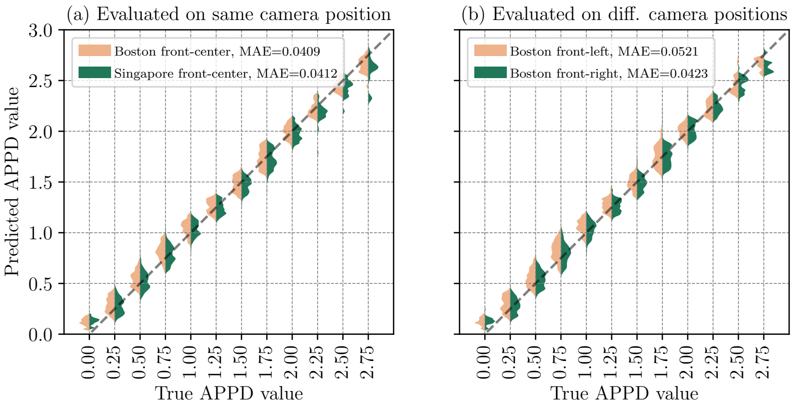


Figure 7.6: APPD prediction accuracy when evaluating on environments and camera positions that were not represented in the training set. The training was performed on a subset of the nuScenes dataset [206] recorded with the front-center camera in Boston. The plots show the distribution of predictions for given quantized APPD values. The dashed line designates perfect prediction.

sensors, as well as the higher resolution of the images, explains why the model trained on nuScenes exhibits better performance.

4.3 *Relationship between APPD and Calibration Parameters*

Some of the effects of the different intrinsic parameters on the APPD value are illustrated in Figure 7.5. The plots are obtained by individually varying one parameter while keeping the others fixed. The difference in effect when varying f_u and u_c compared to f_v and v_c, respectively, is due to the wide aspect ratio (2.72) of the image. This causes parameters along the u-axis to have a stronger effect on the distortion of the images. Another point to observe from Figure 7.5 is the noise, which is more visible at lower APPD values. This noise is due to quantization effects causing numerical imprecision when computing the undistortion maps. The amount of noise can be reduced, at the cost of more computation time, by calculating the APPD at a higher image resolution, but is not necessary for any practical purposes.

4.4 *Relationship between APPD and Reprojection Error*

Reprojection error is a standard measure of the deterioration of a robotic system's performance in various vision-related tasks [207]. Therefore, it is of interest to relate the APPD metric of a misrectification to the reprojection error it causes. As mentioned in Section 3.1, the physical scenario that we are interested in is when a camera experiences a hardware change without the corresponding change in intrinsic parameters. Data for such scenarios is difficult to obtain. Therefore we propose applying the reverse process: the physical sensor stays the same while the parameters are changed. We perform a few simple tests on simulated data to further analyze the relationship between APPD and reprojection error.

First, consider the case when the camera is kept unchanged, but the robot's belief of its intrinsic calibration is changed. One can generate a set of points in front of a virtual camera and then project them into the camera plane using the correct intrinsic parameters of the physical camera. These points can then be rectified with both the correct and incorrect sets of parameters. Since point associations are known, the reprojection error can be calculated as the average distance between the resulting rectified projections in the image plane. Figure 7.7a shows the obtained relationship. Indeed, APPD is a good measure of the reprojection error that arises from rectifying with a wrong calibration parameter set.

Second, it is of interest to know how the real physical scenario would relate with the reverse synthetic scenario in order to evaluate if the method outlined here can be applied to a real system. This can be achieved by repeating the above-described point-projection and rectification procedure, but keeping the intrinsic parameters for the rectification step fixed while varying the set for the projection step. This setting corresponds exactly to the physical situation but cannot be reproduced synthetically on a real image (we cannot 'reproject' reality with a different set of intrinsic parameters). The comparison between the resulting reprojection error and APPD can be seen in Figure 7.7b. The result is that there is no-longer an injective functional relationship from APPD to reprojection error, and the dependence between the two values is less pronounced.

APPD is easy to calculate for a real-world dataset as it is independent of the hardware that is used to obtain the image. Furthermore, it allows the sampling of an almost infinite number of different intrinsic calibration parameters, which is beneficial for training neural networks, which require large amounts of data. Nevertheless, it might not be the most accurate metric for detecting physical miscalibration. In fact, as Figure 7.8 shows, the

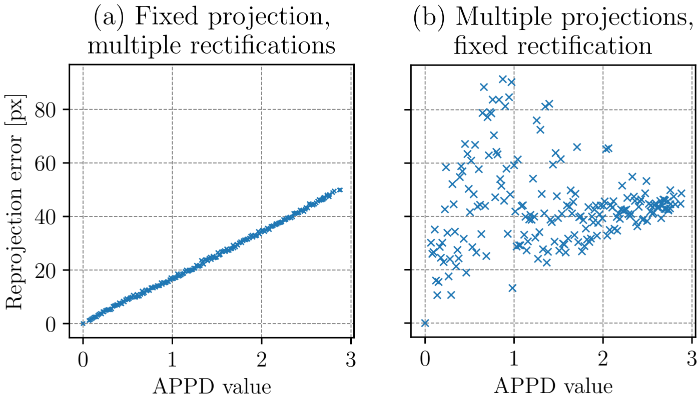


Figure 7.7: Plot between APPD and reprojection error for **(a)** when projecting a set of points with a single set of calibration parameters and rectifying with a variety of parameters, and **(b)** when projecting a set of points with a variety of calibration parameters and rectifying with a fixed set.

reprojection error as computed for the physical scenario is better correlated with the SLAM performance of a system. As mentioned above, the drawback of using reprojection error as a miscalibration metric is that it cannot be calculated for a real-world dataset. No procedure similar to the one in Section 3.1 can be constructed for the physical miscalibration case and its corresponding reprojection error. Therefore, one would need to create a dataset with various camera settings and the respective calibration parameters for each one, which can be impractically time-consuming as it needs to be repeated for each camera individually. The variety of possible calibrations would also be severely limited by the design of the lens, as most lenses only have one degree of freedom. Alternatively, a fully synthetic dataset, e.g. generated in simulation, can be used, but then transferability to real image data would be questionable.

