Towards SLAM-Centric Inspection of Infrastructure

by

Nicholas Charron

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in the departments of
Civil & Environmental Engineering, and Mechanical & Mechatronics Engineering

Waterloo, Ontario, Canada, 2025

© Nicholas Charron 2025

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Shirley Dyke Professor, Dept. of Mechanical Engineering Purdue University
Supervisor(s):	Sriram Narasimhan Adjunct Professor, Dept. Civil and Environmental Engineering University of Waterloo Professor, Dept. Civil and Environmental Engineering & Dept. Mechanical and Aerospace Engineering University of California, Los Angeles
	Steven L. Waslander Professor, Institute for Aerospace Studies University of Toronto
Internal Member:	Scott Walbridge Professor, Dept. of Civil and Environmental Engineering University of Waterloo
Internal-External Member:	William Melek Professor, Dept. of Mechanical And Mechatronics Engineering University of Waterloo

Internal-External Member: John Zelek
Associate Professor, Dept. of Systems Design Engineering
University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

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

Abstract

The inspection and maintenance of civil infrastructure are essential for ensuring public safety, minimizing economic losses, and extending the lifespan of critical assets such as bridges and parking garages. Traditional inspection methods rely heavily on manual visual assessments, which are often subjective, labor-intensive, and inconsistent. These limitations have driven the development of robotic-aided inspection techniques that leverage mobile robotics, sensor fusion, computer vision, and machine learning to enhance inspection efficiency and accuracy.

Despite advancements in robotic-aided inspection, existing works often focus on isolated components of the inspection process—such as improving data collection or automating defect detection—without providing a complete end-to-end solution. Many approaches utilize robotics to capture 2D images for inspection, but these lack spatial context, making it difficult to accurately locate, quantify, and track defects over multiple inspections. Other works extend this by detecting defects within images; however, without a robust 3D representation, defects cannot be precisely geolocated or measured in real-world dimensions, limiting their utility for long-term monitoring. While some studies explore 3D mapping for inspection, the majority rely on image-only Structure-from-Motion, which is known to be unreliable for generating dense and accurate maps, or are restricted to mapping along 2D surfaces, thereby failing to capture the full complexity of infrastructure assets.

This thesis introduces a novel SLAM (Simultaneous Localization and Mapping)-centric framework for robotic infrastructure inspection, addressing these limitations by integrating lidar, cameras, and inertial measurement units (IMUs) into a mobile robotic platform. This system enables precise and repeatable localization, 3D mapping, and automated inspection of infrastructure assets.

Three key challenges that hinder the development of a practical SLAM-centric inspection system are identified and addressed in this work. The first challenge pertains to the design and implementation of SLAM-centric robotic systems. This thesis demonstrates how sensor selection and configuration can be optimized to simultaneously support both high-accuracy SLAM and high-quality inspection data collection. Additionally, it establishes a robotic platform-agnostic design, allowing for flexibility across different infrastructure inspection applications. The second challenge involves precise and reliable calibration of camera-lidar systems, particularly when sensors have non-overlapping fields of view as is the case with the proposed inspection systems. To address this, a novel target-based extrinsic calibration technique is developed, leveraging a motion capture system to achieve high-precision calibration across both sensing modalities. This ensures accurate sensor

fusion, yielding geometrically consistent inspection outputs. The third challenge is the development of a complete end-to-end inspection methodology. This research implements state-of-the-art online camera-lidar-IMU SLAM, with an added offline refinement process and a decoupled mapping framework. This approach enables the generation of high-quality 3D maps that are specifically tailored for infrastructure inspection by prioritizing accuracy, density, and low noise in the map. Machine learning-based defect detection is then integrated into the pipeline, coupled with a novel 3D map labeling method that transfers visual and defect information onto the 3D inspection map. Finally, an automated defect quantification and tracking system is introduced, allowing for defects to be monitored across multiple inspection cycles—completing the full end-to-end inspection workflow.

The proposed SLAM-centric inspection system is validated through extensive real-world experiments on infrastructure assets, including a bridge and a parking garage. Results demonstrate that the system generates highly accurate, repeatable, and metrically consistent inspection data, significantly improving upon traditional manual inspection methods. By enabling automated defect detection, precise localization, and long-term defect tracking within a robust 3D mapping framework, this research represents a paradigm shift in infrastructure assessment—transitioning from qualitative visual inspections to scalable, data-driven, and quantitative condition monitoring.

Ultimately, this thesis advances the field of robotic infrastructure inspection by presenting a comprehensive SLAM-centric framework that integrates state-of-the-art sensing, calibration, and mapping techniques. The findings have broad implications for the future of automated infrastructure management, providing a foundation for intelligent inspection systems that can enhance the efficiency, reliability, and safety of civil infrastructure maintenance worldwide.

Acknowledgements

I would like to start by extending my sincere gratitude to the National Research Council of Canada (NSERC) for their generous funding, which supported the majority of my research endeavors throughout my PhD. The Research Tools and Instruments (RTI) grant program was instrumental in providing me with the ability to purchase all the equipment I needed to build most of the robotic systems used over the years of research. Furthermore, the Canada Graduate Scholarships program allowed me to focus my time on my research without the need to get funding to cover my living expenses during the years I spent in graduate school.

I am also deeply indebted to my PhD supervisors, Dr. Sriram Narasimhan and Dr. Steven Waslander, for their invaluable guidance, mentorship, and unwavering support. Thank you both for trusting me to take on such complex research in a field where I had little knowledge coming into the program. You supported me through many years of learning the robotics skills I needed to make an impact in Civil Engineering that few others in the world had the expertise to do. You also encouraged my internship endeavors during my PhD, which not only helped me learn valuable skills to put towards my research but also set me up for a very successful career post-graduation.

I would also like to express my heartfelt appreciation to my laboratory colleagues, particularly Jake McLaughlin, Stephen Phillips, Evan McLaughlin, and Alexander Thoms, with whom I had the pleasure of collaborating closely in the SDIC lab. Without all your help, I would not have been able to build such complex robotic systems, collect data in the field, and write robust software that helped power much of the research I get to present in this thesis. Our collaborations excelled my research and learning in a way that I cannot even begin to quantify. Furthermore, I would like to thank my laboratory colleagues from the WAVE lab, who all helped teach me many basic robotics skills in the first few years of my studies, notably Stan Brown, Chris Choi, Jason Rebello, and Arun Das. Our early research on UAVs and autonomous driving with the Autonomoose project taught me a lot about how to best approach field testing and building complex robotic systems.

Lastly, I wish to thank my family and friends for their unwavering support, love, and encouragement throughout this journey. Their patience, understanding, and motivation have been a constant source of strength. Thank you for supporting my drive and giving me space to achieve my dreams. And most of all, thank you for eventually stopping to ask the question: "when will you graduate?" :)

Table of Contents

List of Figures **xi**

List of Tables **xiii**

List of Abbreviations **xiv**

1 Introduction **1**

1.1 Motivation 2

1.2 Research Landscape 6

1.3 Contributions 9

1.4 Thesis Organization 10

2 Background **11**

2.1 Three-Dimensional Geometry 11

2.2 Localization and Mapping 13

2.2.1 Recursive EKF Localization 15

2.2.2 Batch Localization 17

2.2.3 Simultaneous Localization and Mapping 21

2.3 Scan Registration 22

2.3.1 Iterative Closest Point and Variants 23

2.3.2 LOAM Matching 24

2.4 Camera Modeling 27
2.4.1 Pinhole 29
2.4.2 Kannala Brandt 30
2.4.3 Double Sphere 31
2.5 Sensor Calibration 32
2.5.1 Camera Intrinsic Calibration 33
2.5.2 Lidar Intrinsic Calibration 33
2.5.3 IMU Intrinsic Calibration 35

3 System Design **36**
3.1 Related Works 37
3.2 Mobility 41
3.3 Sensor Configuration 43
3.3.1 Base Module 44
3.3.2 Inspection Specific Modules 49
3.4 Synchronization 52
3.5 Conclusion 56

4 Calibration **57**
4.1 Related Works 60
4.2 Methodology 62
4.2.1 Problem Formulation and Notation 62
4.2.2 Mathematical Framework 63
4.2.3 Measurement Extractors 65
4.2.4 Correspondence Estimation and Optimization 66
4.3 Results 67
4.3.1 Simulation Data 67
4.3.2 Experimental Data 70
4.4 Conclusions 74

5 Inspection Methodology **76**
5.1 Related Works . . . 77
5.2 Datasets . . . 80
5.2.1 Park Street Bridge . . . 80
5.2.2 Duke Street Parking Garage . . . 81
5.3 Inspection Methodology & Results . . . 82
5.3.1 Trajectory & Map Generation . . . 84
5.3.2 Defect Detection . . . 96
5.3.3 Map Labeling and Defect Quantification . . . 101
5.4 Conclusions . . . 113

6 Conclusions **114**
6.1 Key Conclusions . . . 114
6.2 Significant Contributions . . . 115
6.3 Sources of Uncertainty . . . 116
6.4 Limitations and Directions for Future Study . . . 117
6.5 Implementation in Practice . . . 118
6.6 Implications for Infrastructure Management . . . 119

References **120**

APPENDICES **136**

A List of Publications **137**
A.1 Publications in Referred Journals . . . 137
A.2 Publications in Conference Proceedings . . . 138
A.3 Publications Under Review . . . 139

B Map Quality Analysis **140**
B.1 Existing Quality Analysis Approaches . . . 140
B.2 Proposed Distance To Planes Map Quality Metric . . . 141

List of Figures

2.1 Background on 3D geometry 12
2.2 Factor graph example 19
2.3 Batch vs. Sliding Window Factor Graphs 21
2.4 Pinhole camera model 28
2.5 Camera models 30
2.6 Multi-beam mechanical lidars 34

3.1 Inspection platforms 38
3.2 Base module for inspection systems 43
3.3 Example inspection systems 43
3.4 Inspection-specific sensors 51
3.5 Example inspection images 52
3.6 Vertical vs Horizonatl lidar maps 53
3.7 Example delamination images from infrared camera 54
3.8 Time synchronization diagram 55

4.1 Calibration problem coordinate frames 63
4.2 Example correspondence estimation - diamond target 67
4.3 Calibration results - simulation 70
4.4 Sample calibration results - cylinder target 71
4.5 Experimental setup 72

4.6 Sample lidar results 73
4.7 Example experimental camera results 75

5.1 Park Street bridge dataset images 81
5.2 Duke Street parking garage dataset images 82
5.3 Inspection workflow 83
5.4 Mapping workflow 85
5.5 Online SLAM flowchart 86
5.6 Online SLAM map results 89
5.7 Offline pose refinement workflow 91
5.8 Pose refinement results 93
5.9 Trajectory generation 94
5.10 Map builder workflow 95
5.11 Lidar map results from map builder 97
5.12 Defect detection workflow 98
5.13 Example image extraction 99
5.14 Bridge example defect labels - front camera 101
5.15 Bridge example defect labels - rear camera 102
5.16 Garage example defect labels 103
5.17 Map labeling workflow 104
5.18 Bridge map labeling results - front camera 107
5.19 Bridge map labeling results - rear camera 108
5.20 Garage map labeling results 109
5.21 Bridge defect segmentations 110
5.22 All bridge defects on map 111
5.23 Garage defect segmentations 112

B.1 Example maps for comparing map quality metrics 142
B.2 Garage plane extraction results 142

List of Tables

3.1 Base module sensors 46
3.2 Hardware selection 47
3.3 Example ground sampling distance and defect sizes 49

4.1 Simulated lidar calibration errors 69
4.2 Simulated camera calibration errors 69
4.3 Experiment calibration errors 74

5.1 Bridge defect quantification summary 110
5.2 Garage defect quantification summary 111

B.1 Map quality metric comparisons 144

List of Abbreviations

ARVR Augmented Reality and Virtual Reality 45

ASCE American Society of Civil Engineers 3

BIM Building Information Modeling 5

CAD Computer Aided Design 36

DNN Deep Neural Network 78

DOF Degree(s) of Freedom 13, 24, 32, 39, 46, 65, 68, 78

DSLR Digital Single Lens Reflex 40, 42

E2E End-to-End 2, 4–6, 9, 10, 56, 77, 79, 84, 112–115

EKF Extended Kalman Filter 15, 16, 18, 22, 38, 78

FG Factor Graph 18–21

FOV Field of View 2, 8, 9, 27, 29–31, 42, 43, 46–48, 50, 52, 58, 60, 61, 66, 71, 73, 74, 79, 97, 101, 107

GDP Gross Domestic Product 2

GICP Generalized Iterative Closest Point 24

GNSS Global Navigation Satellite System 7, 15, 18–21, 37, 38, 40

GSD Ground Sampling Distance 48, 49

ICP Iterative Closest Point 23, 24, 91

IMU Inertial Measurement Unit 5, 14, 19, 24, 26, 32, 35, 37, 38, 40, 42, 44–47, 49, 54, 55, 78, 84–90, 92, 95, 117

IR Infrared 38, 39

LIO Lidar Inertial Odometry 45, 49

LOAM Lidar Odometry and Mapping 24, 25

LVI Lidar Visual Inertial 86

LVIO Lidar Visual Inertial Odometry 45, 49

MCS Motion Capture System 9, 58, 59, 62–69, 71, 72, 74

ML Machine Learning 76, 100

NNS Nearest Neighbor Search 66

RGB Red Green Blue 8, 38, 39, 42, 44, 45, 50, 60, 78, 102, 104, 106, 107, 113

RGBD Red Green Blue Depth 8, 22, 39, 46, 77, 78

ROS Robot Operating System 55, 67, 68, 72, 83, 86, 87, 92

SfM Structure from Motion 7, 23, 24, 27, 36

SLAM Simultaneous Localization and Mapping 1, 2, 5, 6, 8–10, 13, 18, 20–24, 27, 35–40, 42, 44–53, 56–58, 61, 74–80, 83–86, 88–92, 94, 95, 102, 112–119

SOTA State-of-the-Art 5, 9, 45, 47, 53, 56, 59, 73, 74, 76–78, 84, 113

TLS Terrestrial Laser Scanner 7, 8

UAV Uncrewed Aerial Vehicle 6, 8, 15, 24, 38–42, 47, 78, 79

UGV Uncrewed Ground Vehicle 6, 8, 37–39, 41–44, 50, 51, 56, 80

USV Uncrewed Surface Vessel 6, 39, 41–44, 47, 56

UUV Uncrewed Underwater Vessel 6

VIO Visual Inertial Odometry 38, 45, 49

Chapter 1

Introduction

In the context of civil engineering, one of the major financial burdens to society is that of infrastructure construction and maintenance. Naturally, this component of the industry has begun to adopt new technologies such as machine learning, computer vision, and robotics technologies in an attempt to greatly improve efficiency and save society significant cost burdens while also improving safety. To properly maintain infrastructure, regular and high-quality inspections are a key requirement enabling effective planning. As a result, academia and industry have been focusing a great deal of effort on technological advancements targeting inspection tasks.

The overarching goal of this thesis is to study the potential for using mobile robotics for visual inspection of infrastructure. Elaborating on this goal, the term mobile robotics in this work encompasses all technologies that enable the functionality of a mobile robotic system, including computer vision, image processing, machine learning, state estimation, control, and others. The focus will be on the applicability of these technologies for automating visual inspection of infrastructure to improve efficiency, accuracy, and safety compared to standard practice. The term infrastructure refers to any physical structure or facility required for the operation of society; however, the majority of the tests will focus on bridges and parking garages, as these public infrastructure components are easy to access and are highly susceptible to deterioration due to their exposure to the outside environment.

More specifically, this thesis proposes a novel approach to this problem space by introducing the concept of Simultaneous Localization and Mapping (SLAM)-centric infrastructure inspection. The proposed SLAM-centric infrastructure inspection approach designs the robotic platform and accompanying algorithms around the use of multi-sensor fusion

for precise robot tracking and 3D map generation. This type of system has the potential to solve many of the problems with existing robotic-aided inspection works, but it also presents many new challenges that need to be addressed. All of these issues are discussed, and novel solutions are presented, demonstrating the viability of such a system End-to-End (E2E).

The main contribution of this thesis occurs at the system level and solves many shortcomings that exist today with robotic infrastructure inspection technologies. In addition, the proposed slam-centric system also surfaces unique new problems and highlights the impact of existing problems related to robot-aided inspection. For example, selecting and integrating sensors to enable precise and robust SLAM while also generating the required data to extract important inspection information is a challenge addressed in Chapter 3. Such systems also often have unique configurations and non-overlapping Field of View (FOV)s[1] that are a challenge to calibrate precisely and reliably, which is addressed in Chapter 4. Lastly, the problem of how to best use SLAM outputs for generating maps tailored to inspection is covered in Chapter 5, as well as the challenges associated with detecting defects and properly combining these with 3D maps to quantify and track defects is another major challenge also addressed in Chapter 5. During each of these chapters, this thesis introduces and validates several new algorithmic contributions aimed at solving these problems.

1.1 Motivation

Government agencies around the world invest large proportions of their spending on constructing new and maintaining existing infrastructure. The cost burden resulting from these investments is rapidly increasing and becoming unsustainable as a result of growing infrastructure needs, as well as the deterioration of existing infrastructure. The McKinsey Global Institute (McKinsey) estimated that $57 trillion in infrastructure investment will be required worldwide between 2013 and 2030 just to keep up with projected Gross Domestic Product (GDP), which is approximately 60% more than the previous 18 years [58]. Shockingly, this number excludes the costs of addressing the significant backlogs and deficiencies in the maintenance of the existing infrastructure. In fact, McKinsey estimates that this $57 trillion would not be sufficient to address these backlogs. Recently, the Biden-Harris administration has passed a historic $400 billion investment in infrastructure maintenance, recognizing the severity of the infrastructure state in the United States.

[1]FOV is the observable area that a sensor can capture, usually depicted by horizontal and vertical angles.

Bridges are of particularly high concern due to the number of bridges approaching or exceeding their design life and the risk associated with their failure. In the last few years, there have been at least five major bridge disasters, including two footbridges in 2017 in Germiston, South Africa, and Prague, Czech Republic, the 2018 fatal collapse of the Morandi Bridge in Genoa, Italy, the 2020 Albiano Magra Bridge in Italy [95], and the 2022 Fern Hollow Bridge in Pittsburgh, Pennsylvania. According to the American Society of Civil Engineers (ASCE) 2021 American Infrastructure Report Card [7], there are 617,000 bridges across the USA, where approximately 42% are at least 50 years old. Of these bridges, 7.5% (46,156) are considered structurally deficient, resulting in a backlog of $125 billion in bridge repair. At the current investment rate in bridge repair, it was estimated that it will take until 2071 to make all current necessary repairs [7]. Similarly, in Canada, 12.4% of bridges (9,661) are rated as being in very poor condition[2] or poor condition[3], with 26.3% in fair condition[4] [25].

Due to the sheer number of aging bridges worldwide and the safety risks associated with their failure, bridges are subjected to stringent inspection requirements through standards. Such standards provide guidance to trained inspectors on how to inspect structures using various techniques ranging from visual inspections to specialized inspections using equipment such as radar or ultrasound. Many jurisdictions (e.g., Ontario) require visual inspections to be performed every 2 years for all bridges with a minimum specified span [98], and more detailed inspections are undertaken as warranted by the condition. As a result of the systematic inspection procedures laid out for bridge inspection, these same inspection practices are often translated to many other types of infrastructure that also benefit from regular inspection.

This thesis generally restricts discussion to visual inspection of concrete structures. Contact-based inspections are also required in some cases during routine inspections. Examples include hammer sounding for delamination detection, electrical resistivity testing, or even core sampling. Furthermore, other structure types have additional visual defects that need to be assessed, including hairline cracks in steel or corrosion. Since concrete structures make up a large portion of physical infrastructure, and visual defect assessment represents an important element of these inspections, this thesis focuses on improving this

[2] *very poor condition*: The asset is unfit for sustained service. It is near or beyond its expected service life and shows widespread signs of advanced deterioration. Some assets may be unusable.

[3] *poor condition*: There is an increasing potential for its condition to affect the service it provides. The asset is approaching the end of its service life, the condition is below the standard, and a large portion of the system exhibits significant deterioration.

[4] *fair condition*: The asset requires attention. The asset shows signs of deterioration, and some elements exhibit deficiencies.

area. In the context of visual inspection of infrastructure, E2E means detecting, quantifying, and tracking changes of defects.

Visual assessment to uncover material defects, performance deficiencies, and maintenance needs for infrastructure generally requires an inspector to gain good visual access to all element locations, including the underside of supporting infrastructure. Where easy access is not possible, specialized equipment such as boom lifts and scissor lifts are used to gain access to all components. In the case of bridge inspection, these alternative access tools often require temporarily closing the bridge for traffic and require specialized training to operate. Regardless of access issues, ensuring consistent quality, repeatability, and objectivity between inspections remains challenging with current visual inspection practices. Missing information, lack of photographic evidence, and variation in condition ratings between inspectors are some key issues that persist with purely visual inspection methods. In [100], condition rating variations were measured as being up to 50% between inspectors, with only 68% of ratings lying within an acceptable deviation of 10% from the mean.