The advantage of using APPD is that it facilitates training with very large sets of data that are easily obtained via the procedure detailed in Section 3.1. The type of artifacts introduced by the dataset generation in Section 3.1 can be considered similar to the ones introduced by a physical miscalibration. By demonstrating that APPD is learnable by a neural network, we show that it might be possible to also learn the reprojection error.

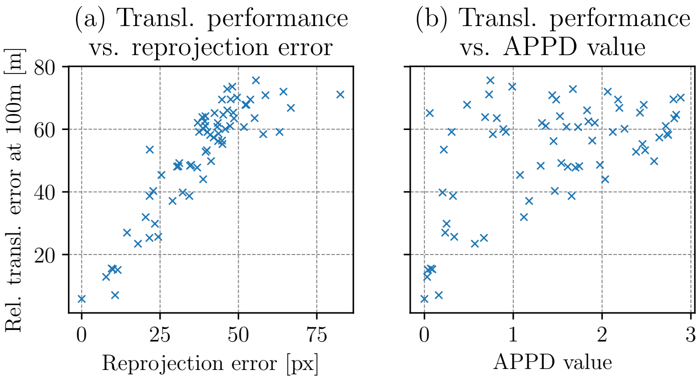


Figure 7.8: Plot between the performance of ORB-SLAM [59], evaluated on the KITTI odometry sequence 10, and **(a)** reprojection error when projecting a set of points with a single set of parameters and rectifying with a variety of sets, and **(b)** the corresponding APPD.

5 CONCLUSION

We proposed a novel semi-synthetic data generation procedure that requires no data labeling and a corresponding camera miscalibration metric called the *average pixel position difference (APPD)*. These tools can then be used to train a simple CNN, which we show is able to predict the APPD values from images with no additional data necessary. The performance of the network was evaluated on different real-world datasets and cameras. Provided the camera's true intrinsic parameters remained close, the network was able to generalize well to different cameras and environments that it had not seen before. Such a network can then be deployed on a real robotic platform, running at a very low frequency, to determine if a more expensive recalibration procedure needs to be executed.

ACKNOWLEDGMENTS

The authors would like to thank Jen Jen Chung, Lionel Ott, Juan Nieto and Davide Scaramuzza for their feedback and valuable insights.

BIBLIOGRAPHY

[1] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving," *IEEE Transactions on Intelligent Vehicles (T-IV)*, vol. 2, no. 3, pp. 194–220, 2017.

[2] K. Blomqvist*, M. Breyer*, A. Cramariuc*, J. Förster*, M. Grinvald*, F. Tschopp*, J. J. Chung, L. Ott, J. Nieto, and R. Siegwart, "Go Fetch: Mobile Manipulation in Unstructured Environments," in *IEEE International Conference on Robotics and Automation (ICRA) Workshop on Perception, Action, Learning*, 2020.

[3] J. L. Jones, "Robots at the tipping point: the road to iRobot Roomba," *IEEE Robotics and Automation Magazine (RAM)*, vol. 13, no. 1, pp. 76–78, 2006.

[4] D. Ungureanu, F. Bogo, S. Galliani, P. Sama, X. Duan, C. Meekhof, J. Stühmer, T. J. Cashman, B. Tekin, J. L. Schönberger *et al.*, "HoloLens 2 Research Mode as a Tool for Computer Vision Research," *arXiv preprint arXiv:2008.11239*, 2020.

[5] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 10, pp. 1053–1072, 2017.

[6] R. Dubé*, A. Cramariuc*, D. Dugas, H. Sommer, M. Dymczyk, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: Segment-based mapping and localization using data-driven descriptors," *The International Journal of Robotics Research (IJRR)*, vol. 39, no. 2-3, pp. 339–355, 2020.

[7] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[8] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: An Open-Source Library for Real-Time Metric-Semantic Localization and Map-

ping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1689–1696.

[9] Y. Tian, Y. Chang, F. Herrera Arias, C. Nieto-Granda, J. How, and L. Carlone, "Kimera-Multi: Robust, Distributed, Dense Metric-Semantic SLAM for Multi-Robot Systems," *IEEE Transactions on Robotics (T-RO)*, pp. 1–17, 2022.

[10] Y. Chang, K. Ebadi, C. E. Denniston, M. F. Ginting, A. Rosinol, A. Reinke, M. Palieri, J. Shi, A. Chatterjee, B. Morrell, A.-a. Agha-mohammadi, and L. Carlone, "LAMP 2.0: A Robust Multi-Robot SLAM System for Operation in Challenging Large-Scale Underground Environments," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 4, pp. 9175–9182, 2022.

[11] M. Karrer, P. Schmuck, and M. Chli, "CVI-SLAM—collaborative visual-inertial SLAM," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 2762–2769, 2018.

[12] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," *IEEE Transactions on Robotics (T-RO)*, vol. 37, no. 6, pp. 1874–1890, 2021.

[13] P. Schmuck, T. Ziegler, M. Karrer, J. Perraudin, and M. Chli, "COVINS: Visual-Inertial SLAM for Centralized Collaboration," in *ISMAR-Adjunct*, 2021, pp. 171–176.