Many visual inspection shortcomings can be overcome using robotics and non-contact sensing technologies. They not only provide a means to gain access to critical and hard-to-reach bridge elements, e.g., the underside of bridge decks but they can also be used to make inspections repeatable and more precise, thus eliminating variations between inspections. By creating 3D maps in the field[5], elements of interest can be automatically located on the structure, which makes inspections of the same area repeatable and can also provide precise geometric information of identified defects.

In the broad context of infrastructure management, the main goal of this thesis is to explore new and improved ways to use mobile robotic platforms, including sensor modalities and data-processing software, to enable precise and repeatable infrastructure inspection in an automated to semi-automated fashion. By automating the process of detecting, locating, and quantifying defects, the opportunity for more advanced asset management will emerge. This approach aims to fundamentally shift the current qualitative approach of most infrastructure inspections to a formal quantitative approach, leading to better quality, repeatability, and reliability of inspections while providing an opportunity to develop quantitative infrastructure condition models. Such quantitative models can be enabled by developing quantitative features (e.g., spall volume, crack lengths) as inputs required for probabilistic condition assessments (e.g., degradation modeling and residual life estimation).

[5] *maps*: in this work, maps refer to point cloud representations, which constitute a discrete sampling of the spatial extent of a structure in 3D. Maps represent the surfaces of the structure and can be post-processed to construct mesh representations or even further converted into a set of parametric shapes (usually coined as a building information model)

This thesis argues that a SLAM-centric approach is not only a viable approach to solve the above-stated broad goal for robot-aided infrastructure inspection, but this approach also unlocks more capabilities for robot-aided inspection than what has been shown in the literature to date. The majority of previous research focuses on small components of an overall inspection system without detailing how these small components can be broadly useful in practice. Examples of such components include building a robot specific for collecting data (images or pointclouds), detecting defects in 2D images, segmenting components or defects from pointcloud maps, etc. Very few works describe how such solutions can be integrated into an E2E versatile inspection system. Although useful, each component cannot be fully utilized without a method to co-locate all extracted inspection data in 3D space. All data (e.g., images, 3D lidar data, defects extracted from image or lidar data) must be precisely combined in the 3D world, enabling full context relative to the entire structure. This would allow inspectors to not only view such data in an easily digestible way, but also combine it with existing 3D Building Information Modeling (BIM) models and track changes in such data over time between subsequent inspections. Some works have attempted to generate inspection data tied to a 3D map; however, these solutions are far from perfect. In most cases, 3D maps are generated naively with only 2D images using Structure from Motion which has proven to have too many challenges to produce inspection maps reliably. In some works, multi-sensor data fusion including lidar, Inertial Measurement Unit (IMU), and cameras are used in SLAM framework, which does generate better maps. However, all SLAM-based inspection research uses either rudimentary SLAM solutions or off-the-shelf SLAM solutions with their robotic systems that were designed for inspection, not SLAM. SLAM works, unfortunately, prioritize localization accuracy, not map quality, which is more critical to inspection. The author is unaware of any inspection paper that has implemented State-of-the-Art (SOTA) SLAM methodologies designed to prioritize generating the most useful inspection maps. Furthermore, to the author's knowledge, no published studies show how to design a robotic system that is jointly optimized for collecting inspection and SLAM data. Lastly, no published studies have shown how to tailor an E2E inspection pipeline to best use the results from a SLAM-enabled platform. Examples of challenges from such a task that have been neglected in literature to date include: how to best use SLAM trajectories to generate the highest quality inspection map, how to fuse defects extracted from images with a 3D SLAM map, and how to jointly analyze 3D data labeled with defects.

The following section describes closely related literature and presents the gap that this thesis aims to fill with a SLAM-centric inspection system.

1.2 Research Landscape

This literature review aims to provide an overview of published work related to the inspection of infrastructure using robotics and associated technology. This includes robots used to perform any part of infrastructure inspection, as well as algorithms generated to process robotic-related data such as images, lidar data, etc. Chapter 3 will give a more detailed review of robotic systems built or used for inspection, outlining the limitations of the platforms built to date, which this thesis aims to solve. Chapter 4 will give a more targeted review of existing calibration methods and how a gap in solutions tailored towards platforms built for inspection exists. Finally, Chapter 5 gives a detailed review of inspection systems that have been proposed, which explicitly use SLAM to help generate inspection data. This final chapter demonstrates the lack of an E2E system designed from the ground up to produce inspection results in a SLAM-enabled framework.

Robotic inspection platforms can be divided into three general classes: ground (Uncrewed Ground Vehicle (UGV)), aerial (Uncrewed Aerial Vehicle (UAV)), and marine robots (Uncrewed Surface Vessel (USV) or Uncrewed Underwater Vessel (UUV)) [4]. Ground robots are among the most popular choices for inspection devices [4]. These robots can be sub-classified into wheeled, legged, and climbing robots. Aerial robots are also a popular choice since they have very few accessibility limitations. Since this thesis focuses on visual inspection of above-ground infrastructure, discussions on UUVs will be neglected.

A common approach to using robots for inspection is to rely on their mobility to enable data collection in areas that are usually difficult to access by humans. The typical examples of such systems are UAVs and climbing robots. In the case of UAVs for collecting data, they are usually equipped with visible and/or infrared spectrum cameras. These UAVs are used to collect close-up images of hard-to-access areas of a structure [48, 164, 99, 39, 127]. Inspectors then need to analyze these images after data collection to determine areas of concern. Analyzing large quantities of photos by an inspector can be challenging without knowing exactly what each photo is looking at in 3D space. Furthermore, when quantitative measurements are needed, these approaches would not suffice since the precise real-world scale is often hard to determine. Similarly, climbing robots have been used to collect equivalent information [115, 133]. These robots are often restricted to steel structures, therefore they are limited to a small subset of infrastructure. Furthermore, these robots are primarily small-scale in nature as they are designed to inspect local areas and not entire structures as a whole [4], and they are not sufficiently advanced for the long-term operation required for inspection of large structures [61].

To automate the process of analyzing photos, many researchers have implemented ways

to automatically detect features of interest from the images. Examples include crack detection using computer vision and machine learning [1, 2, 152, 153, 116, 80, 155], spall detection [41] or delamination detection from infrared images [91, 108]. For more information on image-based defect detection for concrete and asphalt inspection, refer to Feng et al. [42]. These methods do help reduce an inspector's time spent analyzing the data since it can automatically detect defects, however, this does not solve the scale ambiguity, nor does it help the inspector locate the defects on the structure rendering this data difficult to use. Monocular images can provide size information relative to the pixels in the images, however the pixel scale in world coordinates is unknown unless some reference scale is provided. To determine the size of a defect, each monocular image needs a reference scale or several reference scales if the pixel scale is inconsistent across the image.

Additional valuable information about the structure can be gathered by generating 3D point cloud representations of structures, enabling defects to be located and quantified (and tracked over time during periodic inspections) on this 3D map. Early attempts at generating 3D maps for inspection involved using images alone. The process of generating point clouds from monocular imagery is known as Structure from Motion (SfM)[6], which has been heavily studied for inspection [59, 141, 86, 66, 65]. 3D SfM models with tagged defects are useful for tracking defects over time, but the accuracy of quantification or tracking of small changes in defects is strictly limited by the quality of the point cloud. This limitation is a common issue with SfM generated point clouds, as they tend to be noisy, often have misaligned objects, and even miss crucial areas when little texture is available in the images. Previous studies, for example, Khaloo et al. [66], have acknowledged quality issues of point clouds generated using SfM and suggested using other sensor types. Overall, SfM as a means to generate point clouds of bridges is fraught with challenges such as poor/variable lighting conditions, lack of visual features to track in the images, and challenges associated with maintaining high and consistent overlap in the images. These problems are aggravated when no pose estimate (from Global Navigation Satellite System (GNSS), for example) is available for initialization of the optimization or to capture real-world scale in the 3D map.

On the other end of the spectrum, many researchers have attempted to use precise 3D mapping sensors to generate a map of a structure. The first attempts used Terrestrial Laser Scanner (TLS) equipment. TLS generated point clouds have been shown to be very useful for the condition assessment of infrastructure. [79, 84] showed that these point clouds can be used for concrete loss detection, and [79] even showed that concrete delamination can be detected by tracking concrete movement. [77, 142] showed that cracks can be detected in TLS point clouds, and dimensions (both width and length) can be measured. TLS

[6]SfM is a computer vision method that reconstructs a 3D environment from 2D images taken from different viewpoints.

has also been combined with images from an external camera to improve defect detection [144]. However, TLS processes are often expensive, time-consuming, and impractical to use on mobile platforms, thereby not offering the same ease of access advantages as mobile platforms such as UAVs or UGVs.

A mix of 2D images and mobile 3D sensing, such as lidars and depth cameras, provides an excellent opportunity to capitalize on the advantages of both systems. Properly combined, these two sensor modalities can provide a means to collect dense visual information from the cameras for detecting inspection quantities such as defects, while also using the 3D data to generate to-scale maps to combine the image data with quantitative measurements and tracking of quantities. Many of the existing works that combine these sensor modalities, however, focus on generating 2D maps for inspection of 2D surfaces such as bridge decks [82, 75, 76, 47, 114]. Unfortunately, these methods greatly limit the use cases to 2D spaces and usually are incapable of inspecting complex 3D geometry such as beams or columns.

Very few research studies have used a combination of visual and 3D sensing modalities for generating 3D maps of structures for inspection. Some works [62, 61, 69] combine lidar and camera data using SLAM, however, these systems are built without understanding the fundamental time synchronization and calibration needs to be able to do proper sensor fusion or SLAM from a mobile device. The CityFlyer [150] UAV does build a proper SLAM system that is also used for inspection with the SLAM generated 3D map. However, the CityFlyer relies on structured light-based Red Green Blue Depth (RGBD) sensors, which do not operate well in daylight at ranges required in many inspection applications. Feng et al. [40] present a similar solution that relies on solid-state lidar and an Red Green Blue (RGB) camera properly calibrated and time-synchronized, which does work at greater distances outdoors. Since the inspection procedure proposed in Feng's work depends on the pointcloud and mesh generated using R^3LIVE [83], this work is fundamentally tied to having a dense FOV overlap between the lidar and camera and therefore does not work with the more common traditional mechanically rotating lidars. Both [150] and [40] also require careful navigation during inspection to ensure they continuously maintain objects with significant environment structure within their narrow FOV so that tracking is maintained. Other essential characteristics of a SLAM-centric inspection system that the above works neglect to address are the following:

1. Enabling additional sensors to be integrated to tailor to a wide variety of inspection use cases through a configurable inspection system,

2. Generating 3D maps optimized for inspection as opposed to SLAM,

3. Correlating and tracking inspection results between multiple inspections.

In summary, no E2E research solutions in the literature demonstrate that accurate and repeatable measurements of inspection quantities can be obtained from a mobile mapping solution.

1.3 Contributions

This thesis presents a novel framework for E2E mobile robotic-aided visual inspection of infrastructure. The core novelty of the framework is to create a SLAM-centric design where the platforms and algorithms are designed from the ground up with the idea of running multi-sensor SLAM to generate precise 3D maps for measuring and storing inspection results. The proposed system is designed to produce accurate and reliable 3D maps by following SOTA SLAM in robotics literature with additional inspection-specific modifications. SLAM results are combined with image data in novel ways, unlocking the ability to detect, locate, quantify, and track changes in defects over multiple inspections. A defect detection methodology is used, which enables defects to be detected using machine learning techniques on images and enables results to be transferred to the 3D map, where the defects can then be measured and tracked over time. The main contributions of this thesis are as follows:

- **Inspection system design**: The first multi-sensor SLAM-centric inspection system that optimizes sensor layout for SLAM and inspection. Sensor requirements are discussed including synchronization, and mobility requirements for such a system to adequately solve the challenges of infrastructure inspection. The design demonstrates how a system can be adapted for use on various mobile platforms, satisfying the inspection needs for a wide range of infrastructure asset types.
- **Target-based extrinsic calibration of non-overlapping camera-lidar systems**: The first target-based extrinsic calibration methodology using a Motion Capture System (MCS) for calibrating any camera-lidar system that has no restrictions on the FOV overlap of the sensors. This method achieves similar accuracy as state-of-the-art target-based camera-lidar calibration approaches, and the accuracy is independent of the number of sensors being calibrated, in contrast to existing methods.
- **Multi-sensor online SLAM with offline refinement and de-coupled inspection map generation**: An online SLAM solution with offline refinement to ensure

reliable map generation in real-time while ensuring data can be post-processed to generate the highest quality maps needed for inspection tasks. This also includes a de-coupled localization and inspection map generation process which produces much better maps for inspection rather than taking maps generated from SLAM.

- **Automated defect assessment methodology**: The first E2E inspection methodology that demonstrates how to process slam-centric robotic inspection data in order to detect, quantify, and track defects over time for an entire piece of infrastructure, including examples shown on a bridge and parking garage structure.

1.4 Thesis Organization

The remainder of this thesis is divided into six chapters as follows:

- **Chapter 2**: To ensure clarity and understanding of the work presented in this thesis, this chapter starts by giving an introduction to 3D geometry and discusses the notation used throughout this thesis. Next, the principal theories and algorithms used in practice are described, such as localization and mapping, scan registration, camera modeling, and sensor calibration.

- **Chapter 3**: In this chapter, the SLAM-centric mobile inspection system design is presented. Achieving versatile mobility to suit the needs of any type of inspection is discussed, as well as the best sensor configurations and synchronization strategies for such a system.

- **Chapter 4**: A novel calibration methodology is presented, motivated by the non-overlapping nature of the sensor requirements given in Chapter 3.

- **Chapter 5**: The E2E inspection pipeline is presented, which enables the generation of automated defect detection, localization, quantification, and change tracking. This includes the description of the full SLAM and mapping methodology, as well as defect detection, map labeling, and defect quantification and tracking. Calibration results using the algorithms from Chapter 4 are applied to the custom inspection platform design presented in Chapter 3, which collects the data used to generate inspection results presented in this chapter.

- **Chapter 6**: In the final chapter, the main contributions of the thesis are summarized, and the main takeaways from this research are discussed, including areas for future study and potential impact of the proposed technology.

Chapter 2

Background

This chapter provides the mathematical background for the work presented in the rest of the thesis. First, an overview of relevant 3D geometry is presented along with the notation used. A primer on localization and mapping is then presented, along with the basic mathematical formulations that govern these principles and some typical example solutions. Next, scan registration is defined, and the variations used in this work are described in detail, followed by camera modeling. Finally, sensor calibration is described, including intrinsic and extrinsic calibration.

2.1 Three-Dimensional Geometry

Let $\mathbf{P^a}$ be the coordinates of a point $a \in \mathbb{R}^3$. $\mathbf{P^a_M}$ is defined as the vector from the origin of the coordinate frame $\mathcal{F}_\mathbf{M}$ to the location of point a. Similarly, $\mathbf{P^a_B}$ is defined as the vector from the origin of frame $\mathcal{F}_\mathbf{B}$ to point $\mathbf{P^a}$. Fig 2.1 illustrates these variables graphically.

To convert points between different Euclidean reference frames, the rotation and translation between the two frames is needed. The translation from frame $\mathcal{F}_\mathbf{M}$ to frame $\mathcal{F}_\mathbf{B}$ is expressed as a vector from frame $\mathcal{F}_\mathbf{M}$ to frame $\mathcal{F}_\mathbf{B}$ and can be written as $\mathbf{t_{MB}}$. Note that three quantities are actually needed to define a translation vector: the origin frame, the destination frame, and the frame in which the vector is expressed. Commonly, this is expressed as ${}_\mathbf{M}\mathbf{t_{MB}}$, however, to simplify notation, this thesis will drop the prescript and assume the vector is always expressed in the destination frame resulting in the notating: $\mathbf{t_{MB}}$. Similarly, for rotations, $\mathcal{R}_\mathbf{MB}$ is a 3×3 rotation matrix that represents the rotation from frame $\mathcal{F}_\mathbf{B}$ to frame $\mathcal{F}_\mathbf{M}$. Using this convention, any point a expressed in one frame can be converted to another frame using Eq. 2.1:

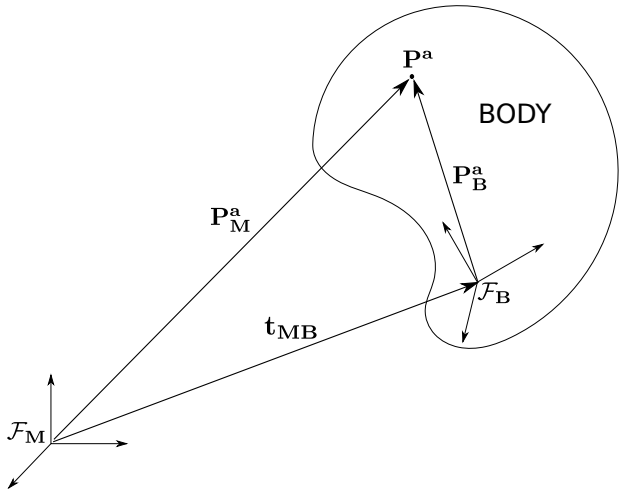


Figure 2.1: Background on 3D geometry

$$\mathbf{P^a_M} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_M^a = \mathcal{R}_\mathbf{MB}\mathbf{P^a_B} + \mathbf{t_{MB}} \tag{2.1}$$

Eq. 2.1 can also be rewritten in a more compact form using transformation matrices, as shown in Eq. 2.2. Where $\mathcal{T}_\mathbf{MB}$ is the 4×4 transformation matrix that represents the transformation from frame $\mathcal{F}_\mathbf{B}$ to frame $\mathcal{F}_\mathbf{M}$, expressed in frame $\mathcal{F}_\mathbf{M}$. Again, for simplicity, $\mathcal{T}_\mathbf{MB}$ in this work is equivalent to ${}_\mathbf{M}\mathcal{T}_\mathbf{MB}$. It should also be noted that $\mathbf{P^a_M}$ now must be written in homogeneous form due to the use of the transformation matrix.

$$\mathbf{P^a_M} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_M^a = \mathcal{T}_\mathbf{MB}\mathbf{P^a_B}; \quad \mathcal{T}_\mathbf{MB} = \begin{bmatrix} \mathcal{R}_\mathbf{MB} & \mathbf{t_{MB}} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.2}$$

Note that $\mathcal{R}_\mathbf{MB}$ is a member of the Special Orthogonal group $\mathbf{SO(3)}$, whereas $\mathcal{T}_\mathbf{MB}$ is a member of the Special Euclidean group SE(3). The set of rotation matrices and transformation matrices do not form a vector space, in fact they form matrix Lie groups which require special algebra. The mathematics behind matrix Lie groups can be found in references such as [12] and are omitted here for the sake of brevity.

2.2 Localization and Mapping

To better understand the mapping presented in this work, it is helpful to first understand the state estimation problem. Note that the terms state estimation and localization will be used interchangeably in this section. State estimation is the more generic term, whereas localization is used when referring to estimating some rigid body's 6Degree(s) of Freedom (DOF) pose using state estimation theory. The same solutions that exist for state estimation are also adapted to solve the problem of SLAM, which will be discussed later. The main difference between localization and SLAM is that the state gets augmented to include map states in the optimization. State estimation can generally be divided into two main approaches: (1) recursive and (2) batch. Since all the sensors record data in discrete intervals, only discrete-time state estimation will be discussed.

Recursive state estimation is often referred to as filtering or probabilistic state estimation [38, 10]. For recursive state estimation, the state is estimated at every time step using the information available up to that time step. These filters assume a Markov process, meaning that all previous information can be contained within the current state [38], eliminating the need to store historical measurements. In other words, filter-based methods update their estimate of the state at each time step considering only the previous state and the new information available at the current time step (new measurements and inputs).