[14] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame, "DOOR-SLAM: Distributed, online, and outlier resilient SLAM for robotic teams," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 1656–1663, 2020.

[15] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "maplab: An Open Framework for Research in Visual-inertial Mapping and Localization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 1418–1425, 2018.

[16] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *The International Journal of Robotics Research (IJRR)*, vol. 34, no. 3, pp. 314–334, 2015.

[17] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Robotics: Science and Systems (RSS)*, 2014.

[18] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast Direct LiDAR-Inertial Odometry," *IEEE Transactions on Robotics (T-RO)*, vol. 38, no. 4, pp. 2053–2073, 2022.

[19] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research (IJRR)*, vol. 32, no. 11, pp. 1231–1237, 2013.

[20] M. Helmberger, K. Morin, B. Berner, N. Kumar, G. Cioffi, and D. Scaramuzza, "The Hilti SLAM Challenge Dataset," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 3, pp. 7518–7525, 2022.

[21] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research (IJRR)*, vol. 35, no. 10, pp. 1157–1163, 2016.

[22] D. Detone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-supervised interest point detection and description," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018, pp. 337–349.

[23] Z. Luo, L. Zhou, X. Bai, H. Chen, J. Zhang, Y. Yao, S. Li, T. Fang, and L. Quan, "ASLFeat: Learning Local Features of Accurate Shape and Localization," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6589–6598.

[24] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-Net: A Trainable CNN for Joint Description and Detection of Local Features," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8092–8101.

[25] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning Feature Matching with Graph Neural Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4938–4947.

[26] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "LoFTR: Detector-Free Local Feature Matching With Transformers," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 8922–8931.

[27] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[28] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *The International Journal of Robotics Research (IJRR)*, vol. 31, no. 2, pp. 216–235, 2012.

[29] G. Grisetti, R. Kümmerle, H. Strasdat, and K. Konolige, "g2o: A general framework for (hyper) graph optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 9–13.

[30] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver 2.1," https://github.com/ceres-solver/ceres-solver, 2022.

[31] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1366–1373.

[32] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto, "Voxgraph: Globally Consistent, Volumetric Mapping Using Signed Distance Function Submaps," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 1, pp. 227–234, 2019.

[33] A. Cramariuc*, L. Bernreiter*, F. Tschopp*, M. Fehr, V. Reijgwart, J. Nieto, R. Siegwart, and C. Cadena, "maplab 2.0 – A Modular and Multi-Modal Mapping Framework," *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 2, pp. 520–527, 2023.

[34] J. Zhong*, Z. Ye*, A. Cramariuc, F. Tschopp, J. J. Chung, R. Siegwart, and C. Cadena, "CalQNet-Detection of Calibration Quality for Life-Long Stereo Camera Setups," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 1312–1318.

[35] L. Chen*, Y. Ao*, F. Tschopp, A. Cramariuc, M. Breyer, J. J. Chung, R. Siegwart, and C. Cadena, "Learning Trajectories for Visual-Inertial System Calibration via Model-based Heuristic Deep Reinforcement Learning," in *Conference on Robot Learning (CoRL)*, 2020.

[36] Y. Ao*, L. Chen*, F. Tschopp, M. Breyer, R. Siegwart, and A. Cramariuc, "Unified Data Collection for Visual-Inertial Calibration via Deep Reinforcement Learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1646–1652.

[37] C. Xing*, X. Sun*, A. Cramariuc, S. Gull, J. J. Chung, C. Cadena, R. Siegwart, and F. Tschopp, “Descriptellation: Deep Learned Constellation Descriptors for SLAM,” *arXiv preprint arXiv:2203.00567*, 2022.

[38] “Team CERBERUS in the DARPA SubT challenge,” https://www.subt-cerberus.org/, accessed: 04.01.2023.

[39] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, “SegMatch: Segment-based place recognition in 3D point clouds,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5266–5272.

[40] P. Pfreundschuh, H. F. Hendrikx, V. Reijgwart, R. Dubé, R. Siegwart, and A. Cramariuc, “Dynamic Object Aware LiDAR SLAM based on Automatic Generation of Training Data,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 641–11 647.

[41] G. Kim, S. Choi, and A. Kim, “Scan Context++: Structural Place Recognition Robust to Rotation and Lateral Variations in Urban Environments,” *IEEE Transactions on Robotics (T-RO)*, 2021.

[42] G. Tinchev, A. Penate-Sanchez, and M. Fallon, “Learning to See the Wood for the Trees: Deep Laser Localization in Urban and Natural Environments on a CPU,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 2, pp. 1327–1334, 2019.

[43] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, “Suma++: Efficient lidar-based semantic slam,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4530–4537.

[44] X. Kong, X. Yang, G. Zhai, X. Zhao, X. Zeng, M. Wang, Y. Liu, W. Li, and F. Wen, “Semantic Graph Based Place Recognition for 3D Point Clouds,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 8216–8223.

[45] A. Cramariuc*, A. Petrov*, R. Suri, M. Mittal, R. Siegwart, and C. Cadena, “Learning Camera Miscalibration Detection,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4997–5003.

[46] A. Cramariuc*, F. Tschopp*, N. Alatur, S. Benz, T. Falck, M. Brühlmeier, B. Hahn, J. Nieto, and R. Siegwart, “SemSegMap – 3D Segment-based Semantic Localization,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1183–1190.

[47] A. Bühler, A. Gaidon, A. Cramariuc, R. Ambrus, G. Rosman, and W. Burgard, "Driving Through Ghosts: Behavioral Cloning with False Positives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5431–5437.