Eq. 2.3 shows the general mathematical formulation of recursive estimators:

$$P(x_t|x_{0:t}, u_{0:t}, y_{0:t}) \rightarrow \mathcal{N}(\hat{x}_t, \hat{P}_t) \tag{2.3}$$

For the current time step, t, the probability distribution P of the state at the current time, x_t is estimated, given all previous states $x_{0:t-1}$, inputs ($u_{0:t}$) and all previous measurements ($y_{0:t}$). Assuming the solution is Gaussian, which is necessary for most recursive estimators that will be considered, the *estimated* mean state ($\hat{x}_t$) and associated covariances ($\hat{P}_t$) can be computed. Using the Markov assumption, Eq. 2.3 simplifies to:

$$P(x_t|x_{t-1}, u_t, y_t) \rightarrow \mathcal{N}(\hat{x}_t, \hat{P}_t) \tag{2.4}$$

Common examples of recursive state estimation include the Bayes filter and the Kalman filter [132]. For linear systems with Gaussian noise, the Kalman filter solution is the best linear unbiased estimate [12]. However, these solutions cannot be applied for nonlinear systems, which is the case with most robotics systems, or for non-Gaussian noise. With nonlinear systems and/or non Gaussian noise, other approaches have been developed, including the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), and Particle

Filter (PF) [132]. All of these solutions are extensions of the Bayes filter. The fundamental limitation of all these methods, which limits the possible accuracy of the results, is that they employ the Markov assumption. With large nonlinear systems, the EKF and UKF can diverge by linearizing about incorrect points, or they can become biased or inconsistent [12]. The PF attempts to overcome this problem using a Monte-Carlo approach instead of linearizing the systems, however, PF formulations tend to be computationally expensive for systems with large orders. For best results with nonlinear systems, batch state estimation is often a better alternative.

Batch state estimation sets up the problem as an optimization problem over the entire trajectory of the robot. The solution is one that is optimal over the entire set of inputs and measurements, simultaneously. Eq. 2.5 shows the general formulation of the batch estimation problem:

$$\hat{P}_{0:T} = \underset{x_{0:T}}{\arg\min}(x_{0:T}|\bar{x}_{0:T}, y_{0:T}, u_{0:T}) \tag{2.5}$$

Here, T is the final time step, and $\bar{x}$ is the initial estimate of the poses. For localization, the initial estimate is often determined using the system inputs and the motion model for the robot or using inertial measurements such as those provided by an IMU. Batch solutions are often referred to as the *gold standard* solutions since they make fewer assumptions than the filtering-based approaches and use all of the available information [12]. The significant drawback to these methods is that they are generally more challenging to implement due to the nonlinear optimization process and tend to be computationally expensive. To implement these approaches *online*[1], batch approaches can be adapted using a sliding window approach, which only considers a window to the most recent measurements in the optimization process [12].

The following subsections will present the mathematical formulation for the EKF and the batch state estimation problems. For batch state estimation, the general batch nonlinear non-Gaussian formulation and solution will be given. Specific nonlinear solvers that can be applied to the optimization problem will also briefly be discussed.

[1] *online/offline*: in this work, online processing refers to processing data in real-time. Conversely, offline processing refers to processing data at a rate that is lower than the data rates produced by the sensors. Here, online means the state can be estimated on-board the robot as it moves. Batch state estimation usually needs to be applied offline after the robot has navigated the environment.

2.2.1 Recursive EKF Localization

The EKF uses a two-step approach: (1) a prediction step and (2) a correction step. For robot localization, the prediction step uses the robot motion model and the inputs (e.g., thrust applied to a UAV) to predict the robot's state. This prediction is then updated using the measurements (e.g., GNSS position measurements). Before presenting the Extended Kalman Filter (EKF), the variables, and models used in the problem formulation will be discussed.

Eq. 2.6 shows a generic nonlinear motion model for a robot:

$$x_t = f(x_{t-1}, u_t, w_t); \quad t = (1, 2, ..., T) \tag{2.6}$$

$$x_t \in \mathbb{R}^N; \quad x_0 \sim \mathcal{N}(\hat{x}_0, \hat{P}_0); \quad w_t \in \mathbb{R}^N \sim \mathcal{N}(0, Q_t)$$

As before, let x_t be the vector representing pose at time t. For robot localization, the pose usually consists of x, y, z and $roll, pitch, yaw$ (or some variant rotational representation such as quaternions), which represents the position in some fixed reference frame and the orientation in the body (robot) frame. Let y_t represent the vector of measurements at time t, u_t represent the vector of inputs at time t, and w_t represent the current noise parameter which is assumed to be zero mean Gaussian, where the covariances (Q_t) are determined based on the noise of the raw input commands. The motion model is the expected pose at time t given the previous pose, current inputs, and current noise parameters (w_t). x_t and w_t are in $\mathbb{R}^N$, where N equals the number of components used to define the pose. Note that some prior ($\hat{x}_0$) on the pose is required for the EKF (assumed to be Gaussian). This prior can be set in different ways. Often, the coordinate system of the map is set to be at the location of the initial pose, therefore the position is set to zero, the rotation matrix is set to identity (or Euler angles set to zero), and the covariance on the prior is set to be low (high confidence level on the prior).

Eq. 2.7 shows a generic nonlinear measurement model:

$$y_t = g(x_t, n_k); \quad t = (0, 1, ..., T) \tag{2.7}$$

$$y_t \in \mathbb{R}^M; \quad n_t \in \mathbb{R}^M \sim \mathcal{N}(0, R_t)$$

This measurement model relates the expected sensor measurements at time t to the current pose and noise parameters (n_t). Again, n_t is assumed to be zero mean Gaussian,

where the covariances (R_t) are determined based on the noise of the raw measurements. Note that here, y_t and n_t are in $\mathbb{R}^M$ where M is the number of measurements received at each time step. The derivation of the EKF solution can be found in [12], where the equations are derived as special cases of the general Bayes Filter. Equations 2.8 to 2.12 show the canonical form of the EKF.

$$\bar{P}_t = F_{t-1}\ \hat{P}_{t-1}\ F_{t-1}^T + Q_t^{'} \tag{2.8}$$

$$\bar{x}_t = f(\hat{P}_{t-1}, u_t, 0) \tag{2.9}$$

$$K_t = \bar{P}_t\ G_t^T (G_t\ \bar{P}_t\ G_t^T + R_t^{'})^{-1} \tag{2.10}$$

$$\hat{P}_t = (\mathbf{I} - K_t\ G_t)\bar{P}_t \tag{2.11}$$

$$\hat{P}_t = \bar{x}_t + K_t(y_t - g(\bar{x}_t, 0)) \tag{2.12}$$

In the above equations, the prediction step consists of equations Eq. 2.8 and 2.9, whereas the correction step consists of equations Eq. 2.11 and 2.12. Eq. 2.10 is known as the Kalman Gain equation. Note that in all these equations, $\bar{(.)}$ is used to show *predicted* variables, $\hat{(.)}$ is used to show the *corrected* parameters (or the final estimate of the states and covariances), and $\mathbf{I}$ is the identity matrix. Furthermore, F, G, $Q^{'}$, and $S^{'}$ are the linearized motion model, measurement model, motion noise covariances, and measurement noise covariances, respectively. Eq. 2.13 to 2.16 show the linearized terms. Note that the motion model and motion noise are linearized about the corrected pose at the previous time step, whereas the measurement model and measurement noise are linearized about the predicted pose of the current time step.

$$F_{t-1} = \left.\frac{\partial f(x_{t-1}, u_t, w_t)}{\partial x_{t-1}}\right|_{\hat{P}_{t-1}, u_t, 0} \tag{2.13}$$

$$w_t^{'} = \left.\frac{\partial f(x_{t-1}, u_t, w_t)}{\partial w_t}\right|_{\hat{P}_{t-1}, u_t, 0} w_t \tag{2.14}$$

$$G_t = \left.\frac{\partial g(x_t, n_t)}{\partial x_t}\right|_{\bar{x}_t, 0} \tag{2.15}$$

$$n_t^{'} = \left.\frac{\partial g(x_t, n_t)}{\partial n_t}\right|_{\bar{x}_t, 0} n_t \tag{2.16}$$

2.2.2 Batch Localization

For batch localization, the same initial state and measurement vectors (x_t and y_t) are used for each timestamp from $0 : T$, as well as the same motion and measurement models as defined above in Eq. 2.6 and Eq. 2.7, respectively. For batch optimization, the state at all time-steps are being estimated, therefore a stacked version of the state vector, x is used as shown in Eq. 2.17:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_T \end{bmatrix} \tag{2.17}$$

Similarly, a stacked version of errors, $e(x)$, must be defined as shown in Eq. 2.18:

$$e(x) = \begin{bmatrix} e_u(x) \\ e_y(x) \end{bmatrix} = \begin{bmatrix} e_{u,0}(x) & e_{u,1}(x) & \dots & e_{u,T}(x) & e_{y,0}(x) & e_{y,1}(x) & \dots & e_{y,T}(x) \end{bmatrix}^T \tag{2.18}$$

The measurement error, $e_y(x)$, can be defined as the difference between the true measurement and the expected measurement using the measurement model as shown in Eq. 2.19:

$$e_{y,t}(x) = y_t - g(x_t, 0); \quad t = (0, 1, ..., T) \tag{2.19}$$

For the motion error, $e_u(x)$, however, the error may be defined differently based on the problem formulation. For example, a simple motion model based on inputs can result in Eq. 2.20 and Eq. 2.21:

$$e_{u,t}(x) = f(x_{t-1}, u_t, 0) - x_t; \quad t = (1, ..., T) \tag{2.20}$$

$$e_{u,t}(x) = \hat{x}_t - x_t; \quad t = 0 \tag{2.21}$$

When the motion model is formulated using SE(3), an error term cannot be defined by subtracting the predicted pose from the actual pose since transformation matrices do not

form a vector space (not closed under addition). Instead, a motion error can be defined as shown in Eq. 2.22 and Eq. 2.23:

$$e_{u,t}(x) = \ln(\exp(\ \Delta t_t \omega_t^{\hat{\circ}}\ )\boldsymbol{\mathcal{T}_{t-1}}\boldsymbol{\mathcal{T}_t}^{-1})^{\check{\circ}}\ ; \quad t = (1, ..., T) \tag{2.22}$$

$$e_{u,t}(x) = \ln(\hat{\boldsymbol{\mathcal{T}_t}}\boldsymbol{\mathcal{T}_t}^{-1})^{\check{}}; \quad t = 0 \tag{2.23}$$

where ($\hat{\circ}$) and ($\check{\circ}$) are the skew-symmetric transform and inverse skew-symmetric transform, respectively. ω_t are the rotation rates and linear velocities, which are inputs in this example. Therefore, the estimated pose $\hat{\boldsymbol{\mathcal{T}_t}}$ can be obtained by multiplying ω_t by the time interval Δt_t and by the previous pose, $\boldsymbol{\mathcal{T}_{t-1}}$. More details on this SE(3) motion model and alternative models can be found in [12]. As shown in Eq. 2.23, if the predicted pose and measured pose are equal, then $\hat{\boldsymbol{\mathcal{T}_t}}\boldsymbol{\mathcal{T}_t}^{-1}$ would tend to identity, and therefore the log would also tend to zero resulting in no error added to the optimizer.

The objective function, $F(x)$, can then be set up as a sum of the squared error terms weighted by the inverse of the covariances (W) as expressed in Eq. 2.24:

$$F(x) = \frac{1}{2} e(x)^T W^{-1} e(x); \quad t = (1, ..., T) \tag{2.24}$$

This sets up a nonlinear optimization problem, which can be solved using many tools available in literature [12]. For state estimation, this optimization problem is usually solved using Gauss-Newton or Levenberg-Marquardt optimization, and the details for this approach are described in [12]. Solving the batch state estimation and SLAM problems using Lie Groups and Lie Algebra for representing the robot states are also described in detail in [12]. Using Lie Groups for representing robot states requires special precaution to define perturbation terms and error terms in Lie Algebra space (SE(3)). To further accelerate computations, tools are often used to exploit the sparsity in the state estimation or SLAM problem, including the Schur complement or Cholesky decomposition [12]. It should also be noted that the nonlinear optimization problems that describe these batch problems also require some initial estimates for the state before optimizing. These initial estimates can be obtained from sensor measurements (e.g., GNSS) or from a motion model (similar to the EKF localizer presented above).

A common way to setup the batch state estimation problem for easy visualization and implementation is through the use of Factor Graph (FG)s. A FG is a graphical method originating from coding theory, which is to be used to construct an optimization

problem [87]. Fig. 2.2 shows an example of a FG used for a simple state estimation problem. A FG consists of nodes and edges. Each node represents a state, whereas each edge represents a factor. The graph in Fig. 2.2 has three states (x_1, x_2, x_3) and six factors (f_1 to f_6). In this state estimation problem, f_0 is the prior estimate on the state, f_1 and f_2 are motion estimates, and f_3 to f_5 are measurements. A prior is an initial estimate of a state, which is mathematically expressed as a measurement on that state with covariance representing the uncertainty on that estimate. This initial estimate may be from a sensor measurement of the pose in a global reference frame or a marginalization prior that results from removing old states from the batch problem in a windowed batch optimizer (to be discussed later). Initial pose priors in robot localization can also come from a relocalization solution, where the pose of the robot relative to a pre-built map is estimated, for example, see [104]. The motion estimates, f_1 and f_2, may come from odometry measurements, integration of inertial measurements, or motion model estimates based on robot inputs. Often in FG terminology, unary and binary factors will be used to describe a type of constraint. A unary factor is a constraint that is only connected to a single state, such as a prior or absolute state measurement (e.g., GNSS position, or IMU orientation). A binary factor is a constraint connecting two states, for example a wheel odometry or a scan registration measurement.

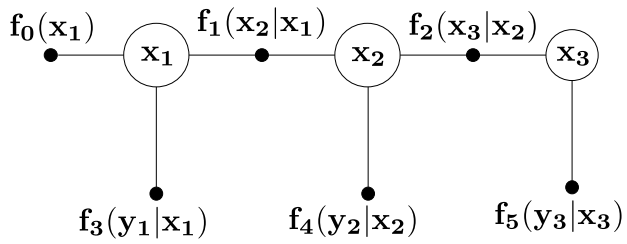


Figure 2.2: Generic illustration of a factor graph with six factors and three nodes

If the constraints are formulated in a FG as probability distributions (e.g., $f_1(x_2|x_1) = P(x_2|x_1)$), then Maximum A-Posteriori (MAP) inference can be performed to solve the graph by solving for the maximum of the factors multiplied together as shown in Eq. 2.25 and Eq. 2.26, where $\mathcal{L}(x_1, x_2, x_t)$ is the likelihood function.

$$\hat{P} = \arg\max_{x} \ \mathcal{L}(x_1, x_2, x_3) \tag{2.25}$$

$$= \arg\max_{x} \ f_0(x_1) f_1(x_2|x_1) f_2(x_3|x_2) f_3(y_1|x_1) f_4(y_2|x_2) f_5(y_3|x_3) \tag{2.26}$$

$$= \arg\max_{x} \ P(x_0) \ \prod_{t=1}^{T} P(x_t|x_{t-1}) P(y_t|x_t) \tag{2.27}$$

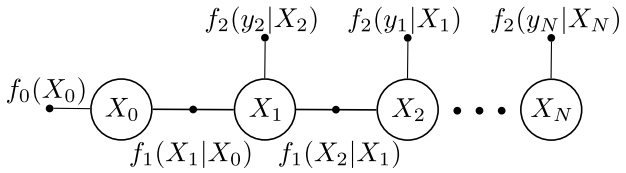
To relate this to the batch problem described above, it can be shown that this formulation can be simplified to the objective function shown in Eq. 2.24. Assuming the factors are zero-mean Gaussians (i.e., the motion and measurement models have zero-mean Gaussian noise), Eq. 2.27 can be written as:

$$\hat{P} = \arg\max_{x} \ \prod_{t=1}^{T} \exp\big[-\frac{1}{2} \ e_u(x_t)^T W_u^{-1} e_u(x_t) \big] \ \exp\big[-\frac{1}{2} e_y(x_t)^T W_y^{-1} e_y(x_t) \big] \tag{2.28}$$

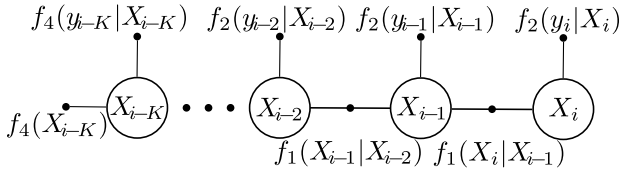
$$= \arg\min_{x} \ \Big[\sum_{t=1}^{T} \frac{1}{2} \ e_{u,t}(x_t)^T W_{u,t}^{-1} e_{u,t}(x_t) + \sum_{t=1}^{T} \frac{1}{2} e_{y,t}(x_t)^T W_{y,t}^{-1} e_{y,t}(x_t) \Big] \tag{2.29}$$

$$\tag{2.30}$$

To enable batch estimation for online mapping, it is common to take advantage of marginalization to limit the size of the optimization problem, or in other words, limit the number of states and measurements being solved at one time. This ensures that the optimizer can run at a fixed frequency to provide a real-time SLAM estimate for robot control or motion planning. The process of bounding the measurements via marginalization is commonly referred to as a sliding window optimizer or fixed-lag smoother, where the term window refers to the time windows of measurements and states retained by the optimizer after marginalization, and lag refers to the length of this window in terms of number of states. The mathematical formulation of the marginalization process is omitted, however more information can be found in [12]. Figures 2.3a and 2.3b illustrate the difference between a full batch FG and a sliding window FG. In these figures, f_0 is the prior on the first pose, f_1 is a binary factor between two consecutive poses (e.g., scan registration measurement), f_2 is a unary measurement factor on a pose (e.g., GNSS measurement), and f_4 is the unary marginalization factor on the first pose in the window. f_0 and f_4 are



(a) Batch factor graph



(b) Sliding window factor graph

Figure 2.3: Comparison between (a) a batch factor graph, and (b) a sliding window factor graph with binary factors between consecutive poses and unary measurement factors on each pose.

mathematically implemented the same in the FG, however, the mean and covariance of the f_4 factor are determined during marginalization, and they represent the estimate of the state given all previous information. The longer the optimization window, the more the optimization resembles full batch localization, and in the extreme case where the window is of size 0, the result is equivalent to a recursive estimator.

2.2.3 Simultaneous Localization and Mapping

The process of localizing a robot in an unknown environment while building a map of the environment is well studied in the robotics literature and referred to as SLAM [38]. SLAM is a challenging problem because localization requires knowledge of the map while map building requires knowledge of where the robot is within this map [132]. Therefore, errors in the robot pose estimate will propagate to errors in the map and vice-versa. Such errors will continue to grow over time unless global measurement sources with bounded errors are incorporated, such as GNSS measurements or returning to previously visited locations (loop closure). Despite a large volume of literature [11, 38], SLAM continues to be an active research area, with the most recent focus being in robust, task-aware SLAM with semantic maps [22], which is the focus of the SLAM work in this thesis.

To solve the SLAM problem, the localization methods described in Sections 2.2.1 and 2.2.2 are extended to also estimate the state of the environment or map features. Examples

of environmental features that would be extracted from the sensor data are lidar features such as surfaces or edges in the scan lines [148], or image features such as ORB [104] or SIFT [88] features. The state of the detected map features are appended to the state vector in the state estimation problem as shown in Eq. 2.31:

$$x = \begin{bmatrix} x_0 & x_1 & \dots & x_T & l_0 & l_1 & \dots & l_M \end{bmatrix}^T \tag{2.31}$$

l_m is landmark m or map feature m. Since the standard optimizers used in SLAM solve the objective function (Eq. 2.24) using the first and/or second derivatives of the objective function (and therefore of the error functions), the major difference with the SLAM optimizers is that they are also required to compute the jacobians of the error functions with respect to the map features. For an in-depth description of the solutions to recursive and batch SLAM problems, refer to [12].

At the early stages of SLAM research (in the 1990s), filtering-based methods were dominant. The main reason behind the popularity of these approaches was the low computational resources needed to run these algorithms in real-time while still properly capturing the probabilistic characteristic of robotic systems (i.e., sensor noise, input noise, state uncertainty) [10]. This popularity is still common in many state estimation problems where computational resources are low or more advanced state estimation solutions are unnecessary. As a result, many open-source EKF SLAM software solutions are available for researchers to use. In the last few decades, sliding window batch estimators have become the dominant back-end for SLAM as these methods have become more efficient, the computational power of devices has increased, and the accuracy of these approaches has been significantly better than the recursive approaches.

2.3 Scan Registration

Scan registration is the process of comparing one scan to another scan to determine the rigid body transformation that explains the motion between the reference frames of the two scans. For example, in a mobile lidar data collection scenario where successive scans are collected over time and distance, scan registration can be used to estimate the motion of the lidar. The term scan describes a set of points collected from a ranging sensor (e.g., lidar or RGBD camera) at some time interval. The term pointcloud refers to a set of points in some reference frame and can be a single scan or a set of scans converted to the same reference frame. Pointclouds and scans are often used interchangeably. Other

terminology used in place of scan registration are scan matching, pointcloud alignment, point set alignment, and point set registration.

When using scan registration for localization, there are two general approaches: scan-to-map registration or scan-to-scan registration. The former registers each new scan to a pointcloud map to estimate the absolute pose of the scan relative to the map frame. The map may be built incrementally from new scan registration measurements or from any previous map generation process. The latter registers each new scan to its previously collected scan to find the relative motion between the two scans. Both methods are used in this work, as will be discussed in Chapter 5. The following subsections will describe the common scan registration approaches that are relied upon heavily in this work.