[48] S. Lionar*, L. Schmid*, C. Cadena, R. Siegwart, and A. Cramariuc, "NeuralBlox: Real-Time Neural Representation Fusion for Robust Volumetric Mapping," in *International Conference on 3D Vision (3DV)*, 2021, pp. 1279–1289.

[49] F. Tschopp*, C. von Einem*, A. Cramariuc*, D. Hug, A. W. Palmer, R. Siegwart, M. Chli, and J. Nieto, "Hough^{2}Map – Iterative Event-Based Hough Transform for High-Speed Railway Mapping," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 2745–2752, 2021.

[50] M. Dymczyk, S. Lynen, T. Cieslewski, M. Bosse, R. Siegwart, and P. Furgale, "The gist of maps - Summarizing experience for lifelong localization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2767–2773.

[51] M. Dymczyk, S. Lynen, M. Bosse, and R. Siegwart, "Keep it brief: Scalable creation of compressed localization maps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 2536–2542.

[52] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From Coarse to Fine: Robust Hierarchical Localization at Large Scale," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 716–12 725.

[53] A. Loquercio, M. Segu, and D. Scaramuzza, "A General Framework for Uncertainty Estimation in Deep Learning," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 3153–3160, 2020.

[54] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 740–755.

[55] B. Li, D. Zou, D. Sartori, L. Pei, and W. Yu, "TextSLAM: Visual SLAM with Planar Text Features," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2102–2108.

[56] N. Zimmerman, L. Wiesmann, T. Guadagnino, T. Läbe, J. Behley, and C. Stachniss, "Robust Onboard Localization in Changing Environments Exploiting Text Spotting," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 917–924.

[57] L. Bernreiter, S. Khattak, L. Ott, R. Siegwart, M. Hutter, and C. Cadena, "Collaborative Robot Mapping using Spectral Graph Analysis," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[58] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics (T-RO)*, vol. 32, no. 6, pp. 1309–1332, 2016.

[59] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics (T-RO)*, vol. 33, no. 5, pp. 1255–1262, 2017.

[60] M. Labbé and F. Michaud, "Multi-Session Visual SLAM for Illumination-Invariant Re-Localization in Indoor Environments," *Frontiers in Robotics and AI*, p. 115, 2022.

[61] A. Gawel, C. Del Don, R. Siegwart, J. Nieto, and C. Cadena, "X-View: Graph-Based Semantic Multi-View Localization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 1687–1694, 2017.

[62] F. Taubner, F. Tschopp, T. Novkovic, R. Siegwart, and F. Furrer, "LCD - Line Clustering and Description for Place Recognition," in *International Conference on 3D Vision (3DV)*, 2020, pp. 908–917.

[63] L. Bernreiter, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "Multiple Hypothesis Semantic Mapping for Robust Data Association," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 3255–3262, 2019.

[64] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 1879–1884.

[65] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *International Conference on Computer Vision (ICCV)*, 2011, pp. 2564–2571.

[66] S. Leutenegger, M. M. Chli, and R. Y. Siegwart, "Binary Robust Invariant Scalable Keypoints," in *International Conference on Computer Vision (ICCV)*, 2011, pp. 2548–2555.

[67] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast Retina Keypoint," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 510–517.

[68] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *International Joint Conference on Artificial Intelligence*, vol. 2, 1981, pp. 674–679.

[69] S. Lynen, T. Sattler, M. Bosse, J. Hesch, M. Pollefeys, and R. Siegwart, "Get Out of My Lab: Large-scale, Real-Time Visual-Inertial Localization," in *Robotics: Science and Systems (RSS)*, 2015, p. 18.

[70] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration." *VISAPP*, vol. 1, pp. 331–340, 2009.

[71] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to MAV navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 3923–3929.

[72] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," *IEEE International Conference on Rehabilitation Robotics (ICORR)*, 2011.

[73] R. G. Valenti, I. Dryanovski, and J. Xiao, "Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs," *Sensors*, vol. 15, no. 8, pp. 19 302–19 330, 2015.

[74] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, and K. Alexis, "CERBERUS in the DARPA Subterranean Challenge," *Science Robotics*, vol. 7, no. 66, 2022.

[75] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets Open-source library and experimental protocol," *Autonomous Robots*, vol. 34, no. 3, 2014.

[76] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," *Robotics: Science and Systems (RSS)*, vol. 2, no. 4, p. 435, 2009.

[77] D. G. Lowe, "Object recognition from local scale-invariant features," in *International Conference on Computer Vision (ICCV)*, vol. 2, 1999, pp. 1150–1157.

[78] T. Shan, B. Englot, C. Ratti, and D. Rus, "LVI-SAM: Tightly-coupled Lidar-Visual-Inertial Odometry via Smoothing and Mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5692–5698.

[79] D. Streiff, L. Bernreiter, F. Tschopp, M. Fehr, and R. Siegwart, "3D3L: Deep Learned 3D Keypoint Detection and Description for LiDARs," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 13 064–13 070.

[80] T. Mertens, J. Kautz, and F. Van Reeth, "Exposure fusion," in *Pacific Graphics*, 2007, pp. 382–390.

[81] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[82] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5297–5307, 2016.

[83] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," in *IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3645–3649.

[84] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[85] P. S. Manoj, L. Bingbing, Y. Rui, and W. Lin, "A Closed-form Estimate of 3D ICP Covariance," in *IAPR International Conference on Machine Vision Applications (MVA)*, 2015, pp. 526–529.

[86] F. Tschopp, M. Riner, M. Fehr, L. Bernreiter, F. Furrer, T. Novkovic, A. Pfrunder, C. Cadena, R. Siegwart, and J. Nieto, "VersaVIS—An Open Versatile Multi-Camera Visual-Inertial Sensor Suite," *Sensors*, vol. 20, no. 5, p. 1439, 2020.