It should be noted that scan registration is not limited to estimating rigid body transformations. Scan registration can also be used to estimate scale. An example of when scale estimation would be required in scan registration is while registering a visual SLAM or SfM map to a map with scale (e.g., lidar map or visual-inertial SLAM map). When using vision alone for mapping, a pointcloud can be generated with an unknown scale. Registration can then be performed against a map with scale to recover the unknown scale parameter of the visual map. This work, however, does not use these types of scan registration, therefore their formulation will be omitted from this section.

2.3.1 Iterative Closest Point and Variants

Iterative Closest Point (ICP) registration is one of the most common scan registration methods, and many extensions and variations have been proposed, including variations to the selection of matches, error terms between matching points, and the minimization strategy [122]. To give context to ICP methods and their variants, this section will describe a few common ICP approaches.

ICP is a least squares optimization problem where the cost function being minimized is the summation of some squared error metric between points on the *target cloud* to corresponding points on the *reference cloud* [16]. The point correspondences are estimated by taking the nearest neighboring points on the other scan, and the process is iterated by updating the correspondences after each optimization until reaching a predefined threshold [16]. This allows the transformation to be estimated between each target and reference cloud.

The first method of ICP is the Point-to-Point (P2p) ICP, where the sum of the Euclidean distance between points on the new cloud to the nearest points on the reference cloud is used in minimization. This is expressed in Eq. 2.33:

$$\mathcal{T}_{\mathbf{RT}} = \arg\min_{\mathcal{T}_{\mathbf{RT}}} \Psi(S_T, S_R) \tag{2.32}$$

$$= \arg\min_{\mathcal{T}_{\mathbf{RT}}} \frac{1}{N_s} \sum_{i=1}^{N_s} [\mathbf{P_R^j} - \mathcal{T}_{\mathbf{RT}}\ \mathbf{P_T^i}]^2 \tag{2.33}$$

In Eq. 2.33, N_s is the number of corresponding scan point pairs, S_T is the set of points in the target scan, S_R are the set of points in the reference frame, $\mathcal{T}_{\mathbf{RT}}$ is the transformation from the target frame to the reference frame that is being optimized for, and $\mathbf{P_R^j}$ and $\mathbf{P_T^i}$ are the corresponding reference and target points expressed in their scan frames, respectively.

A simple yet common example of a variant to ICP is the Point-to-Plane (P2N) ICP. The cost-minimizing metric in this method is the distance between points on the target cloud to a plane defined by the points surrounding the corresponding point on the reference cloud [123]. The planes in this algorithm are defined by local clusters of points. The details for P2N ICP can be found in [123]. Another common extension to this is that of generalized-ICP, or Generalized Iterative Closest Point (GICP) [126]. In GICP, the distance minimized is between the planes defined by a cluster of points around each corresponding point on target and reference clouds. For more ICP variants, refer to [122].

2.3.2 LOAM Matching

This work refers to Lidar Odometry and Mapping (LOAM) matching as the registration approach presented in the seminal paper by Zhang et al. (2014) [157, 159, 158]. LOAM was first introduced as an odometry method that uses lidar and IMU (and later extended to use visual SLAM) for 6DOF pose estimation with low computational complexity and low drift. Prior to LOAM, the majority of scan registration methods were too computationally expensive to be run in real-time on low-powered devices (such as UAVs). LOAM introduced the first lightweight registration method specifically designed for lidar data by only registering edge features to lines and planar features to surfaces. Note that here, edge features refer to a set of points that form an edge in the lidar scan and surface features refer sets of planar points, however, these are not to be confused with visual features that are common in visual SLAM or SfM. Visual features are described using an interest point (e.g., corner) and a descriptor (e.g., ORB descriptor [104]), whereas LOAM features are only an interest point (either edge or surface lidar point) without an associated descriptor. Therefore, LOAM registration must use a nearest neighbor approach for correspondence estimation, similar to ICP and its variants.

The first stage of LOAM registration is to extract edge and surface features from a scan. To do this, the scan is first separated into scan rings, where one scan ring refers to the points collected during one full revolution of one light beam in a mechanically rotating multi-beam lidar. Then, the volume occupied by the scan points is separated into even sub-volumes to ensure that feature extraction is distributed along all regions of the scan. Next, surface smoothness is calculated for each point in the scan according to Eq. 2.34, and points are grouped into edge and planar features based on large and small smoothness values.

$$c_m^i = \frac{1}{|C_m^i| \cdot ||P_m^i||} \left|\left| \sum_{P_m^j \in C_m^i, j \neq i} (P_m^i - P_m^j) \right|\right| \tag{2.34}$$

In Eq. 2.34, c_m^i is the smoothness for point i in scan m, $P_m^i = [x_m^i, y_m^i, z_m^i]$ is a 3D point i expressed in the frame of scan m, and C_m^i is the set of consecutive points of i in the same scan line. Intuitively, this assigns small smoothness to a point if its neighboring points have large depth discontinuities. To select the final features to use for registration, the largest and smallest smoothness values are selected in each sub-volume while avoiding points that: (1) have neighbors that have already been selected, (2) are points on the boundary of occluded regions, or (3) are points whose local surface direction is near parallel to the laser beam that measured that point.

Correspondence estimation is done by searching for neighboring points in the reference scan after the target scan has been transformed into the estimated frame of the reference scan. It is also possible to perform LOAM registration to a LOAM map, which is an aggregate of LOAM features extracted from scans which have been transformed into a common reference frame. In this case, the corresponding points are searched for in the LOAM map, and the pose of the target scan is estimated relative to the map frame. For each edge feature in the target scan, the two nearest edge features in the reference cloud are taken as correspondences, and a constraint is added to the optimizer if the two correspondences are found within some proximity of the edge point. Eq. 2.35 describes the constraint added for edge features, where this constraint represents a distance from a point in the target scan to a line defined by the two correspondences in the reference scan.

$$d_{e,m}^i = \frac{\left|(\hat{P}_m^i(\theta_m, t_m) - \hat{P}_{m-1}^j) \times (\hat{P}_m^i(\theta_m, t_m) - \hat{P}_{m-1}^k)\right|}{\left|\hat{P}_{m-1}^j - \hat{P}_{m-1}^k\right|} \tag{2.35}$$

In Eq. 2.35, $d^i_{e,m}$ is the distance error for edge elements, P^j_{m-1} and P^k_{m-1} are the corresponding edge points in the scan frame $m-1$ (i.e., the reference scan) and $\hat{P}^i_m(\theta_m, t_m)$ is the predicted point position of feature point i from scan frame m, converted the reference frame $(m-1)$ and is defined in Eq. 2.36:

$$\hat{P}^i_m(\theta_m, t_m) = \hat{T}_{m,m-1}\hat{P}^i_{m-1} \tag{2.36}$$

In Eq. 2.36, θ_m represents the rotation between the two frames, and t_m represents the translation. $\hat{T}_{m,m-1}$ is the predicted transform from scan frame $m-1$ to scan frame m. The predicted pose can be obtained from some motion prior (e.g., IMU pre-integration or a visual-inertial estimator) or from a motion model.

Similarly, for each surface feature in the target scan, the three nearest surface features in the reference cloud are taken as correspondences, and a constraint is added to the optimizer if the three correspondences are found within some proximity of the surface point. Eq. 2.37 describes the constraint added for surface features, where this constraint represents a distance from a point in the target scan to a plane defined by the three correspondences in the reference scan.

$$d^i_{s,m} = \frac{\left|(\hat{P}^i(\theta_m, t_m) - P^j_{m-1})(P^j_{m-1} - P^k_{m-1}) \times (P^j_{m-1} - P^l_{m-1})\right|}{\left|(P^j_{m-1} - P^k_{m-1}) \times (P^j_{m-1} - P^l_{m-1})\right|} \tag{2.37}$$

In Eq. 2.37, $d^i_{s,m}$ is the distance error for surface measurements while P^j_{m-1}, P^k_{m-1} and P^l_{m-1} are the corresponding points in the reference scan to surface feature i in the target scan. To solve for the motion between the two scans, a nonlinear least squared optimization problem can be formulated by summing up the edge and surface error measurements as shown in Eq. 2.39:

$$T_{m-1,m} = \underset{T_{m-1,m}}{\arg\min}\ \Psi(S_{m-1}, S_m) \tag{2.38}$$

$$= \underset{T_{m-1,m}}{\arg\min}\ \sum_{i=1}^{N_s}[d^i_{s,m}(\hat{T}_{m-1,m})]^2 + \sum_{i=1}^{N_e}[d^i_{e,m}(\hat{T}_{m-1,m})]^2 \tag{2.39}$$

In Eq. 2.39, N_e and N_s represent the total number of edge and surface features, respectively. Since this is an iterative optimization approach, $\hat{T}$ represents the current estimate of the scan poses which gets updated during each iteration.

2.4 Camera Modeling

For most robotic applications that use camera data, a camera model is essential for extracting geometric information from the image data. For example, visual SLAM or SfM need to match 3D features to 2D image coordinates in order to build a map of the environment and localize relative to the map. To know how 3D world information would appear in a 2D image, a camera model is needed to project estimated 3D locations into the 2D image plane. This is referred to as a projection model (or forward projection model, or full model) [63]. If, for example, corresponding 2D features have been detected in images, and their estimated 3D location is required, then a ray projected outwards from each camera center through the corresponding 2D pixel location is needed to determine the intersecting point in the 3D world. The process of determining the ray extending outwards from some pixel is referred to as a back-projection model (or backward model) [63]. When this work refers to a camera model, it generally refers to the combination of the projection and back-projection models, which define the majority of the camera modeling needed for most computer vision tasks in robotics.

This section will describe a select few camera models commonly used in computer vision that are employed throughout this thesis. This section will describe the forward and back-projection models for the following three models: pinhole, Kannala Brandt (KB), and Double Sphere (DS). This section will also discuss which models apply to narrow versus wide (or fisheye) FOV lenses. For more information on these model definitions, including the definitions of the domains of the project and back-project functions, refer to [143].

Before defining the camera models, a commonly used coordinate frame convention will first be presented for all camera models in this work, as well as some other notation. Fig. 2.4 (retrieved from [20]) illustrates the pinhole camera model and the assumed coordinate systems. The origin of the main coordinate frame in Fig. 2.4 represents the optical center, which is the assumed location that all projections cross through. This coordinate frame is always oriented with the z coordinate pointed out of the camera, the y coordinate in the same direction as the v-axis of the image plane, and the x coordinate in the same direction as the u-axis of the image plane. Note that the image plane is the grayed-out square in Fig. 2.4, where the coordinates of a pixel on the image plane are defined by the 2D coordinate $U = [u, v]^T$. The point where the z-axis intersects with the image plane is

known as the principal point, and the coordinate frame of the image plane originates at the top left corner of the plane.

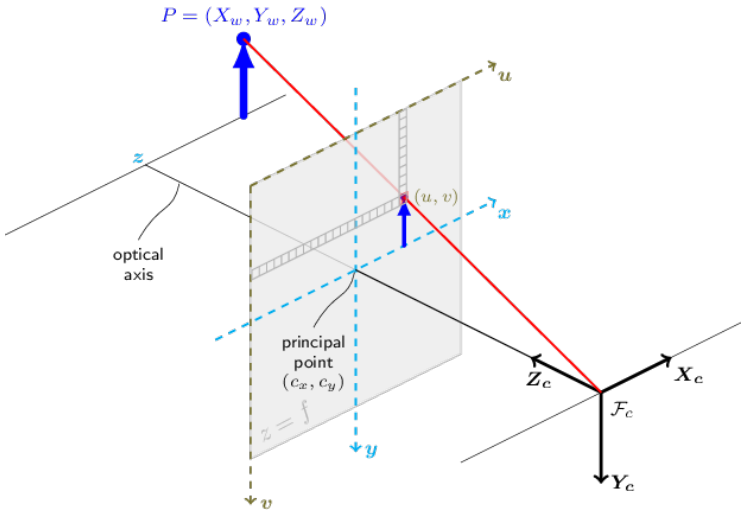


Figure 2.4: Illustration of a pinhole camera model with the coordinate frames displayed, taken from [20]

Eq. 2.40 shows a generic projection function:

$$U = \pi(\mathbf{P}, i) \tag{2.40}$$

$U = [u, v]^T \in \Theta \subset \mathbb{R}^2$ is a pixel coordinate where Θ is the image domain that points can be projected to, $\mathbf{P} = [x, y, z]^T \in \Omega \subset \mathbb{R}^3$ is a 3D point expressed in the camera frame where Ω is the set of 3D points that have valid projections, i is a vector of camera intrinsic parameters, π is the projection function. Eq. 2.41 shows the back projection function:

$$\mathbf{r} = \pi^{-1}(U, i) \tag{2.41}$$

π^{-1} is the back projection function, and $\mathbf{r} = [r_x, r_y, r_z]^T$ is the normalized back-projection ray that starts at the camera center and passes through some pixel coordinate.

2.4.1 Pinhole

The pinhole camera model is a common model used with low to moderate lens distortion, as lenses with FOVs greater than 120 deg begin to degrade with the pinhole model [143]. Eq. 2.42 presents the projection function for the pinhole camera model:

$$\pi(\mathbf{P}, i) = \begin{bmatrix} f_x \frac{x}{z} \\ f_y \frac{y}{z} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{2.42}$$

The intrinsics $i = [f_x, f_y, c_x, c_y]$ represent the focal length in the x and y directions, followed by the camera center in the x and y directions. Eq. 2.43 shows the back-projection function:

$$\begin{aligned} \pi^{-1}(U, i) &= \frac{1}{\sqrt{m_x^2 + m_y^2 + 1}} \begin{bmatrix} m_x \\ m_y \\ 1 \end{bmatrix} \\ m_x &= \frac{u - c_x}{f_x} \\ m_y &= \frac{v - c_y}{f_y} \end{aligned} \tag{2.43}$$

For convenience, the pinhole camera model is often commonly presented as shown in Eq. 2.44, where K (expanded in Eq. 2.45) is known as the intrinsics matrix, and s is a scaling constant.

$$s\ U = \mathbf{K}\ \mathbf{P} \tag{2.44}$$

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.45}$$

To account for low to moderate lens distortion, the pinhole camera model is also often accompanied by a distortion model. A distortion model that is commonly combined with the pinhole model is the Brown's distortion model [21], also known as the rad-tan model, or the plumb-bob model. The Brown's distortion model tries to model the distortion by a combination of radial distortion, and tangential distortion. The radial distortion is modeled according to Eq. 2.46:

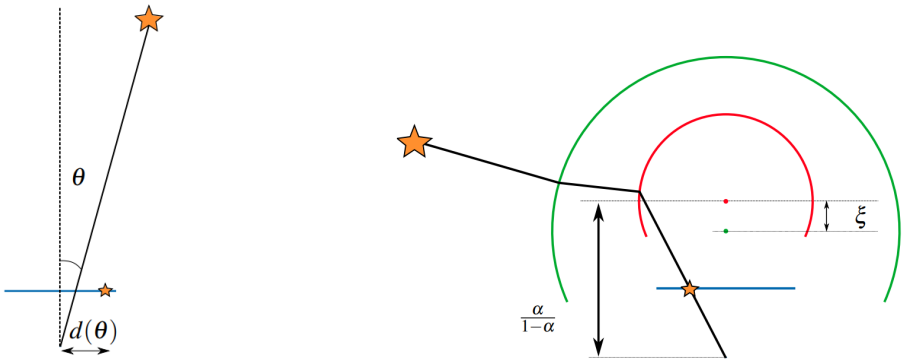
$$
\begin{aligned}
x_{\text{distorted}} &= x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\
y_{\text{distorted}} &= y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\
r^2 &= x^2 + y^2
\end{aligned}
\tag{2.46}
$$

The distortion coefficients are k_1, k_2, k_3, p_1, p_2 and these are considered part of the overall camera model parameters to be calibrated for each lens. The tangential distortion in the Brown's distortion model is defined by Eq. 2.46:

$$
\begin{aligned}
x_{\text{distorted}} &= x + \left[2p_1 xy + p_2\left(r^2 + 2x^2\right)\right] \\
y_{\text{distorted}} &= y + \left[p_1\left(r^2 + 2y^2\right) + 2p_2 xy\right]
\end{aligned}
\tag{2.47}
$$

2.4.2 Kannala Brandt

The Kannala Brandt (KB) model was created to be a generic camera model to model narrow, moderate, and wide FOV lenses [63]. Fig. 2.5a illustrates the KB model. As described in [63], the displacement of the projection from the optical center is proportional to $d(\theta)$, where $d(\theta)$ is a polynomial function of the angle between the point and the camera z-axis (shown as θ here). Since the KB model can be modeled using a polynomial of different degrees, the number of intrinsic parameters depends on the implementation.



(a) Kannala Brandt (b) Double Sphere

Figure 2.5: Camera model illustrations, retrieved from [63]

Here KB is modeled with eight intrinsics parameters $i = [f_x, f_y, c_x, c_y, k_1, k_2, k_3, k_4]$. The projection model is defined as:

$$
\begin{aligned}
\pi(P,i) &= \begin{bmatrix} f_x d(\theta) & \frac{x}{r} \\ f_y d(\theta) & \frac{y}{r} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \\
r &= \sqrt{x^2 + y^2} \\
\theta &= \operatorname{atan} 2(r, z) \\
d(\boldsymbol{\theta}) &= \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9
\end{aligned}
\tag{2.48}
$$

The back-projection function for the KB model are defined as:

$$
\begin{aligned}
\pi^{-1}(U,i) &= \begin{bmatrix} \sin(\theta^*)\frac{m_x}{r_x} \\ \sin(\theta^*)\frac{m_y}{r_u} \\ \cos(\theta^*) \end{bmatrix} \\
m_x &= \frac{u - c_x}{f_x} \\
m_y &= \frac{v - c_y}{f_y} \\
r_u &= \sqrt{m_x^2 + m_y^2} \\
\theta^* &= d^{-1}(r_u),
\end{aligned}
\tag{2.49}
$$

The KB model is also sometimes used as a distortion model for the pinhole model (e.g., the equidistant distortion model in Kalibr [18], or the fisheye model in OpenCV [20]). Note that the 9th-degree polynomial must be solved to determine the back-projection function, which can be computationally expensive.

2.4.3 Double Sphere

The Double Sphere (DS) camera model was explicitly created for better approximating wide FOV (fisheye) lenses while also providing a closed-form back-projection function. Fig. 2.5b illustrates the DS model. As described in [63], the point is initially projected onto the first sphere (green) and then onto the second (red) which is shifted by ξ. The point is then projected onto the image plane which is shifted by $\alpha/(1-\alpha)$. The closed-form back-projection function allows for faster computation when back-projecting pixels, unlike other wide FOV models such as the KB, which requires you to solve a high-degree polynomial function.

Eq. 2.50 describes the projection function for the DS model, and Eq. 2.51 describes the back-projection function.

$$
\begin{aligned}
\pi(P,i) &= \begin{bmatrix} f_x \frac{x}{\alpha d_2 + (1-\alpha)(\xi d_1 + z)} \\ f_y \frac{y}{\alpha d_2 + (1-\alpha)(\xi d_1 + z)} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \\
d_1 &= \sqrt{x^2 + y^2 + z^2} \\
d_2 &= \sqrt{x^2 + y^2 + (\xi d_1 + z)^2}
\end{aligned}
\tag{2.50}
$$

$$
\begin{aligned}
\pi^{-1}(\mathbf{u}, \mathbf{i}) &= \frac{m_z \xi + \sqrt{m_z^2 + (1 - \xi^2)\, r^2}}{m_z^2 + r^2} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \xi \end{bmatrix} \\
m_x &= \frac{u - c_x}{f_x} \\
m_y &= \frac{v - c_y}{f_y} \\
r^2 &= m_x^2 + m_y^2 \\
m_z &= \frac{1 - \alpha^2 r^2}{\alpha \sqrt{1 - (2\alpha - 1) r^2} + 1 - \alpha}
\end{aligned}
\tag{2.51}
$$

2.5 Sensor Calibration

Sensor calibration in robotics is generally divided into two categories: intrinsic and extrinsic. Intrinsic calibration refers to determining a particular sensor's intrinsic parameters (e.g., camera focal length, internal noise parameters of an IMU, etc.). These parameters are generally used to model how real-world quantities interact with the sensor. For example, camera intrinsics can be used to model how light interacts with a lens and where a light ray is expected to contact the sensor pad. IMU intrinsics such as acceleration bias, for example, can be used to estimate the true motion that a system underwent, given the output measurements of the sensor. Procedures for calculating intrinsic parameters vary drastically based on the type of sensor.