[87] S. Lowry, N. Sunderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, “Visual Place Recognition: A Survey,” *IEEE Transactions on Robotics (T-RO)*, vol. 32, no. 1, pp. 1–19, 2016.

[88] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2D LIDAR SLAM,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.

[89] R. B. Rusu, N. Blodow, and M. Beetz, “Fast Point Feature Histograms (FPFH) for 3D registration,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 3212–3217.

[90] S. Salti, F. Tombari, and L. Di Stefano, “SHOT: Unique signatures of histograms for surface and texture description,” *Computer Vision and Image Understanding*, vol. 125, pp. 251–264, 2014.

[91] H. Yin, L. Tang, X. Ding, Y. Wang, and R. Xiong, “LocNet: Global Localization in 3D Point Clouds for Mobile Vehicles,” in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 728–733.

[92] R. Dubé, M. G. Gollub, H. Sommer, I. Gilitschenski, R. Siegwart, C. Cadena, and J. Nieto, “Incremental Segment-Based Localization in 3D Point Clouds,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 1832–1839, 2018.

[93] G. Tinchev, S. Nobili, and M. Fallon, “Seeing the Wood for the Trees: Reliable Localization in Urban and Natural Environments,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 8239–8246.

[94] R. Dubé*, A. Cramariuc*, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, “SegMap: 3D Segment Mapping using Data-Driven Descriptors,” in *Robotics: Science and Systems (RSS)*, 2018.

[95] M. Bosse and R. Zlot, “Place Recognition using Keypoint Voting in Large 3D Lidar Datasets,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 2677–2684.

[96] A. Gawel, T. Cieslewski, R. Dubé, M. Bosse, R. Siegwart, and J. Nieto, “Structure-based vision-laser matching,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 182–188.

[97] Y. Zhuang, N. Jiang, H. Hu, and F. Yan, "3-D-Laser-Based Scene Measurement and Place Recognition for Mobile Robots in Dynamic Indoor Environments," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 2, pp. 438–450, 2013.

[98] B. Steder, G. Grisetti, and W. Burgard, "Robust place recognition for 3D range data based on point features," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 1400–1405.

[99] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard, "Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 1249–1255.

[100] T. Röhling, J. Mack, and D. Schulz, "A fast histogram-based similarity measure for detecting loop closures in 3-D LIDAR data," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 736–741.

[101] K. Granström, T. B. Schön, J. I. Nieto, and F. T. Ramos, "Learning to close loops from range data," *The International Journal of Robotics Research (IJRR)*, vol. 30, no. 14, pp. 1728–1754, 2011.

[102] M. Magnusson, H. Andreasson, A. Nüchter, and A. J. Lilienthal, "Automatic appearance-based loop detection from three-dimensional laser data using the normal distributions transform," *Journal of Field Robotics*, vol. 26, no. 11-12, pp. 892–914, 2009.

[103] K. Cop, P. Borges, and R. Dubé, "DELIGHT: An Efficient Descriptor for Global Localisation using LiDAR Intensities," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3653–3660.

[104] E. Fernández-Moral, W. Mayol-Cuevas, V. Arevalo, and J. Gonzalez-Jimenez, "Fast place recognition with plane-based maps," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 2719–2724.

[105] E. Fernández-Moral, P. Rives, V. Arévalo, and J. González-Jiménez, "Scene structure registration for localization and mapping," *Robotics and Autonomous Systems*, vol. 75, pp. 649–660, 2016.

[106] R. Finman, L. Paull, and J. J. Leonard, "Toward object-based place recognition in dense rgb-d maps," in *IEEE International Conference on*

Robotics and Automation (ICRA) Workshop on Visual Place Recognition in Changing Environments, 2015.

[107] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, "Probabilistic data association for semantic slam," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1722–1729.

[108] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "magenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012, pp. 1097–1105.

[109] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1355–1361.

[110] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 371–382.

[111] G. Riegler, A. Osman Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3577–3586.

[112] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," in *Robotics: Science and Systems (RSS)*, 2016.

[113] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D shapenets: A deep representation for volumetric shapes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.

[114] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3D pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3109–3118.

[115] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 652–660.

[116] Y. Fang, J. Xie, G. Dai, M. Wang, F. Zhu, T. Xu, and E. Wong, "3D Deep Shape Descriptor," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2319–2328.

[117] L. P. Tchapmi, C. B. Choy, I. Armeni, J. Gwak, and S. Savarese, "SEG-Cloud: Semantic Segmentation of 3D Point Clouds," in *International Conference on 3D Vision (3DV)*, 2017.

[118] B. Graham, M. Engelcke, and L. van der Maaten, "3D Semantic Segmentation With Submanifold Sparse Convolutional Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 9224–9232.

[119] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1887–1893.

[120] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016, pp. 82–90.

[121] A. Dewan, T. Caselitz, and W. Burgard, "Learning a Local Feature Descriptor for 3D LiDAR Scans," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4774–4780.

[122] M. Velas, M. Spanel, M. Hradis, and A. Herout, "CNN for IMU assisted odometry estimation using velodyne LiDAR," in *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2018, pp. 71–77.

[123] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning Local Geometric Descriptors From RGB-D Reconstructions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1802–1811.

[124] Y. Ye, T. Cieslewski, A. Loquercio, and D. Scaramuzza, "Place Recognition in Semi-Dense Maps: Geometric and Learning-Based Approaches," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[125] G. Elbaz, T. Avraham, and A. Fischer, "3D Point Cloud Registration for Localization using a Deep Neural Network Auto-Encoder," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2472–2481.

[126] A. Brock, T. Lim, J. Ritchie, and N. Weston, “Generative and Discriminative Voxel Modeling with Convolutional Neural Networks,” in *Conference on Neural Information Processing Systems (NeurIPS) Workshop on 3D Deep Learning*, 2016.

[127] V. Guizilini and F. Ramos, “Learning to reconstruct 3D structures for occupancy mapping,” in *Robotics: Science and Systems (RSS)*, 2017.

[128] A. Dai, C. Ruizhongtai Qi, and M. Niessner, “Shape Completion Using 3D-Encoder-Predictor CNNs and Shape Synthesis,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5868–5877.

[129] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, “Shape completion enabled robotic grasping,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2442–2447.

[130] D. Ricao Canelhas, E. Schaffernicht, T. Stoyanov, A. J. Lilienthal, and A. J. Davison, “Compressed Voxel-Based Mapping Using Unsupervised Learning,” *Robotics*, vol. 6, no. 3, p. 15, 2017.

[131] J. L. Schönberger, M. Pollefeys, A. Geiger, and T. Sattler, “Semantic Visual Localization,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6896–6906.

[132] O. M. Parkhi, A. Vedaldi, A. Zisserman *et al.*, “Deep face recognition,” in *Proceedings of the British Machine Vision Conference (BMVC)*, 2015, pp. 1–12.

[133] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, “Signature Verification using a "Siamese" Time Delay Neural Network,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 1994, pp. 737–744.

[134] K. Q. Weinberger, J. Blitzer, and L. K. Saul, “Distance metric learning for large margin nearest neighbor classification,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2006, pp. 1473–1480.

[135] R. Dubé, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, “An online multi-robot SLAM system for 3D lidars,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1004–1011.

[136] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, vol. 9, 2010, pp. 249–256.

[137] K. D. P. and B. J. L., "ADAM: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2015.

[138] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[139] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*, 2015, pp. 448–456.

[140] J. Elseberg, S. Magnenat, R. Siegwart, and A. Nüchter, "Comparison of Nearest-Neighbor-Search Strategies and Implementations for Efficient Shape Registration," *Journal of Software Engineering for Robotics*, vol. 3, no. 1, pp. 2–12, 2012.

[141] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[142] M. Weinmann, B. Jutzi, and C. Mallet, "Semantic 3D scene interpretation: A framework combining optimal neighborhood size selection with relevant features," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 2, no. 3, p. 181, 2014.

[143] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823.

[144] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[145] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[146] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics (T-RO)*, vol. 31, no. 5, 2015.

[147] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *IEEE*

International Conference on Robotics and Automation (ICRA), 2012, pp. 1643–1649.

[148] J. Revaud, P. Weinzaepfel, C. R. de Souza, and M. Humenberger, "R2D2: Repeatable and Reliable Detector and Descriptor," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[149] M. Elhousni and X. Huang, "A Survey on 3D LiDAR Localization for Autonomous Vehicles," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1879–1884.

[150] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, "Large-Scale Image Retrieval With Attentive Deep Local Features," in *International Conference on Computer Vision (ICCV)*, 2017, pp. 3456–3465.

[151] A. Benbihi, S. Arravechia, M. Geist, and C. Pradalier, "Image-Based Place Recognition on Bucolic Environment Across Seasons From Semantic Edge Description," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3032–3038.

[152] H. Hu, Z. Qiao, M. Cheng, Z. Liu, and H. Wang, "DASGIL: Domain Adaptation for Semantic and Geometric-aware Image-based Localization," *IEEE Transactions on Image Processing*, vol. 30, pp. 1342–1353, 2020.

[153] L. Hammarstrand, F. Kahl, W. Maddern, T. Pajdla, M. Pollefeys, T. Sattler, J. Sivic, E. Stenborg, C. Toft, and A. Torii, "Benchmarking Long-term Visual Localization," `https://www.visuallocalization.net/`, Accessed 2020-11-23.

[154] K. Ok, K. Liu, K. Frey, J. P. How, and N. Roy, "Robust Object-based SLAM for High-speed Autonomous Navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 669–675.

[155] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, 1992, pp. 239–256.

[156] L. Bernreiter, L. Ott, J. Nieto, R. Siegwart, and C. Cadena, "PHASER: a Robust and Correspondence-free Global Pointcloud Registration," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 855–862, 2021.

[157] H. M. Le, T.-T. Do, T. Hoang, and N.-M. Cheung, "SDRSAC: Semidefinite-Based Randomized Approach for Robust Point Cloud Registration Without Correspondences," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 124–133.

[158] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss, "OverlapNet: Loop Closing for LiDAR-based SLAM," in *Robotics: Science and Systems (RSS)*, 2020.

[159] H. Yin, Y. Wang, X. Ding, L. Tang, S. Huang, and R. Xiong, "3D LiDAR-Based Global Localization Using Siamese Neural Network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1380–1392, 2019.

[160] G. Kim, B. Park, and A. Kim, "1-Day Learning, 1-Year Localization: Long-Term LiDAR Localization Using Scan Context Image," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 2, pp. 1948–1955, 2019.

[161] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song, "L3-Net: Towards Learning Based LiDAR Localization for Autonomous Driving," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6389–6398.

[162] F. Kallasi, D. L. Rizzini, and S. Caselli, "Fast Keypoint Features From Laser Scanner for Robot Localization and Mapping," *IEEE Robotics and Automation Letters (RA-L)*, vol. 1, no. 1, pp. 176–183, 2016.

[163] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, "The perfect match: 3d point cloud matching with smoothed densities," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5545–5554.