Extrinsic calibration refers to the process of determining the physical locations of one sensor relative to another. The main purpose of extrinsic calibration is to be able to fuse data collected from different sensors by enabling the data to be transferred into common reference frames. For extrinsic calibration, the full 6DOF pose is needed (i.e., rotation and translation) of each sensor on a robotic system relative to a common frame (often referred

to as the baselink frame). A sensor pose is usually described relative to the sensor's center (e.g., the optical center of a camera or the laser center of a lidar).

Many extrinsic calibration approaches are intended to calibrate one sensor relative to another. For a multi-sensor platform, these types of calibration techniques require a calibration tree to be generated, allowing data to be transferred between all sensor frames on the platform. This tree consists of a chain of rigid transformations where each sensor center can be determined relative to the reference frame by multiplying together a portion of the transformations in this tree. Extrinsic calibration is primarily done after intrinsic calibration since those parameters are usually required for calibrating the extrinsics.

In the following few sections, intrinsic calibration will be described for the sensors that are used in this thesis. This includes a description of the parameters to be estimated and a high-level explanation of common calibration methods used in practice to estimate these parameters. Since extrinsic calibration is discussed in detail in Chapter 4, this section will be restricted to intrinsic calibration only.

2.5.1 Camera Intrinsic Calibration

Camera intrinsic parameters tend to vary based on the camera model used, however, most models contain a focal length and optical center (in both x and y axes) as well as some number of distortion parameters, as described in 2.4. To calibrate these intrinsic parameters, methods generally use a calibration target with known dimensions of features on the target and try to detect the target features in the images. If features on a target can be detected in the image, the model parameters can be optimized by reducing the overall error between the world coordinates of the target features projected to the image plane using the projection model and the detected pixel coordinates in the image. For example, the popular OpenCV calibration tool uses a checkerboard calibration target where the squares have known dimensions that can be used to determine the true world coordinates, and corners are detected in the image using classical corner detectors [20]. In all these methods, multiple measurements of targets should be collected to ensure full coverage of the sensor model's projection function domain has been achieved.

2.5.2 Lidar Intrinsic Calibration

There are different types of lidars (e.g., solid-state or mechanical) that are associated with different intrinsic parameters. This thesis focuses on mechanical lidars, therefore discussion will be restricted to this type. A multi-beam mechanical lidar has multiple laser beams that

rotate as they emit light or have an internal mirror that rotates, which produces the same effect as a rotating beam. These devices produce range and bearing measurements, where one range measurement is returned with two bearing measurements (elevation and azimuth) in order to describe the 3D point. Fig. 2.6 depicts a typical multi-beam mechanical lidar, where one beam is shown measuring some point, P. In this figure, θ represents the angle for one specific laser beam, while β represents the rotational angle of any of the rotating beams. Given this notation, the lidar model can be written as shown in Eq. 2.52.

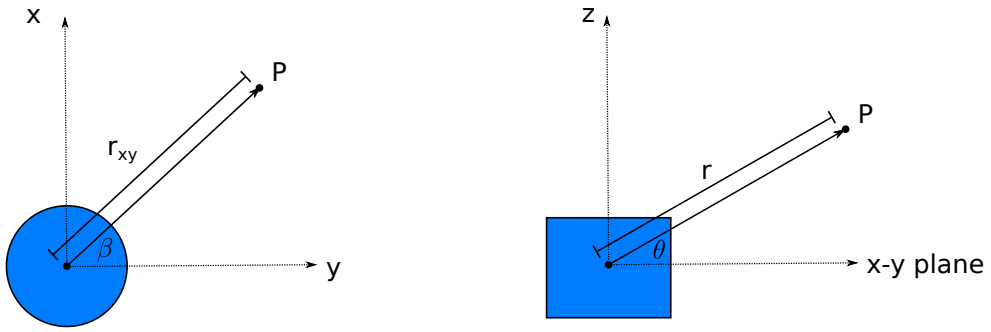


Figure 2.6: Diagram of typical multi-beam mechanical lidar systems, where the left image is the planar view and the right image is a horizontal view of the lidar.

$$\begin{aligned} P_z &= r \sin\theta \\ r_{xy} &= r \cos\theta \\ P_x &= r_{xy} \sin\beta \\ P_y &= r_{xy} \cos\beta \end{aligned} \tag{2.52}$$

The lidar model in Eq. 2.52 has no intrinsic parameters as r is measured, θ is given for each beam by the manufacturer, and β is also given for each measurement from the manufacturer. However, a lidar model with intrinsic parameters can be generated to take into account range and bearing biases, as well as offsets in the optical centers. These biases or offsets can be a result of manufacturing errors, such as timing errors in the range measurements or imperfections in the rotation measurements. Different methods exist for attempting to model these biases and fitting measurements to these models. This thesis does not calibrate the intrinsic of the lidars so the details will be omitted. However, examples include [103], who use a model with five intrinsic parameters, and [14], who use a model with 10 intrinsic parameters for each beam.

2.5.3 IMU Intrinsic Calibration

An IMU is typically made up of three sensors: (1) an accelerometer that measures acceleration in all three axes ($a = [a_x, a_y, a_z]$), (2) a gyroscope that measures rate of rotation ($\omega = [\omega_x, \omega_y, \omega_z]$), and a magnetometer that measures the absolute heading. Absolute heading is not commonly used in SLAM; therefore, this discussion will be limited to accelerometer and gyroscope measurements. A common way to model an IMU is to use an additive noise term which varies rapidly, and a slow varying bias (or random walk) term [45]. Eq. 2.53 shows the standard discrete-time IMU model [45].

$$\begin{aligned} \hat{a}_k &= \mathcal{R}_{\mathbf{IW}} \ (a(t_k) - g_w) + b_a(t_k) + n_{a_k} \\ \hat{\omega}_k &= \mathcal{R}_{\mathbf{IW}} \ \omega(t_k) + b_\omega(t_k) + n_{\omega_k} \end{aligned} \tag{2.53}$$

In Eq. 2.53, $\hat{a}_k$ is the k^{th} acceleration measurement, $\hat{\omega}_k$ is the k^{th} rotational measurement, whereas $a(t_k)$ and $\omega(t_k)$ are the true acceleration and rotation rates at the time of the k^{th} measurement, expressed in the world frame. Furthermore, $\mathcal{R}_{\mathbf{IW}}$ is the rotation matrix that converts points from the world frame to the IMU frame, and g_w is the gravity vector in the world frame. The intrinsic parameters are therefore $b_a(t_k) \in \mathbb{R}^3, b_\omega(t_k) \in \mathbb{R}^3, n_{a_k} \in \mathbb{R}^3$ and $n_{\omega_k} \in \mathbb{R}^3$ which represent the acceleration bias at time of measurement k, the gyroscope bias at time of measurement k, the noise of acceleration measurement k, and the noise of the gyroscope measurement k, respectively. The slow varying biases, $b_a(t_k), b_\omega(t_k)$, are modeled as zero-mean white Gaussian processes [45].

IMU intrinsic parameters are often provided by the manufacturers, however, they can be re-calibrated for added precision which is important when using IMU data for SLAM. The calibration procedure used in this work involves collecting IMU measurements when the device is stationary and then calculating bias and noise parameters from the Allan Standard Deviation. For more details on this calibration process, refer to [45].

Chapter 3

System Design

A wide range of custom or off-the-shelf mobile robotics platforms has been used for inspection, such as climbing robots, ground robots, legged robots, aerial robots, surface vessels, and underwater robots [4]. The majority of these platforms are simply used as a means to collect sensor data from hard-to-access locations, where data is analyzed on its own without any regard for where the data was collected on the structure being inspected. This represents an important part of inspection, however, without having a map to localize elements or detected defects in 3D, the inspection picture is incomplete. An inspector would have to manually summarize the inspection information and present it in a digestible fashion, which usually requires the results to be translated to a drawing or Computer Aided Design (CAD) model. This manual task can be challenging with the large amounts of data such robotic inspection systems produce. Furthermore, to assess the rate of deterioration of a structure, an inspector would also have to manually compare all inspection measurements from one inspection to the previous inspections, which may be difficult depending on how previous inspectors summarized the data.

To avoid such problems with localization and tracking of inspection measurements, some work has been done to create maps of the structure while collecting data. This simplifies the task of measuring change in data and presenting the data in a usable way to either the asset owners or contractors doing repairs. For example, having defects labeled in a georeferenced 3D map would allow asset owners to easily visualize and quantify defect changes over time. Many of 3D mapping approaches rely on image-based 3D mapping such as SfM, however, these methods have been known to be unreliable and inaccurate [66] for mapping large scale areas with complex 3D shapes. To achieve precise and robust 3D localization and mapping, multi-sensor SLAM is the preferred approach [61, 130, 162]. To date, very few inspection systems have implemented multi-sensor SLAM, and those that

have, have not combined lidar, camera, and inertial measurements in a tightly-coupled way for optimal SLAM performance in the challenging degenerate environments common to inspections [130, 162]. Furthermore, no existing literature discusses sensor selection and layout for joint SLAM performance and inspection data collection.

This chapter presents the first multi-sensor SLAM-centric inspection system that optimizes sensor layout not only for SLAM in challenging environments but also for accurate inspection data collection. This chapter is restricted to the hardware design of the system to best enable the SLAM and inspection software development that will be discussed in later chapters in this thesis. The specific contributions of this chapter are as follows:

- A new inspection system is designed whose functionality is platform-independent, allowing this system to be mounted on any mobile robot, eliminating all accessibility problems in data collection.
- An inspection system is presented which demonstrates specific sensor configuration requirements for precise and robust SLAM while simultaneously collecting inspection data that can be customized for the inspection task.
- A modular synchronization strategy is also presented which enables sensor data from all modalities to be fused for SLAM and inspection purposes.

As motivation, Fig. 3.1 shows the proposed inspection platform in action in the field performing infrastructure inspection. The left image shows the UGV platform performing an inspection of a parking garage, where the right image shows the same UGV performing inspection on a bridge. This Chapter will discuss the design leading to the creation of the UGV in Fig. 3.1, whereas Chapter 5 will show the inspection methodology and results.

3.1 Related Works

Section 1.2 discussed methods found in the literature for augmenting infrastructure inspection using robotics. In order to obtain repeatable and precise inspection measurements, a combination of various robotic sensors is needed, including cameras for versatile defect detection and range sensors for accurate scale estimation and dense 3D data. This section will outline previously published robotic inspection devices that use robotic sensors (e.g., camera, lidar, IMU, GNSS, wheel encoders) for creating 3D maps and using these for visual inspection of infrastructure, referred to in this work as SLAM for inspection.



Figure 3.1: Inspection platforms at dataset site locations: Duke Street Parking Garage (left), Park Street Bridge (right)

The majority of the works that use SLAM for inspection are implemented on UGVs or UAVs. In the case of UGVs that use SLAM for inspection, a large proportion of these only focus on creating 2D maps for inspection of 2D surfaces such as bridge decks. The ROCIM system presented by Lim et al. [82] is a wheeled robot for bridge deck inspection, which uses a 2D lidar with a Monte Carlo SLAM approach. The RABIT robot by La et al. [75, 76] also presents a wheeled robot for bridge deck inspection that is also equipped with a variety of non-destructive sensors such as electrical resistivity, impact-echo, and ultrasonic surface waves. RABIT relies on an EKF SLAM approach using wheel encoders, IMU, and high-performance GNSS. This system is fully autonomous and can automatically generate corrosion, delamination, and concrete elastic modulus maps of a bridge deck. Gibb et al. [47] present another wheeled robot with ground penetrating radar for bridge deck inspection. Gibb et al. also use an EKF SLAM approach for outdoor settings, while an EKF-based Visual Inertial Odometry (VIO) approach is used for indoor SLAM. In all the above cases, the solutions are limited to inspecting horizontal surfaces such as decks since they only perform 2D SLAM. Furthermore, the majority of these methods rely on having precise GNSS signal, which is not always the case with many inspections, including, for example, the underside of bridge decks, bridge supports or parking structures.

Kim et at. [70, 147] presented a robot with multiple rotating 2D lidars, Infrared (IR) cameras, and a RGB camera on a stop-and-go scanning UGV which generates full 3D maps

with information from the cameras overlaid onto the maps. These papers refer to the task of inspection, however, they do not demonstrate the use of this robot for inspection. The stop-and-go nature of this SLAM solution also limits the approach to stable platforms such as a UGV (i.e., not a USV or UAV) since the platform must be stationary during the scan time. This also presents challenges with getting full coverage of a structure in a reasonable amount of time compared to continuous scanning platforms as it would take an infinite amount of scan locations to get the same amount of coverage.

The work by Peel, H. [114] presents a UGV capable of generating a 3D map for inspection of bridge bearings. In this work, different SLAM methods were tested, including 2D lidar SLAM and 2D lidar localization from pre-built 3D maps, as well as visual SLAM. Again, the 2D SLAM tests limit the inspection to a device navigating flat terrain, which significantly constrains the type of inspections such a device could be used on. The visual SLAM solution does provide a 3D pose estimate, however, as discussed previously, relying entirely on cameras for SLAM is not robust enough for many inspection environments with low texture and poor lighting.

A more relevant set of works that uses SLAM for inspection has been implemented on UAVs. Since UAVs are moving in 3D, SLAM involving UAVs must track the full 6DOF pose and, therefore can generate 3D maps of structures. Yang et al. [150] present the CityFlyer UAV. The CityFlyer is equipped with a structured light RGBD camera. This work proposes an inspection pipeline spanning from 3D SLAM to defect detection and projection to the 3D map. Unfortunately, structured light-based RGBD cameras are limited to close-range measurements (within a few meters), and the IR light they depend on for depth estimation is highly susceptible to corruption by sunlight, making this sensor inoperable in direct sunlight which is common to inspection environments.

Feng et al. [40] present a similar solution that relies on solid-state lidar and an RGB camera, which can be used outdoors and at farther ranges than RGBD sensors. This work performs SLAM with R^3LIVE [83] and is therefore dependent on their SLAM and meshing techniques, which do not allow the flexibility of adding additional sensor types such as lidars or cameras pointing in other directions. Both [150] and [40] also require the cameras/lidars to be continuously pointing in the direction of high environment structure to ensure stable SLAM tracking, in contrary to other methods that use cameras/lidars with larger fields of view such as mechanically rotating lidars. Furthermore, these methods still depend highly on tracking visual features at all times, which will be limiting in inspection environments such as bridges with difficult lighting conditions and textureless concrete with vegetation and water in the backgrounds.

Jung et al. [62, 61] also introduce a relevant inspection system. In this work, a UAV

is used with a real-time kinematic (RTK) GNSS, 3D lidar, monocular SLAM camera, and a high resolution Digital Single Lens Reflex (DSLR) camera. They summarize many of the existing SLAM solutions and conclude that vision-only SLAM methods are not robust enough for inspection, and lidar-only methods are not optimized for practical inspection, and do not allow for modifying either hardware or software to suit the needs of inspection. Therefore, they build a custom SLAM solution that uses both lidar and visual SLAM. They conclude that more robust SLAM is still needed to perform inspection reliably. However, the data on their UAV is stored separately and therefore is neither software nor hardware synchronized. The authors acknowledge issues that are caused by unsynchronized data, however, there is no mention of the fact that tightly coupling IMU and lidar and/or image data requires precise time synchronization and therefore they are unable to validate the potential of tightly coupled SLAM methods. This recent work perfectly outlines the challenges with using SLAM for inspection and highlights the gap in the current literature, that is, that there is no published work that describes how to create a general purpose SLAM-centric inspection system.

The above studies also neglect to compare sensor types and specifications which led to the design of their platforms. The most notable work that discusses sensor and platform selection and how that affects inspection tasks is [6]. In this work, they present a review of all UAV-based infrastructure inspection works and summarize how different sensors are useful for different inspection tasks. They also discuss what platform specifications are helpful for navigation around bridges. However, this study does not explicitly discuss how to select sensors to enable precise and reliable SLAM.

To summarize, of all the inspection literature that uses multi-sensor SLAM for 3D map generation, three important areas of research have not received the attention they deserve and hence need further study. First, literature is lacking in platform-independent systems which can be ported to any robotic or non-robotic system that suits the needs of an inspection. Second, the issues related to sensor selection and layout have not been studied thoroughly in this context, specifically how these choices affect SLAM and inspection results. Third, hardware integration aspects, including time synchronization and calibration to ensure proper data fusion for SLAM and downstream inspection tasks, have not received attention in this context. This chapter presents an engineered system that addresses each of these gaps.

3.2 Mobility

Section 3.1 presents relevant works that have utilized various mobile robotic platforms for inspection. The most common type of platforms are UGVs (including wheeled and legged robots) [4], however, using such a system significantly limits the number of structures that can be inspected. For example, many bridges span busy roadways or water bodies. UGVs are also limited to low-spanning structures if the sensors are designed for short-range measurements. USVs are an alternative to UGVs and they allow inspections to be performed over water bodies, however, these usually need to be supplemented with some other inspection device for accessing the areas not over water. Climbing robots are another alternative which can access structures spanning water or other areas non-accessible by ground. Unfortunately, these systems are often limited to climbing certain types of elements (e.g., cables, flat walls, etc.), and they are usually more time-consuming to get full coverage of a structure. Many claim that UAVs are the most versatile for the inspection task since they have no restrictions on accessibility [4]. However, using UAVs for inspection presents many other challenges that render their use impractical or impossible for certain applications. One of the main challenges with UAVs is that they have low payload capabilities compared to their counterpart platforms. This makes it difficult to equip the device with all the sensors that may be needed for the inspection task. Furthermore, equipping a UAV with many inspection sensors can reduce the flight time to an impractical time due to power consumption limitations. The other main challenge with using UAVs for inspection is related to flight permissions. Certain inspections need to be performed in areas where UAVs cannot be flown, such as near airfields, near high vehicle traffic areas, or in nuclear power facilities where radiative particulate matter is of concern.

Evidently, no single mobile platform is suitable for all possible infrastructure inspections. To avoid limiting the scope to the applicable inspection tasks based on a specific platform selection, this thesis attempts to design a platform-agnostic system. In other words, the proposed system is designed to be easily transferrable to any of the listed mobile platforms or any other means of mobility not discussed. In fact, early research conducted in this work involved discussions with many inspection practitioners, which revealed that a handheld inspection device can be most suitable for many inspections as it would least impact an inspector's normal routine. A handheld device would also provide the lowest learning curve as they are not required to learn to navigate a robotic platform. This type of device may provide the lower entry barrier needed for inspectors to adopt such a technology.

The main idea behind the proposed platform-agnostic inspection system is to design a self-sufficient module that performs all the core tasks required for the proposed inspec-

tion approach (discussed in detail in Chapter 5. The proposed SLAM-centric inspection system requires at least the following base sensors: 3D lidar, SLAM camera, IMU, and a high-resolution camera. To achieve reliable SLAM in any inspection environment, the combination of lidar, camera, and IMU data is essential (see Chapter 1 for more detail). Each of these sensors will be discussed individually in Section 3.3, including their criteria, the reason behind the criteria, required configurations, and the specific sensor models used in this work. The last criterion for the inspection system to be portable to any mobile platform is to ensure the mass of the base system is low enough for any mobile platform. The four essential sensors, including the Velodyne Puck Lite lidar, two cameras with lenses, the IMU and the micro-controller needed to synchronize the sensors (see Section 3.4) have a combined mass of approximately 958 g. The mass breakdown for each sensor on this system is shown in Table 3.1. Of the typical mobile robotic platforms used for inspection, UAVs tend to have the smallest payloads. Taking, for example, a DJI Matrice 200, which is on the lower end of payloads for customizable UAVs, the max payload is 2.34 kg. This leaves 1.38 kg for an onboard computer and cables which is more than reasonable for a computer capable of collecting the sensor data. Furthermore, other researchers such as Jung et al. [61] have demonstrated that UAVs can be used for inspection with larger sensors, as they use a DSLR camera instead of the proposed lightweight high-resolution camera. This weight range is also reasonable to be used on a handheld device for inspections, provided the computer can be carried in a backpack or something equivalent, which is demonstrated in this work.

Given a base module with the ability to precisely map and localize itself within any inspection environment, an inspection module can then be added in any desired configuration to suit the needs of the specific inspection. For example, for bridge deck inspection, a large majority of structural defects can be found on the underside of the bridge deck. To properly inspect the underside of bridge decks, it may be desirable to mount a vertical lidar, vertical high-resolution RGB camera, and a vertical infrared camera. Fig. 3.2 shows only the base module, and Fig. 3.3 shows two example robotic inspection systems, plus the base module combined with a backpack computer for a handheld inspection solution. Fig. 3.3b shows the base module mounted to a UGV with the vertical inspection module described previously, and Fig. 3.3c[1] shows the same exact sensor suit attached to a USV with the addition of a sonar for subsurface mapping. Note that in both the handheld (Fig. 3.3a) and the UGV (Fig. 3.3b) systems, the base module has an additional camera with a wide FOV lens facing backward for additional images that are not required for the proposed inspection pipeline. The goal of the backward-facing camera is to ensure full 360 degree

[1]The sonar integration on this USV was done by fellow PhD students Alexander Thoms and Gabriel Earle

FOV coverage with both cameras for full coloring of the lidar maps. For more information on map coloring and how to obtain inspection results from these sensor modules, refer to Chapter 5. For more information on calibrating such a system, refer to Chapter 4.