[164] L. Schaupp, M. Bürki, R. Dubé, R. Siegwart, and C. Cadena, "OREOS: Oriented Recognition of 3D Point Clouds in Outdoor Scenarios," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 3255–3261.

[165] A. Zaganidis, A. Zerntev, T. Duckett, and G. Cielniak, "Semantically Assisted Loop Closure in SLAM Using NDT Histograms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4562–4568.

[166] B. Douillard, A. Quadros, P. Morton, J. P. Underwood, M. De Deuge, S. Hugosson, M. Hallström, and T. Bailey, "Scan segments matching

for pairwise 3D alignment," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3033–3040.

[167] J. Nieto, T. Bailey, and E. Nebot, "Scan-SLAM: Combining EKF-SLAM and scan correlation," in *Field and Service Robotics*, vol. 25, 2006, pp. 167–178.

[168] L. Sun, Z. Yan, A. Zaganidis, C. Zhao, and T. Duckett, "Recurrent-OctoMap: Learning State-Based Map Refinement for Long-Term Semantic Mapping With 3-D-Lidar Data," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3749–3756, 2018.

[169] A. Zaganidis, L. Sun, T. Duckett, and G. Cielniak, "Integrating Deep Semantic Segmentation Into 3-D Point Cloud Registration," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 2942–2949, 2018.

[170] S. A. Parkison, L. Gan, M. G. Jadidi, and R. Eustice, "Semantic iterative closest point through expectation-maximization," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

[171] L. Bernreiter, L. Ott, J. Nieto, R. Siegwart, and C. Cadena, "Spherical Multi-Modal Place Recognition for Heterogeneous Sensor Systems," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1743–1750.

[172] S. Ratz, M. Dymczyk, R. Siegwart, and R. Dubé, "OneShot Global Localization: Instant LiDAR-Visual Pose Estimation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 5415–5421.

[173] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[174] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004, pp. 2149–2154.

[175] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Conference on Robot Learning (CoRL)*, 2017.

[176] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta, E. Agafonov, T. H. Kim,

E. Sterner, K. Ushiroda, M. Reyes, D. Zelenkovsky, and S. Kim, “LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving,” in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[177] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles,” in *Field and Service Robotics*, 2018, pp. 621–635.

[178] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, “University of Michigan North Campus long-term vision and lidar dataset,” *The International Journal of Robotics Research (IJRR)*, vol. 35, no. 9, pp. 1023–1035, 2016.

[179] A. Tao, K. Sapra, and B. Catanzaro, “Hierarchical Multi-Scale Attention for Semantic Segmentation,” *arXiv preprint arXiv:2005.10821*, 2020.

[180] Anu, W. Nguatem, and Jan, “Computing a global descriptor with local descriptors,” *Point Cloud Library (PCL) Users mailing list - Computing a global descriptor with local descriptors*, 2015.

[181] Z. Roth, B. Mooring, and B. Ravani, “An Overview of Robot Calibration,” *IEEE Journal of Robotics and Automation*, vol. 3, pp. 377 – 385, 1987.

[182] J. A. Preiss, K. Hausman, G. S. Sukhatme, and S. Weiss, “Simultaneous self-calibration and navigation using trajectory optimization,” *The International Journal of Robotics Research (IJRR)*, vol. 37, no. 13-14, pp. 1573–1594, 2018.

[183] T. Schneider, M. Li, M. Burri, J. I. Nieto, R. Siegwart, and I. Gilitschenski, “Visual-inertial self-calibration on informative motion segments,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6487–6494, 2017.

[184] M. L. Visinsky, J. R. Cavallaro, and I. D. Walker, “Robot fault detection and fault tolerance: A survey,” *Reliability Engineering and System Safety*, vol. 46, no. 2, pp. 139–158, 1994.

[185] K. Ni, N. Ramanathan, M. N. H. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava, “Sensor Network Data Fault Types,” *ACM Transactions on Sensor Networks*, vol. 5, no. 3, p. 25, 2009.

[186] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[187] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A Toolbox for Easily Calibrating Omnidirectional Cameras," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 5695–5701.

[188] B. Atcheson, F. Heide, and W. Heidrich, "CALTag: High Precision Fiducial Markers for Camera Calibration," in *International Workshop on Vision, Modeling and Visualization*, vol. 10, 2010, pp. 41–48.

[189] S. J. Maybank and O. D. Faugeras, "A theory of self-calibration of a moving camera," *International Journal of Computer Vision*, vol. 8, no. 2, pp. 123–151, 1992.

[190] M. Armstrong, A. Zisserman, and R. Hartley, "Self-calibration from image triplets," in *European Conference on Computer Vision (ECCV)*, 1996, pp. 1–16.

[191] L. Agapito, E. Hayman, and I. Reid, "Self-calibration of rotating and zooming cameras," *International Journal of Computer Vision*, vol. 45, no. 2, pp. 107–127, 2001.

[192] P. Sturm, "Critical motion sequences for monocular self-calibration and uncalibrated Euclidean reconstruction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1997, pp. 1100–1105.

[193] H. Wildenauer and A. Hanbury, "Robust camera self-calibration from monocular images of Manhattan worlds," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 2831–2838.

[194] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[195] S. Workman, C. Greenwell, M. Zhai, R. Baltenberger, and N. Jacobs, "Deepfocal: A method for direct focal length estimation," in *IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 1369–1373.

[196] K. Wilson and N. Snavely, "Robust Global Translations with 1DSfM," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 61–75.

[197] M. Lopez, R. Mari, P. Gargallo, Y. Kuang, J. Gonzalez-Jimenez, and G. Haro, "Deep Single Image Camera Calibration With Radial Distortion," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 11 817–11 825.