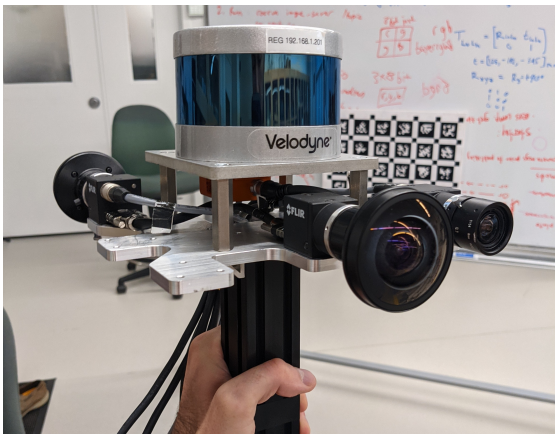
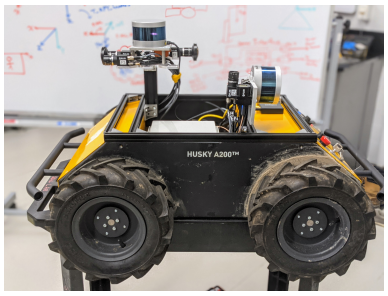


Figure 3.2: Base module for inspection systems



(a) Handheld with base module and backpack computer



(b) UGV with base module and vertical inspection module



(c) USV with base module, vertical inspection module, and sonar

Figure 3.3: Example inspection systems

3.3 Sensor Configuration

This section will discuss the sensors used for both the base unit and an example supplemental vertical inspection unit. In both cases, the selected sensors will be described in more

detail, including the sensor criteria, required specifications, and configuration for best inspection results. This section may also serve as a tutorial for other researchers attempting to build a similar system. Sensor selection is critical when building an inspection system, and very little published work can be found that discusses how to select these devices.

The proposed sensor configuration is composed of two modules: a base module and an inspection module. Fig. 3.3 shows the three proposed systems that each have the base module pointed in the direction of travel. This is to ensure all essential tasks for a SLAM-centric inspection system, such as localization and mapping, can be done successfully. Fig. 3.3 also shows a UGV and USV platform have an additional upward-facing inspection module to capture additional detail specific to the inspection type, which in this case is the inspection of the underside of bridges and parking structures. The design of each module will be discussed in more detail next.

3.3.1 Base Module

As highlighted in Section 3.2, the base module has been designed to perform the essential tasks required for any mobile SLAM-centric inspection system following the inspection pipeline presented in this thesis. These essential tasks can be broken down into three categories:

1. Precise and robust SLAM in inspection environments
2. Dense 3D map generation
3. High resolution RGB image collection

Let's first consider task 1. Inspection environments (e.g., bridges, parking garages, etc.) often present challenging SLAM situations as a result of poor or rapidly changing lighting conditions (e.g., looking up towards the sun, or going into shaded areas under a bridge), poor visual texture (e.g., looking at a large texture-less concrete wall), and poor physical structure (e.g., in a tunnel or between bridge abutment walls). To achieve precise and reliable SLAM, SLAM research has shown that combining lidar, camera, and IMU data in a tightly coupled fashion can provide the most reliable and precise SLAM results [130, 162]. For such SLAM systems, if visual measurements become degraded due to a lack of features, the SLAM algorithms can rely more heavily on lidar and IMU measurements. Similarly, when encountering poor physical structure where lidar measurements become degraded, the SLAM algorithms can rely more on IMU and camera measurements.

The next critical task is that of dense 3D map generation. For defects to be measured and localized on the structure, full 3D maps of the structure must be generated, and these maps must be dense enough to adequately measure defects that were detected in images. Since the proposed base module design already has a 3D lidar, this sensor can be used to generate dense 3D lidar maps from the lidar data and the final trajectory calculated from SLAM. The more points per second measured by the lidar, the more dense the 3D map can be.

The final critical task of the base module is the collection of high-resolution RGB images. All visual inspections require visual information to detect defects of interest. Some defects (e.g., cracks) can be quite small in nature, therefore requiring a camera that collects sufficiently high-resolution image data. Since visual SLAM is often implemented with low-resolution images, a separate high-resolution camera is needed. Furthermore, camera data from SLAM must be high rate to enable high rate pose estimation. On the other hand, for inspection images, a lower rate of data collection is sufficient to still cover the entire structure's area seen by the device. This is similar to the Augmented Reality and Virtual Reality (ARVR) space where SLAM cameras are typically separate from cameras required for tasks like image pass-through, as is the case with the Meta Quest 3, or scene understanding tasks such as is the case with the Aria AR research device [134].

Table 3.1 shows the devices selected to accomplish the above mentioned tasks. The following section will go through each sensor and describe all requirements.

For addressing task 1, a lidar, IMU and SLAM camera are selected to achieve optimal SLAM performance while also ensuring the lidar is sufficient for task 2. For task 3, a separate inspection camera is added, which is not used for SLAM.

For the SLAM sensor selection, SOTA SLAM implementations are compared, and sensors are selected that conform to the minimal specifications used in these SOTA implementations. Table 3.2 shows SOTA Lidar Inertial Odometry (LIO), VIO, and Lidar Visual Inertial Odometry (LVIO) implementations, and their selected sensor details. Since these implementations are generally designed to be versatile to different sensor specifications, this section presents the sensor specifications that were used for testing in their respective research papers. Some of these papers have tested with different sensors, therefore the details of all datasets presented in their publications is shown. Others have neglected to specify all the details of the sensors used, therefore those details are marked as unavailable (NA).

Lidar sensors are not only critical to ensure robust SLAM (some could argue that certain mapping environments have sufficient visual saliency not to require lidar measurements), but lidar is also indispensable for the remainder of the proposed inspection pipeline

Table 3.1: Base module sensors

Sensor Type	Model	Mass (g)	Relevant Specifications
Lidar	Velodyne VLP16 Lite	590	Vertical FOV: 30°, 300,000 points per second, 100 m range
SLAM Camera	Flir BlackflyS	36	3.2 MP resolution, RGB, 55 frames per second
SLAM Lens	Fujinon	160	2.7 mm focal length, 2/3 inch image circle, C mount, FOV: 185°x 140°
Inspection Camera	Flir BlackflyS	36	12.3 MP resolution, RGB, 23 frames per second
Inspection Lens	Flir Tamron	44	8 mm focal length, 1/1.8 inch image circle, C mount, FOV: 50.8°x 38.6°
IMU	Xsens MTi-30	72	Frequency: 400 Hz, Data types: XYZ acceleration, xyz angular velocity, absolute heading, east-north-up orientation
Micro-controller	Teensy 3.6	20	

downstream from SLAM. Since the proposed pipeline requires a dense 3D map of the environment, the only way to guarantee consistently high density in the final map for all environmental features is to use a lidar. This is because lidar does not require high visual texture to produce dense maps, as is the case with image-based reconstruction. This work selects a 3D lidar to ensure the lidar measurements for SLAM have visibility over all 6DOF of the sensor motion. A 2D lidar sensor could be used as an alternative should the sensor be mechanically rotated, such as Kim et at. [70, 147] or Zhang et al. [157]. A rotating beam-based lidar design is also desirable over a solid state lidar for two main reasons: 1) full 360°FOV coverage enables better location tracking, and 2) multibeam 3D lidars return more points per second allowing denser maps to be constructed in the same amount of time. Finally, this work proposes to avoid RGBD sensors as these are usually restricted to short-range measurements and do not work well in direct sunlight, which both present significant issues for outdoor infrastructure inspections. The lidar used in this work is a Velodyne VLP16 Lite. This has 16 beams that rotate a full 360 degrees, cover a vertical

Table 3.2: Popular visual and/or lidar SLAM hardware selection details

SLAM Method	Lidar	Camera Rate/Resolution	IMU Rate
LVI-SAM [130]	Velodyne VLP16	30 Hz / 0.4 MP	500 Hz
SuperOdometry [162]	Velodyne VLP16	N/A / N/A	N/A
UMMLT LVIO [148]	Ouster OS1-64 & Velodyne VLP16	15 Hz / 1.6 MP	100 Hz
VLOAM [159]	Velodyne VLP16 & Velodyne HDL32E	60 Hz / 0.4 MP	400 Hz
VINS-MONO [117]	N/A	30 Hz / 0.4 MP & 20 Hz / 0.4 MP	100 Hz & 200 Hz
LIO-SAM [129]	Velodyne VLP16	N/A / N/A	500 Hz

FOV of 30°, have a 100 m range, and collect approximately 300,000 points per second. This lidar was chosen because it is relatively low cost, and has a sufficient number of points and FOV for SLAM, as many recent popular SLAM implementations (e,g, [129, 130, 162]) have used the VLP16. For examples of lidar types selected for different SOTA SLAM methods, see Table 3.2. The higher the FOV and the more the number of points collected, the better the potential SLAM results; however, the selected lidar is lightweight (590 g) compared to other lidar which makes it a good candidate for building a lightweight sensor kit that may be used on a handheld device or UAV. It should be noted that when purchasing the equipment for this thesis work, the VLP16 was the most common 3D lidar used because it was relatively inexpensive and lightweight. Lidar technology has been progressing rapidly recently, specifically with the attention on autonomous vehicles, and therefore, at the time of writing this thesis, there are already options for higher-density data capture that remain lightweight and inexpensive. It is recommended to use higher-density lidar sensors if possible, given that the weight suits your platform, as this significantly increases performance for both SLAM and inspection tasks.

The selection of cameras for SLAM-centric visual inspection is a bit more intuitive, and camera technology is significantly more mature than lidar technology. For SLAM, literature shows that camera-based SLAM often performs well in degenerate lidar-based SLAM environments such as environments with low structure (e.g., outdoors in open fields or roadways). Second, the high data rate of modern cameras helps capture high-rate motion for platforms such as UAVs, USVs, or hand-held devices, and these high rate-pose estimates can be used for undistorting continuous lidar scans for optimal lidar-SLAM performance [130]. On the inspection side, images are a fundamental necessity for manually or automatically retrieving inspection results, as discussed in Chapter 5. Specifications

may differ for the two use cases; however, to keep weight low for mobility reasons discussed further in the next section, it is possible to select camera specifications that are well suited for both. As shown in Table 3.2, SLAM image rates typically span between 15 and 60 Hz, with the average being around 20-30 Hz, which is set as a minimum for the proposed camera requirements. In terms of resolution, LVI SLAM systems use somewhere in the range of 0.4 MP to 1.6 MP.

Lens selection is another important detail rarely discussed in inspection or SLAM literature. A detailed study of FOV selection for optimal visual SLAM was conducted by Zhang et al. [161]. In this work, they conclude that larger FOV lenses provide increased performance in indoor environments. The high FOV allows features to be tracked for a longer time and provides more overlap in consecutive frames for more robust feature tracking. On the other hand, for outdoor scenes where features are typically at greater distances, lower FOV lenses provide better results as the increased angular resolution drastically improves feature measurement and association. For infrastructure inspections, there is often a mix of feature ranges depending on the application. For example, the indoor inspection of parking garages or industrial facilities would benefit from the larger FOV lenses. Outdoor inspections such as bridges or building facades could benefit from more narrow FOV lenses. That being said, for the proposed sensor suit, the two selected cameras are: one low resolution with a high FOV (3.2 MP and 50.8°x 38.6°FOV), and one high-resolution camera with a low FOV (12.6 MP and 185°by 140°FOV). Since image resolution can be manually downsampled prior to running SLAM, this is appropriate for SLAM and inspection. For SLAM, the high FOV camera can be used when inspecting close-range infrastructure, and the low FOV camera can be used when inspecting longer-range infrastructure. For inspection, the high FOV cameras with low resolution can be used for generating colorized maps of a piece of infrastructure, where coverage area is more important than fine detail since the overall colored map resolution is limited by lidar density. For defect detection, the high-resolution camera can be used to ensure small defects can be reliably detected, such as cracks and exposed re-enforcing bars.

It is worth taking some time to analyze Ground Sampling Distance (GSD) for the selected cameras as a measure of the scale of defects that can be detected. Given the above sensor specification for Camera 1, i.e., 38.6 x 50.8 FOV with 3000 x 4095 pixel resolution, the vertical and horizontal GSD can be calculated for two cases: 1) a typical parking garage inspection where the soffit is being inspected from a ground vehicle, and 2) a typical highway overpass bridge inspection from a ground vehicle. To be conservative, this work will assume the ground vehicle's height to be 0 m above the ground. Given a minimum parking garage height in Ontario of 2.1 m, this will also be rounded to 3 m to be conservative. For highway bridges, given a minimum bridge clearance of 4.15 m, a

clearance of 5 m will be used. Table 3.3 summarizes the results for these two cases. The GSD for the parking garage case is 14.3 pixels/cm (vertical), whereas the GSD for the bridge case is 8.6 pixels/cm. This can be used to calculate the minimum detectable defect size. Taking, for example, a crack, if a minimum of 3 pixels per crack width is required to properly detect the crack, this means that cracks starting from 2.1 mm and 3.5 mm can be detected for the parking garage and bridge examples, respectively.

Table 3.3: Example GSD and defect detection sizes

Case	Distance to Surface (m)	Vertical GSD (pixels/cm)	Horizontal GSD (pixels/cm)	Min. Crack Width (mm)
Parking Garage	3	14.3	14.4	2.1
Highway Bridge	5	8.6	8.6	3.5

For IMU selection, the main selection criteria are the data rate and the type of data that is available. Table 3.2 shows that state-of-the-art SLAM methods use IMUs that output data in the 100 Hz to 500 Hz. LIO, VIO, and LVIO methods typically perform IMU pre-integration to estimate relative changes in pose. For IMU pre-integration, we need acceleration in all three axes (x,y,z) and rotation rates about all three axes (x,y,z). Absolute orientation measurements are also sometimes used, more commonly for initialization. Absolute orientation can be estimated from the magnetometer and accelerations by optimizing for the gravity direction, assuming net zero acceleration on average. Many IMUs provide an estimate of absolute orientation internally with their own estimator, which facilitates SLAM integration. This work selects the Xsens MTi-30, which outputs measurements at 400 Hz and provides accelerations and rotation rates about three axes while also providing an absolute orientation estimate.

It should also be noted that sensor selection was also dependent on time synchronization. Sensors need to be hardware synchronized, which also restricts sensor selection. The microcontroller selection was also based on time synchronization. This will be discussed further in Section 3.4.

3.3.2 Inspection Specific Modules

Shown in the research review in Section 3.1, a variety of inspection tasks have distinct requirements but can benefit from a standard SLAM-centric inspection approach. The previous section outlined the minimum sensor requirements needed to unlock the basic inspection pipeline presented in this thesis (i.e., precise & robust SLAM, dense, high-quality

3D maps, and high-quality localizable images). This section discusses how additional inspection-specific sensor types and layouts can be integrated into the proposed SLAM-centric system.

This section discusses how such targeted sensors and layouts can be applied to the proposed SLAM-centric system by an example. The following will present the additional sensors added to the UGV inspection system shown in the thesis, targeted at performing ground-based inspection of concrete bridges and parking garages, however, various extensions similar to the ones presented here can clearly be added in similar ways to suit different inspection needs. These use cases are also the use cases used throughout the remainder of this thesis, including Chapter 5 on the proposed inspection pipeline implementation.

For the case of concrete bridge and parking garage inspection, many of these inspections can be performed by a UGV, which allows the system to benefit largely from adding additional sensor payloads. The UGV shown in Fig. 3.3b shows the base sensor module presented in Section 3.3.1 with an additional upwards-facing sensor module containing three more sensors:

1. Velodyne VLP-16
2. Flir BlackflyS high resolution RGB camera
3. Flir ADK infrared camera

Fig. 3.4 shows two close-up images of this sensor module. These sensors are pointed upwards for bridge and parking garage inspections because the majority of structural components needing careful inspection are on the soffit of the structure—for example, structural slabs, girders and beams, column capitals, etc. Since the required sensors for SLAM need to be generally facing in the direction of motion to properly estimate motion, having additional sensors facing perpendicular directions can be extremely valuable, especially for these types of inspections. This section will discuss the sensors shown in Fig. 3.4.

Fig. 3.5 shows an example image of a parking garage using the horizontal camera (left) and the vertical camera (right). The horizontal camera, selected in part for precise SLAM results, has a lower resolution and a higher FOV. The vertical camera, selected primarily for inspection, has a higher resolution and a lower FOV for more detail with less distortion. Clearly, the defects are much more visible and less distorted, resulting in a higher ability to detect using ML and quantify accurately.

A vertically mounted Lidar is also included in the inspection module for similar reasons to that of the RGB camera. Since the SLAM-specific lidar needs to mostly point forward,



Figure 3.4: Close up of inspection specific sensors on the UGV platform

having a perpendicularly mounted lidar provides valuable new data. In addition to this obvious advantage, there is a secondary advantage that has a significant effect on the ability to generate useful inspection results. Because the lidar collects 3D data in a direction perpendicular to the overall motion, this acts more like a sweep-scanner, which provides consistent density in the resulting 3D maps, greatly improving their quality. Conversely, lidars facing the direction of motion will generate maps where the point density is very high around the height of the lidar and becomes much more sparse further from that region. In addition, when the lidar is mounted facing the direction of motion (i.e., horizontally in this case), you can also get an increased density of points along the beam lines. Fig. 3.6 illustrates these effects. The images on the left column of Fig. 3.6 show a 3D lidar map of a parking garage (same garage as in Fig. 3.5) generated using a horizontally mounted lidar, whereas the images in the right column of Fig. 3.6 show the resulting map from the same trajectory but using the vertical lidar. Clearly, if using these maps to measure defects, there would be a significantly higher accuracy when using the vertical lidar map. However, keeping the horizontal lidar for SLAM will allow for the best possible trajectory estimates. For more information on how these maps were generated and used for inspection, refer to Chapter 5.

The final inspection-specific sensor included on the UGV platform is an infrared camera. Infrared cameras have been shown to be useful for detecting subsurface issues in concrete. When the steel in reinforced concrete corrodes, it expands and creates a void inside the concrete. It is imperative to detect these defects at an early stage since they can accelerate the deterioration of the structure, and more critically, they can detach from the structure and cause damage or injuries to the public. These voids in the concrete disrupt the heat



Figure 3.5: Example images captured from the horizontal camera (left) vs. the vertical camera (right)

transfer through the materials, and this change in pattern can be detected with infrared cameras. Figure 3.7 shows example infrared images of a concrete bridge structure and the results of combining such data with a 3D map to estimate sizes. For more information on this, please refer to the work in [91] and refer to Chapter 5.

This new sensor configuration does not come without challenges. These sensors need to be synchronized and calibrated to properly fuse data from all sensors. Section 3.4 will go over the synchronization strategy, which is applied to all sensors, including the base sensors, and the inspection-specific additions presented in this section. Extrinsic calibration of the upward facing sensor module is, however, more challenging than calibrating the base sensors on their own. The reason is that these sensors have very little overlap in FOV with the sensors on the base platform. Most calibration techniques that provide repeatable high-quality calibrations rely on setting up fiducial targets that are detected in the FOV of at least two sensors at a time. Given a set of overlapping target measurements, the rigid transformations that define the extrinsics calibrations of the system can be estimated. Since there is very little overlap between the FOV of the sensors, a new extrinsic calibration strategy must be used, which is presented in Chapter 4

3.4 Synchronization

Precise time synchronization is critical to building a SLAM-centric inspection system. This is also an area that is seldom discussed in inspection literature, or even in SLAM

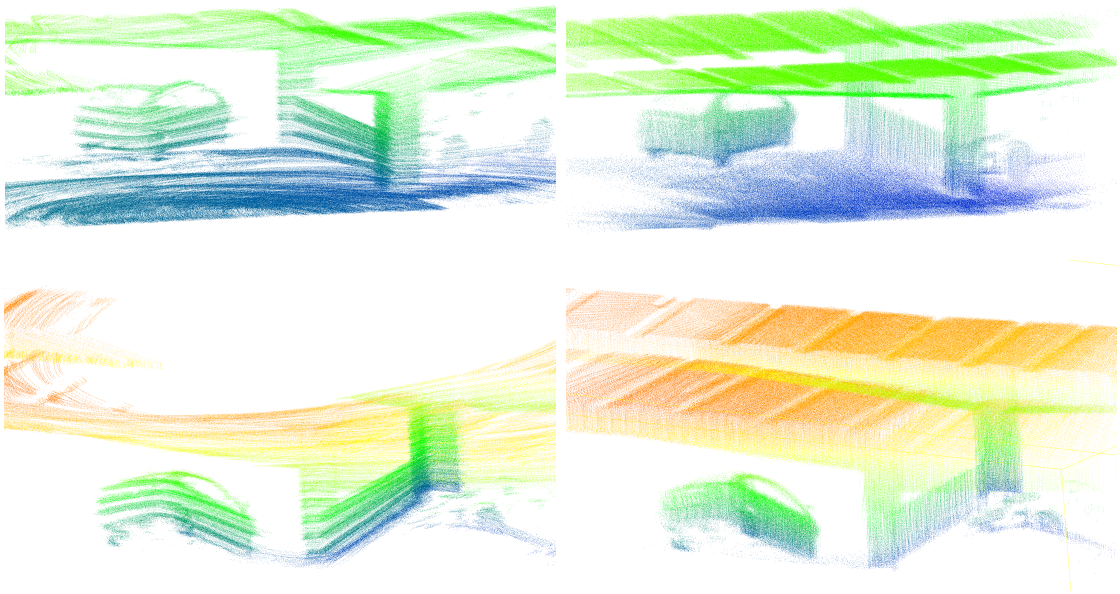


Figure 3.6: Example 3D maps generated with both lidars. The left column shows the map generated using the horizontal lidar, and the right column shows the map generated using the vertical lidar. The top two images show a view from an upwards angle looking down which shows the roof and the floor, the bottom images show a view from below looking upwards where the floor points have been removed so that the soffit is well visible.

literature, but is a necessary requirement for either of their success. SOTA multi-sensor SLAM requires precise time synchronization because it relies on using measurements from different sensors to estimate the same state of the robot at different time instances while in motion. In terms of inspection, combining data from different sensor modalities used for downstream inspection also needs the same detail of synchronization. Since all data is collected while the device is in motion, in order to produce accurate 3D maps where all data can be precisely co-located, the precise time is needed for all sensor data regardless of the device capturing the data, and this needs to be expressed with respect to the same reference time signal.

Before describing the time synchronization strategy proposed in this work, it is important first to define what time synchronization means in this work and in robotics in general. A robot collecting data from different sensors should timestamp the data from all sensors using a single reference clock. There are a few challenges with this. First, some sensors (e.g., lidar) perform the time-stamping internally, and therefore it is not always

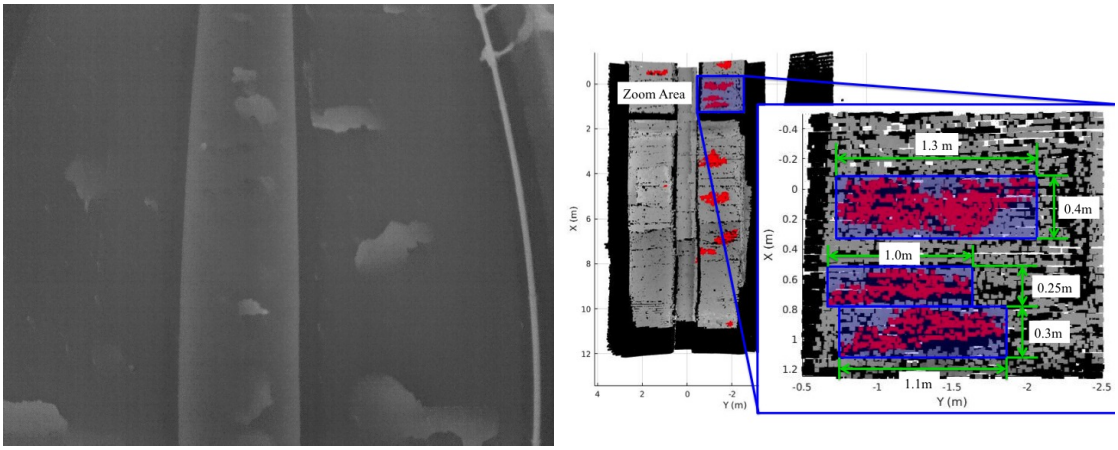


Figure 3.7: Left: Example image of concrete delaminations on a bridge using an infrared camera. Right: example quantification of delaminations using infrared images combined with the 3D lidar map

possible to have a single controller that timestamps all data. Second, data capture time is not always equal to trigger time or receive time. In other words, the device cannot just record the time the sensor was told to capture a data sample or the time the data sample was received by a computer. Feedback is needed from the sensor to know when the capture occurred. Third, computer operating systems have delays in running operations due to their internal job scheduling, thus it cannot be known for sure that incoming data was timestamped without a significant or variable delay.

Fig. 3.8 illustrates the proposed inspection system hardware synchronization strategy, which solves the above problems while keeping an extensible framework enabling a variety of sensor types and any number of sensors to be included. The synchronization strategy is designed around having a microcontroller that does the majority of the time-stamping and a separate computer that stores the sensor data with pre-estimated timestamps. This solves the first and third challenges listed above since it is a single reference clock where the device does not have a large operating system that would slow down operations. How each of the three sensor modalities (camera, lidar, IMU) are synchronized is specific to the sensor model itself, however, the overall approach would be similar for other types of sensors with hardware synchronization capabilities. The Xsens IMU used in this work does not allow for data capture triggering but has the option of outputting a sync signal, which can be used to inform the receiving device when each data was captured. The microcontroller, therefore, reads this signal and records each of the timestamps for the corresponding IMU. The Velodyne lidar used here, like most lidars, does its own time-stamping using the controller

on the sensor. To ensure the timestamps are in the same time domain as the microcontroller (and therefore all other sensor data), it can use a combination of a PPS (one pulse per second) trigger signal with corresponding timestamps in an NMEA string format. The lidar then generates its own Robot Operating System (ROS) message, which is sent to the data collection computer, where all lidar points are time-stamped with a consistent clock. Lastly, the FLIR Blackfly cameras can be triggered by an external Pulse with Modulation (PWM) trigger signal. Since there is a bit of overhead between when the camera receives the trigger and when the center capture of the exposure occurs, the camera returns a sync signal back to the microcontroller, which is then used to log the precise timestamp of the image. The microcontroller, therefore, has access to all timestamps associated with the cameras and IMUs, and those timestamps are packaged into ROS messages and sent to the data collection computer for storage or real-time processing. The sensor data itself always gets sent directly to the data collection computer from the sensors, and this computer finds the corresponding timestamp messages from the microcontroller. For more detailed information, including source code to implement this solution, refer to the GitHub repository `https://github.com/BEAMRobotics/ig_handle`. Note that this repository is a multi-student-run effort that has improvements beyond what has been defined in this thesis work.

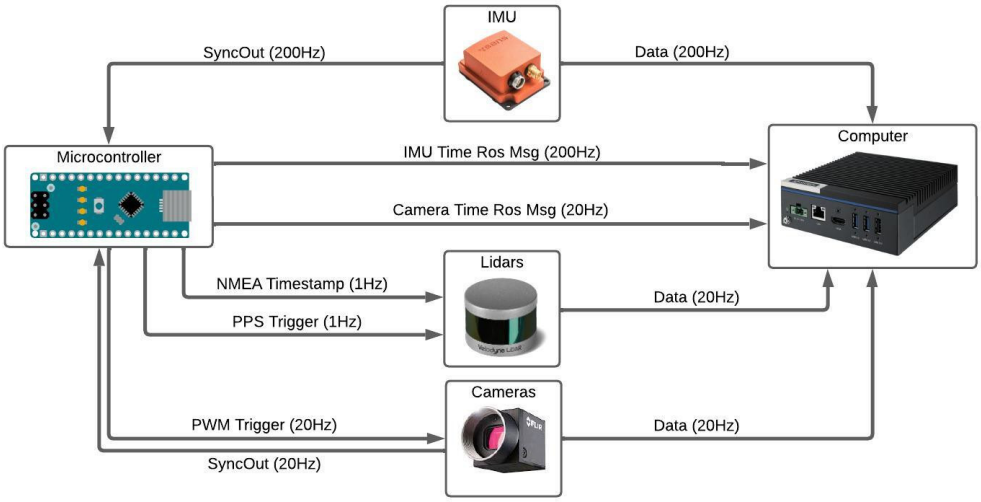


Figure 3.8: Time synchronization diagram

3.5 Conclusion

This chapter proposes a SLAM-centric infrastructure inspection system, including sensor types, configurations, and how to synchronize such a system. The proposed system is shown to be specifically designed to enable accurate and robust SLAM while also enabling an extensible system that can be adapted to perform inspection on a variety of use cases. This chapter starts by outlining the sensor configuration that ensures SOTA SLAM algorithms can be used appropriately onboard this device. The chapter then dives into such algorithms and describes how to select specific sensors given what has been used successfully in literature. The chapter also shows through building example systems that this system can be implemented on a variety of device types, including a handheld device, UGV, or USV, thereby eliminating any accessibility problems common to inspection tasks. It is also demonstrated through an example platform designed specifically for concrete bridge and parking garage inspection that this system can be customized to capture data, ensuring the highest quality inspection results and unlocking capabilities that were not possible prior to such a system. Finally, an extensible synchronization strategy is presented, which provides high-quality timestamp accuracy using a single reference clock, enabling this data to be fused correctly into a single 3D map despite being collected from a moving platform. This chapter presents the first full E2E hardware system designed from the ground up to perform robot-aided inspection from a moving platform running SLAM.

Chapter 4

Calibration

Calibrating robotic sensors to be used for automating inspection, as described in this chapter, presents new challenges that are not common to most robotic systems. Chapter 3 outlined the need to build systems with a variety of sensor configurations and a high quantity of cameras and lidar. Chapter 5 outlines how precise the fusion needs to be between lidar and cameras to unlock precise localization, map generation, and defects quantification. This chapter tackles the unique calibration challenges that surfaced from the SLAM-centric inspection system. The calibration requirements can be summarized as follows:

- **Agnostic to sensor configuration**: To ensure versatility to different inspection tasks, the calibration approach needs to be agnostic to sensor configuration and, therefore should not require any overlap between sensors.
- **Fast**: Different types of inspections will need different sensors, so frequent calibration will be required, and frequent re-calibration will be needed due to loss of alignment from rough site work.
- **Precise**: Precise calibrations are critical for not only SLAM, but more so for combining visual and lidar data for map visualization and defect measurement.
- **Repeatable**: To ensure consistent accuracy can be achieved in the inspection results, consistent accuracy calibrations are also needed.

The above challenges are mostly related to extrinsic calibration, so this chapter will focus on solving them using a new extrinsic calibration approach. Furthermore, the discus-

sions will be narrowed to the calibration of lidars and cameras, which are critical sensors for calibrating the proposed inspection systems.

In general, extrinsic calibration can be divided into two approaches: target-based and targetless. Target-based calibration methods ([160, 67, 46, 5, 121, 145, 53, 36, 74]) are performed in a lab setting and rely on extracting distinct features from specially designed targets that are observed by multiple sensors. These methodologies can achieve high accuracy but require an overlap in the FOV of the sensors in order to detect the same target from at least two sensors simultaneously. Since sensor overlap often only occurs between small subsets of sensors in larger systems, these methods require calibration transformations to be estimated individually and a chain of transformations between sensors to be generated. This approach can lead to compounding of calibration errors along the chain. The calibration accuracy also depends on the amount of overlap between sensors since the measurement sample space is restricted to lie within the overlapping region [118].

Targetless calibration approaches ([26, 81, 112, 137, 125, 106, 138, 8, 55, 60]) can overcome the sensor overlap requirement by moving the sensors through an environment and performing joint calibration and SLAM. These methods can associate various features in the scene between sensors, construct a map with multiple sensor streams while optimizing over SLAM states and calibration parameters jointly, or perform a hand-eye calibration that aligns SLAM trajectories created independently with each sensor. Contrary to target-based calibration methods, the accuracy of these methodologies cannot be guaranteed since corresponding features, building maps without calibration information, or aligning trajectories derived from the lidar and camera data are all susceptible to incorrect correspondence in natural scenes (i.e., without targets), leading to both reduced calibration accuracy and repeatability.

This Chapter presents a novel target-based calibration method that can perform extrinsic calibration of non-overlapping lidars and cameras without the requirement of performing SLAM. The presented method uses a MCS [96] to track the pose of the sensors and the targets. Therefore, the repeatable high precision benefits of target-based methods can be achieved, as well as the non-overlapping benefit of the targetless calibration methods by replacing SLAM with a MCS. Since the MCS accuracy is at least an order of magnitude greater than the accuracy of the sensors [96], the additional error introduced by the MCS to overcome the sensor overlap limitation is negligible.

The presented MCS-based methodology also introduces other benefits beyond high-accuracy target-based calibrations for non-overlapping cameras. Firstly, since all sensors can be calibrated with respect to one common frame of reference, the accuracy of the calibration is independent of the number of sensors and, therefore, eliminates the problem

of compounding errors from estimating a chain of transformations. Second, since the location of the targets can be estimated relative to the sensors at any given time, the search space for the targets in the sensor data can be reduced in order to eliminate outlier measurements and further increase the calibration accuracy and repeatability of the target-based calibration method.

To further increase the impact of this work, the implementation is designed to be agnostic to target design or feature extraction methods. This is done using point-to-point error terms, combining all target observations into a single calibration problem for each sensor, and using an iterative correspondence approach that is only possible with the MCS measurements. This allows any target to be used as long as that target can be extracted from the lidar and camera data, even if this target presents a normally degenerate case that could not fully constrain the calibrations with a single measurement. Despite the large variety of target designs used in previous calibration work, no mathematical framework or open source software exists that allows the user to easily implement any target design or feature extraction method of their choice into a calibration framework.

The specific contributions of this work are as follows:

- The first target-based calibration method for non-overlapping lidar-camera systems is presented, which outperforms SOTA in terms of accuracy and repeatability
- A novel target-agnostic calibration formulation is used, allowing for various target designs with and without unique feature points.
- An extensible open source software[1] is released, specifically designed for generic lidar & camera extrinsic calibration with a motion capture system.

Despite the high costs of a typical MCS, the proposed approach has many suitable applications. Firstly, the majority of research labs that work in robotics and computer vision already have a MCS for getting ground truth results for their research projects. Furthermore, many industries have significant budgets for setting up proper calibration labs for performing mass-scale calibration, such as fleet calibration for automotive industries or calibration of augmented/virtual reality consumer devices.

[1] https://github.com/nickcharron/vicon_calibration

4.1 Related Works

Following the prevailing camera calibration methods, many have implemented lidar-camera calibration using a planar checkerboard. Zang et al. [160] use a checkerboard with a two-step estimation approach to calibrate one RGB camera to one 2D lidar. Kim et al. [67] use a checkerboard to calibrate a 16-beam 3D lidar with six monocular cameras, where each camera-lidar pair is treated as its own estimation problem. In both cases, the targets must be moved around to collect measurements of the target at different viewpoints. In the work by Geiger et al. [46], multiple checkerboards are used to calibrate any number of cameras and lidars using a single measurement of the checkerboards. Similarly, Alismail et al. [5] use a planar target with a single black circle that can be detected in the image which is used to calibrate a rotating 2D lidar to a camera.

In all of the above cases, distinct features on the target cannot be detected in the lidar data, and therefore, the overall shape of the target is used with iterative correspondence methods or estimated plane properties. To avoid this issue and improve the accuracy of calibrations, others have designed targets with distinct detectable features. Rodriguez et al. [121] use a planar target with a single hollowed circle, which allows the center and radius of the circle to be measured and used for optimization. Both Velas et al. [145] and Guindel et al. [53] use a planar target with four hollowed circles, and the centers of the circles are extracted from both the image data and the lidar data to be used as features.

In all the above cases, care has to be taken to place the targets within the FOV of the sensors to ensure sufficient scan points have contacted the targets and the target is fully visible in the camera. This limits the possible sensor configurations to those with sufficient overlap, and calibration results may degrade as overlap becomes smaller due to a reduced measurement sample space [118].

Thus far, all the aforementioned works are target-based calibration approaches that have been implemented to estimate one lidar to one camera extrinsic calibration. For systems with multiple lidars and/or cameras, these calibrations must be repeated for every camera-lidar pair. This can lead to a build up of error if a chain of transformations is required and each link is estimated independently from one another. Domhof et al. [36] solve this issue by performing global optimizations of multiple cameras and lidars. This methodology also uses a planar target with four circular holes while extending the approach to radar extrinsic calibration using a protruding triangle in the center. Kummerle et al. [74] use a large sphere as a target where the sphere's center is detected in the image and lidar measurements. This method also performs a global optimization to estimate the extrinsic transformations for any number of cameras and lidars, which eliminates the possibility of

error accumulation with a pairwise extrinsic calibration method. These methods, however, still require overlap in the sensor FOVs, and the benefits of the global optimization of all calibrations may not be beneficial if the overlap is restricted to a few sensor pairs.

Targetless methods seek to define a calibration approach that does not explicitly rely on a particular calibration target, enabling calibration in the field and online calibration during robot operation. To do so usually requires robust correspondence of features or maps between lidar and camera data of some form. In the case of Castorena et al. [26] and Levinson et al. [81], targetless calibration is performed by associating depth discontinuities in lidar data with intensity discontinuities in image data. This approach can be problematic because color discontinuities do not always match with range discontinuities. Pandey et al. [112] and Taylor et al. [137] maximize mutual information in both the lidar data and camera data, but this method suffers from incorrect correspondences when initial calibration estimates are insufficient [74]. Repeatable and accurate calibrations can be challenging in these methods contrary to methods performed in a lab setting with a known calibration environment having distinct features. Zhou et al. [163] and Yuan et al. [154] use edge-based targetless approaches for extrinsic calibration between camera and lidar but heavily rely on scenes rich in geometric features. Koide et al. [73] proposed a general targetless lidar-camera calibration toolbox, but still requires enough sensor FOV overlaps for a good result. However, these approaches can be well suited for checking for calibration errors that develop while in operation after reliable initial calibration has been performed. So far, the above targetless calibration methods still do not solve the problem of requiring significant overlap in the sensor FOVs. Two general approaches have been proposed for non-overlapping targetless calibration: (1) SLAM is performed while the sensors are in motion and the calibrations are optimized ([125], [106], [85]), or (2) motion estimates are constructed with each sensor data stream individually, and a hand-eye calibration is performed to align the motion estimates offline ([138], [8], [55]). Since SLAM can be very environment-dependent, these calibration approaches are limited by the appearance and geometric properties of the environment, which means calibration accuracy cannot be guaranteed. Jeong et al. [60] also solve this problem for a lidar and stereo camera pair by using lane markings in roads, but this is limited to those applications for which such markings are available.

4.2 Methodology

4.2.1 Problem Formulation and Notation

This section will present the mathematical formulation of the problem and notation for a calibration setup with one lidar, one camera, and one target. It is important to highlight that this formulation can be extended easily to any number of cameras, lidar, and targets, and the released open-source tooling has such implementation included. Nonetheless, the descriptions herein are simplified to a single camera-lidar pair for clarity.

Fig. 4.1 depicts a typical calibration scenario and illustrates all of the frames and transformations involved. Let F_L, F_C, and F_T be the coordinate frames for the lidar, camera, and target, respectively. F_M is the stationary map frame in which all MCS measurements are expressed. Finally, a robot frame, F_R, is defined as the sensor base frame tracked by the MCS and can be arbitrarily located on the robot.

There are three types of quantities that are measured and estimated. The first type, denoted by dotted lines in Fig. 4.1, are the keypoints on the target as measured from the sensor frames. For the case of the lidar, $P_L^{l^n}$ is the measured vector from F_L to the lidar keypoint l^n. Similarly, $P_C^{c^m}$ is the measured ray (with unknown depth) from F_C to the camera keypoint c^m. The second type, denoted by solid lines, are transformations measured by the MCS. Let T_{MT} be the rigid transformation that can be used to transform points from the target frame to the map frame. Similarly, let T_{MR} be the transformation from the robot frame to the map frame. Both of these transformations are estimated at a high rate relative to the sensor data rate and to a sub-mm level of accuracy [96]. The third type, denoted by dashed lines in Fig. 4.1, are the calibration transformations being estimated. In this case, the unknown calibrations are from F_R to F_C and from F_R to F_L.

If initial estimates of the calibration transformations (dashed lines) are available, all frame poses have estimates at all times, forming a closed loop for each measurement. In this formulation, a measurement is a keypoint observation from a camera or lidar at a specific time. Since the MCS measurement frequency can be an order of magnitude greater than that of the lidar and camera data, all sensor measurements should have an associated MCS measurement very close in time; thus the interpolation error can be minimized without explicit hardware synchronization of sensor and MCS capture. This ensures that measurements can be taken reliably while the robot and/or the targets are in motion and without the need to integrate timing circuitry between the MCS and robotic system. Since the data for most lidars is continuous, if the lidar is in motion, it is expected that the lidar points are motion-compensated to discrete time intervals. The MCS and initial calibration estimates can be used for lidar motion compensation.