[198] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba, "Recognizing Scene Viewpoint using Panoramic Place Representation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 2695–2702.

[199] X. Yin, X. Wang, J. Yu, M. Zhang, P. Fua, and D. Tao, "FishEyeRecNet: A Multi-Context Collaborative Deep Network for Fisheye Image Rectification," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 469–484.

[200] J. P. Mendoza, M. Veloso, and R. Simmons, "Mobile robot fault detection based on redundant information statistics," in *IROS Workshop on Safety in Human-Robot Coexistence and Interaction*, vol. 945, 2012.

[201] P. Sundvall and P. Jensfelt, "Fault detection for mobile robots using redundant positioning systems," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 3781–3786.

[202] S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey, "Sensor fault detection and identification in a mobile robot," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Innovations in Theory, Practice and Applications*, vol. 3, 1998, pp. 1383–1388.

[203] Y. Lu, E. G. Collins Jr, and M. F. Selekwa, "Parity Relation Based Fault Detection, Isolation and Reconfiguration for Autonomous Ground Vehicle Localization Sensors," Florida State University Tallahassee Department of Mechanical Engineering, Tech. Rep., 2004.

[204] T. Schneider, M. Li, C. Cadena, J. Nieto, and R. Siegwart, "Observability-aware self-calibration of visual and inertial sensors for ego-motion estimation," *IEEE Sensors Journal*, vol. 19, no. 10, pp. 3846–3860, 2019.

[205] J. Heikkila, "Geometric camera calibration using circular control points," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1066–1077, 2000.

[206] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A Multimodal Dataset for Autonomous Driving," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 621–11 631.

[207] L. Oth, P. Furgale, L. Kneip, and R. Siegwart, “Rolling shutter camera calibration,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1360–1367.

CURRICULUM VITAE

Andrei Cramariuc
born August 11, 1993, citizen of Romania and Finland

EDUCATION

Jan 2019 – Feb 2023	*ETH Zurich, Switzerland* PhD at the Autonomous Systems Lab, under the supervision of Prof. Roland Siegwart, on the topics of mapping and learning for robotics.
Sep 2016 – Jun 2019	*ETH Zurich, Switzerland* MsC in Electrical Engineering and Information Technology. Thesis "Learning Visual Object Descriptors for Place Recognition."
Sep 2012 – Jul 2015	*Tampere University of Technology, Finland* BsC in Electrical Engineering.

WORK

Sep 2015 – Mar 2016	*Tampere University of Technology, Finland* Research Assistant and Teaching Assistant for two courses at the Computer Vision Group.

ACHIEVEMENTS

- Supervised over 45 students in almost many projects.
- Over 400 citations on Google Scholar with an h-index of 10.
- A main contributor to an open source mapping framework, with 2000+ stars and 500+ forks (`https://github.com/ethz-asl/maplab/`).
- International Olympiad in Informatics (Bronze 2011, Bronze 2012), Baltic Olympiad in Informatics (Silver 2012), Finnish Nationals in Informatics (Silver 2012).

FIRST AUTHOR PUBLICATIONS

- **A. Cramariuc***, *et al.*, "maplab 2.0 – A Modular and Multi-Modal Mapping Framework", *RA-L*, vol. 8, no. 2, pp. 520–527, 2023.
- **A. Cramariuc***, *et al.*, "SemSegMap – 3D Segment-Based Semantic Localization", *IROS*, pp. 1183–1190, 2021.
- **A. Cramariuc***, *et al.*, "Hough2Map – Iterative Event-based Hough Transform for High-Speed Railway Mapping", *RA-L*, vol. 6, no. 2, pp. 2745–2752, 2021.
- **A. Cramariuc***, *et al.*, "SegMap: Segment-based mapping and localization using data-driven descriptors", *IJRR*, vol. 39, no. 2-3, pp. 339—355, 2020.
- **A. Cramariuc***, *et al.*, "Learning Camera Miscalibration Detection", *ICRA*, pp. 4997–5003, 2020.
- **A. Cramariuc***, *et al.*, "SegMap: 3D Segment Mapping using Data-Driven Descriptors", *RSS*, 2018.
- **A. Cramariuc**, *et al.*, "Clustering benefits in mobile-centric WiFi positioning in multi-floor buildings", *ICL-GNSS*, pp. 1–6, 2016.

OTHER PUBLICATIONS

- Y. Ao*, L. Chen*, *et al.*, **A. Cramariuc**, "Unified Data Collection for Visual-Inertial Calibration via Deep Reinforcement Learning", *ICRA*, 2022.
- P. Pfreundschuh, *et al.*, **A. Cramariuc**, "Dynamic Object Aware LiDAR SLAM based on Automatic Generation of Training Data", *ICRA*, 2021.
- S. Lionar*, L. Schmid*, *et al.*, **A. Cramariuc**, "NeuralBlox: Real-Time Neural Representation Fusion for Robust Volumetric Mapping", *3DV*, 2021.
- J. Zhong*, Z. Ye*, **A. Cramariuc**, *et al.*, "CalQNet - Detection of Calibration Quality for Life-Long Stereo Camera Setups", *IV*, 2021.
- L. Chen*, Y. Ao*, F. Tschopp, **A. Cramariuc**, *et al.*, "Learning Trajectories for Visual-Inertial System Calibration via Model-based Heuristic Deep Reinforcement Learning", *CoRL*, 2020.
- A. Bühler, A. Gaidon, **A. Cramariuc**, *et al.*, "Driving Through Ghosts: Behavioral Cloning with False Positives", *IROS*, 2020.