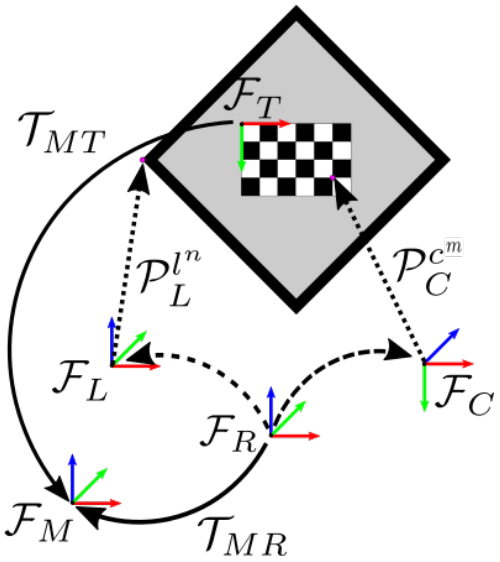


Figure 4.1: Calibration problem with all coordinate frames

4.2.2 Mathematical Framework

First, this section will consider the case of extrinsic lidar calibration to the robot frame as a separate problem. The problem can be written as follows: given a set of measured lidar keypoints, $\mathbf{P}_L^l = \{P_L^{l^1}, \ldots, P_L^{l^N}\}$, an initial calibration estimate, T_{RL}, as well as the set of corresponding ground truth keypoint locations expressed in the robot frame, $\mathbf{P}_R^l = \{P_R^{l^1}, \ldots, P_R^{l^N}\}$, estimate the robot to lidar frame transformation parameters. In this case, N is the total number of lidar keypoints, which depends on the number of target observations and the number of keypoints per target observation. To determine the ground truth location of the keypoints in the robot frame, the location of the keypoints in the target frame can be used (i.e., $P_T^{l^n}$), which are known a priori (e.g., hand measured or known due to target dimensions) and the high precision MCS measurements according to Eq. 4.1. For details on how measured lidar keypoints are extracted and their corresponding ground truth values are determined, see Section 4.2.3.

$$P_R^{l^n} = T_{RM}\, T_{MT}\, P_T^{l^n} \tag{4.1}$$

Using Eq. 4.1 and the initial calibration estimate, an error associated with the measurement of each lidar keypoint can be defined as a function of the calibration, as shown in Eq. 4.2. This error represents the Euclidean distance between the detected lidar point,

expressed in the lidar frame, and its corresponding ground truth lidar point transformed to the lidar frame using the current calibration estimate. This error should diminish as the current calibration estimate converges to the true value.

$$e_{l,n}(T_{RL}) = P_L^{l^n} - T_{RL}^{-1}\, T_{RM}\, T_{MT}\, P_T^{l^n} \tag{4.2}$$

Now, the case of camera extrinsic calibration will be considered in isolation. Given a set of measured camera keypoints, $\rho^c = \{\rho^{c^1}, \ldots, \rho^{c^M}\}$, an initial calibration estimate, T_{RC}, as well as the set of ground truth keypoint locations expressed in the robot frame, $\mathbf{P}_R^c = \{P_R^{c^1}, \ldots, P_R^{c^M}\}$, estimate the true calibration. Similar to the lidar case, an error term can be defined using the MCS measurements and the initial calibration. In this case, the camera-specific projection function $\pi(\ldots)$ can be used to project the estimate of the ground truth keypoint location into the image plane. Eq. 4.3 shows this error function which represents the difference between the detected pixels and the projection of their corresponding keypoints after they have passed through the MCS transforms and the current calibration estimate. Note that this work assumes that the intrinsics have already been calibrated; however, expanding this implementation to optimize over intrinsics is a straightforward extension that will be left for future work.

$$e_{c,m}(T_{RC}) = \rho^{c^m} - \pi(T_{RC}^{-1}\, T_{RM}\, T_{MT}\, P_T^{c^m}) \tag{4.3}$$

In practice, the keypoint locations relative to the target frame measured by the MCS cannot be determined exactly. Therefore, to account for this error, an additional target alignment error term is added, T_E, that is also included in the optimization. T_E essentially represents a transformation from the estimated target frame to the corrected target frame. This has been shown to significantly increase the accuracy of the final results. Including these target error parameters yields Eqns. 4.4 & 4.5.

$$e_{l,n}(T_{RL}) = P_L^{l^n} - T_{RL}^{-1}\, T_{RM}\, T_{MT}\, T_E\, P_T^{l^n} \tag{4.4}$$

$$e_{c,m}(T_{RC}) = \rho^{c^m} - \pi(T_{RC}^{-1}\, T_{RM}\, T_{MT}\, T_E\, P_T^{c^m}) \tag{4.5}$$

To solve for the calibration parameters, a non-linear solver is used to find the set of T_{RL}, T_{RC}, and T_E transformations that minimize the sum of the squared error terms for camera and lidar measurements as shown in Eqn. 4.6.

$$T_{RL}, T_{RC}, T_E = \underset{T_{RL}, T_{RC}, T_E}{\arg\min} \sum_{n=0}^{N} [e_{l,n}]^2 + \sum_{m=0}^{M} [e_{c,m}]^2 \tag{4.6}$$

4.2.3 Measurement Extractors

The previous section presents how calibration parameters are optimized given MCS measurements of target poses, MCS measurements of robot poses, a set of measured keypoints on the lidar scan or camera image, and a matching set of known keypoint locations in the target frame. This section discusses how the keypoint measurements are extracted from a lidar scan or image. As discussed in the introduction, a key novelty of this work is that the problem is formulated in a way that is agnostic to the target being used, and to the method for extracting measured keypoints. Knowing that, with initial estimates of the sensor extrinsics, a fully closed loop of transformations is available, measured keypoints and their corresponding known 3D locations can be transformed into a consistent reference frame and the corresponding features can be found using nearest neighbors. The following section will discuss more about the correspondence estimation. Using a point-to-point error term in the optimizer then makes this formulation agnostic to any target design. Furthermore, all point-to-point measurements are added to a single optimization problem, meaning that each observation of a target does not need to constrain all 6DOF. Instead, the solution can rely on multiple measurements of possibly degenerate target measurement to constrain all calibration DOFs. As a result, this work presents the first implementation that researchers can use to experiment on a large variety of different target designs and feature extraction methods.

For the target design, a diamond-shaped board with a center checkerboard is selected while also demonstrating another cylinder target design option in simulation presented in Section 4.3.1. The diamond shape provides a large surface area for the lidar to scan, with long distinct edges. The center checkerboard allows robust keypoints to be extracted from camera images. For extracting lidar measurements, experimentation was done using either the four corners of the diamond board as keypoints, or all points on the board. Results showed that the latter provided better calibration results, and therefore, those results will be presented. For the checkerboard detection, OpenCV's checkerboard detector was used.

Data pre-processing steps can be taken to decrease the probability of using outlier target measurements, which are only possible due to the unique circumstances offered by the MCS (i.e., the target's pose is always approximately known). First, by iterating

through the collected data at each time step, the target pose can be estimated relative to the camera. If the target is expected to outside the FOV of that sensor, that data can be ignored as there is no sense in trying to extract a measurement. Next, the target's velocity relative to the robot can be estimated so that high-motion measurements can be rejected which may have motion distortion in the lidar data or amplified measurement errors due to minor time synchronization imperfections. Once it is known that a target is likely to be observed by a sensor and is sufficiently stationary, the sensor data can be cropped to a smaller FOV to capture only the area surrounding the estimated target location. After cropping the lidar data to the expected location of the target, the lidar points are further narrowed down by performing Euclidean clustering and selecting the most likely cluster. The cluster tolerance can be predetermined for each measurement given the known minimal angular resolution of the lidar and the known distance to the target. The most likely cluster is selected based on a weighted score using the difference between the target pose and the measured cluster error, as well as the difference between the measured volume and the known target volume. This results in a small subset of the collected data for both sensor types in which to identify the target, which increases the likelihood of properly detecting the correct keypoints on the target. It is argued that these data pre-processing steps enable this method to have much more robust measurements compared to existing methods.

4.2.4 Correspondence Estimation and Optimization

To construct the error terms presented in Eqns. 4.5 & 4.4, keypoint measurements in the sensor frame $(P_L^{l^n}, \rho^{c^n})$ must be associated with the ground truth keypoint locations. Since each keypoint is not always uniquely identifiable, the correspondences between the measured points and the ground truth keypoints are estimated iteratively.

To solve this correspondence problem for a given target, the MCS measurements and the current estimate of the calibration are utilized to perform a Nearest Neighbor Search (NNS). First, the ground truth target keypoints are transformed from their target frames into the sensor frames. Following this step, the centroids of both point clouds are calculated, and the transformed keypoints are translated such that the centroids of both pointclouds align. A NNS is then performed to determine the correspondences between the measured and original keypoints on the target. For the case of the camera measurements, the keypoints are also projected into the image plane and NNS is done on the 2D pixel coordinates. Fig. 4.2 shows an example of the camera and lidar correspondence estimation for the diamond target.

The optimization of the presented cost function is solved using Levenberg-Marquadt

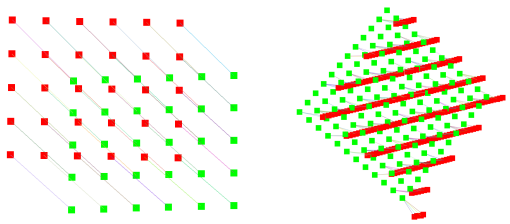


Figure 4.2: Example correspondence estimation for diamond target: camera keypoints (left), lidar keypoints (right). Red: measured keypoints. Green: transformed ground truth keypoints.

(LM) optimization and implemented with Ceres[2]. Since the original correspondences are often incorrect due to errors in the initial calibration parameters used to transform the keypoints, the correspondence search and optimization are iterated until convergence. That is, once the optimizer converges, the correspondences are recalculated, and a new optimization problem is constructed and solved. This process is iterated until either the change in total cost or the change in calibrations is below a threshold.

4.3 Results

This section will discuss the validation procedure for the proposed calibration approach and present key results. The results will be presented for simulation and experimental data, where simulation is used to test against known calibrations that have been perturbed. Both simulation and experimental data are validated using average keypoint reprojection error for the case of the camera calibrations and average keypoint Euclidean distance error for the lidar calibrations.

4.3.1 Simulation Data

A simulation environment was created using Gazebo and the Robot Operating System ROS to simulate a MCS calibration environment with one camera and one Velodyne VLP-16 lidar. To simulate the camera, a simple pinhole camera with no lens distortion is used to avoid any possible issues with improper intrinsic calibrations. Gaussian noise was

[2]Ceres is available at: http://ceres-solver.org/

added to the pixel intensities measured by the simulated camera with a mean of 0 and standard deviation of 1.8. The lidar was simulated using Velodyne's ROS simulator. To simulate the MCS, a ROS node was created that converts the model states provided by Gazebo to transform messages that describe the target motion relative to the map frame and the robot frame motion relative to the map frame. These odometry messages have been designed to emulate the data format from a MCS. No measurement noise is added to the MCS measurements due to the high accuracy that these devices can achieve relative to the accuracy of the sensors being calibrated. The Vicon MCS used in this work can achieve a mean absolute error of 0.15 mm [96], which is orders of magnitude smaller than, for example, typical lidar error which are on the order of several centimeters.

For simulation data, the true calibrations are known, therefore the rotational and translational errors of the calibrated transformations are presented. For the rotational error, the angles are converted to angle-axis representations, and the absolute difference in the angle is presented. For the translational error, the absolute difference in the translation norms is presented. For the case of the camera calibration, the reprojection error is also presented by taking the L2-norm of the vector between the projected points and the detected pixels. For the case of the lidar, the mean Euclidean error is presented, which is calculated by taking the norm of the vector between the target keypoints transformed into the lidar frame and the detected keypoints in the lidar scan data. To validate the sensitivity of the proposed approach towards various initial calibration parameter estimates, the known calibrations are perturbed by sampling linear and rotational perturbations from a random uniform distribution. For translational DOFs, perturbations are taken from -3 cm to +3 cm, and for rotational DOFs, perturbations are taken from -5 degrees to +5 degrees. The maximum perturbations are justified based on the maximum error that could be expected by hand-measuring transformations between sensor centers.

Table 4.1 shows the lidar calibration results using the diamond target. Tests were done with 5, 15, and 30 measurements of the target at different positions and orientations. In general, the mean and standard deviations of the sensor extrinsics and Euclidean errors are reduced as the number of measurements increases, but calibration parameter convergence occurs even with a small number of measurements. Looking at the diamond target results, a mean of 0.3 mm of translational error and 0.004 degrees of rotational error in the extrinsics is achieved after only five measurements.

Table 4.2 presents the camera calibration results using the diamond target. Similar to the lidar calibration results, the camera calibration converges with only a few measurements. After five measurements, the mean translation error and rotational error are 0.1 mm and 0.03 degrees, respectively. The reduction in the errors as the number of measurements increases is less apparent in this case. This is likely because the calibration errors

Table 4.1: Simulated lidar calibration errors

Test	No.	Trans. (mm)		Rot. (deg)		Euc. (mm)	
	Meas.	μ	σ	μ	σ	μ	σ
1	5	0.3	4.3e-6	3.8e-3	3.5e-8	2.2	9.0
2	15	0.1	2.0e-6	2.4e-3	3.6e-7	0.4	0.6
3	30	0.1	1.7e-6	1.8e-3	5.1e-8	0.5	0.1

are so low with only five measurements.

Table 4.2: Simulated camera calibration errors

Test	No.	Trans. (mm)		Rot. (deg)		Proj. (pixels)	
	Meas.	μ	σ	μ	σ	μ	σ
1	5	0.136	1.9e-6	0.034	1.3e-8	0.131	3.9e-7
2	15	0.066	3.6e-6	0.035	3.6e-6	0.167	6.2e-8
3	30	0.111	1.3e-6	0.035	9.2e-8	0.158	1.4e-8

To illustrate the calibration results, the target's pose at each measurement from the MCS can be conveniently used to overlay results. Given the template cloud for each target, the template cloud points and the measured keypoints can be transformed into the map frame using the initial, optimized, and ground truth calibration to see the accuracy of the calibration. Figure 4.3 shows one measurement example for each lidar target, where the blue points are the template cloud points transformed into the map frame using the ground truth calibration (i.e., the ground truth target pose), the green points are the scan points transformed to the map frame using the optimized calibrations, and the red points are the scan points converted to the map frame using the initial calibration (i.e., the perturbed ground truth calibration). The green points line up very well with the blue template cloud, which visually demonstrates the accuracy of the calibration.

Figure 4.3 shows similar results for the camera calibration; however, the points are projected into the image plane and plotted on the image measurement for visualization. In this case, the keypoints and the outline of the diamond target are shown to better visualize the results.

To demonstrate that the approach can be used for various target designs, including targets normally considered degenerate on their own, results are presented using a cylinder target. A cylinder target is an example that is very easy to construct with common construction material, easy to generate template models, and provides a 3D target shape

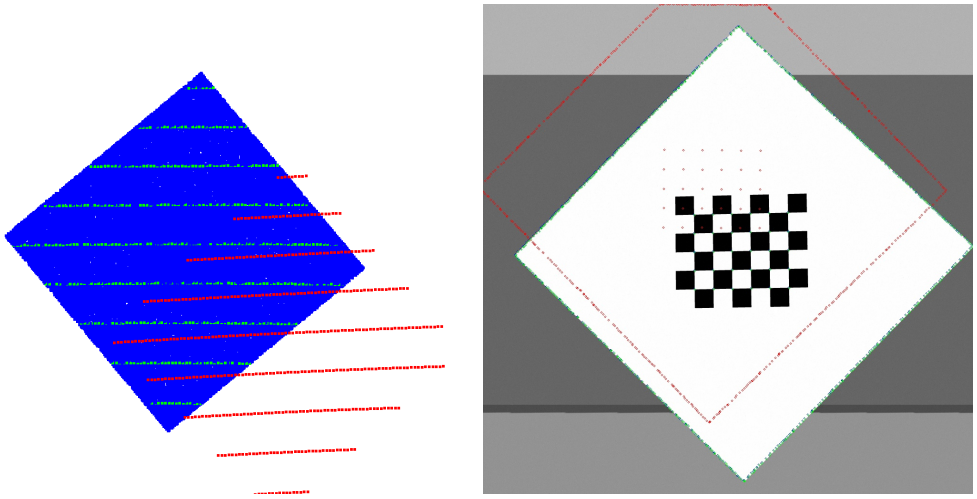


Figure 4.3: Example lidar calibration (left) and camera calibration (right) results using simulation data with diamond target. The red points are the initial estimates of the keypoints given the perturbed calibrations, and the green points are the keypoint estimates after optimization of the calibration parameters.

as opposed to the common 2D targets. Fig. 4.4 shows example calibration results using this cylinder. For extracting lidar keypoints, all lidar points that contact the cylinder are used. For camera keypoint extraction, color thresholding is first used to extract the target from the image, and then the convex hull from those points is extracted. Since this is a cylinder, a single measurement of the target would present a degenerate case for lidar and camera since the rotational axis of the target is ambiguous. However, the calibrations can still be accurately recovered using multiple measurements of the target. In these tests, with only 15 measurements, a calibration error of 0.1 mm & 0.002 deg is achieved for lidar calibration, and 4.6 mm & 0.1 deg error for camera calibration. These results are similar to the results achieved with the diamond target, except for the translation error for camera calibration which is a bit higher.

4.3.2 Experimental Data

To validate the presented methodology in the real world, several datasets were collected using a single lidar and camera rigidly mounted to a hand-held device, as shown in Fig. 4.5. The lidar is an Ouster OS0-32 Rev6, which is a 32-beam mechanically rotating lidar. The camera is a Flir Blackfly camera (BFS-PGE-16S2C-CS), with 720×540 resolution and

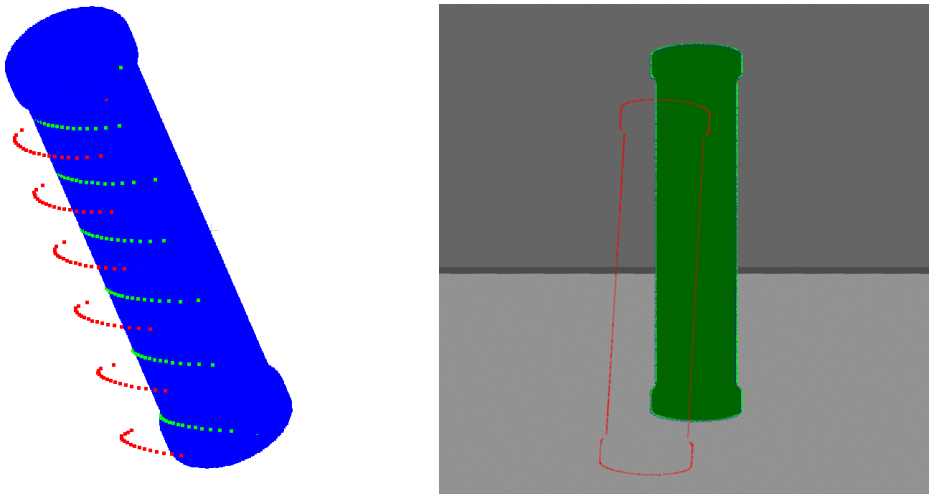


Figure 4.4: Sample results using cylinder target for: lidar calibration (left) and camera calibration (right)

a 49 × 38 degree FOV lens. Note that this work was done prior to the work in Chapters 5 and 3 and therefore uses a different set of sensors, however, this same methodology was used to calibrate the devices shown in this thesis. The camera is intrinsically calibrated using the Matlab camera calibration toolbox with a pinhole camera model with radial-tangential lens distortion. The MCS used in the experimental data is a Vicon MCS [96], and markers were setup on the targets (as shown in Fig 4.5) so that the MCS can directly measure the pose of the targets. All keypoint locations were hand-measured a priori in these coordinate frames, and template clouds were generated with the coordinate centers aligning with that of the real targets. MCS markers were added to the sensor rig, ensuring good visibility by the MCS cameras. For convenience, the x and y axes of the checkerboard target are aligned with the x and y axes of the diamond target, as seen in Figure 4.5. MCS markers can be precisely placed along the axis of the diamond target. The location of the robot coordinate frame is irrelevant since all calibrations are measured with respect to the same robot coordinate frame, and therefore, relative calibrations between all sensors can be determined in the end.

Since the lidar, camera, and MCS system are software synchronized in the above setup, the target and sensors were kept fixed at discrete locations while taking measurements. The MCS measurements were used to extract measurements only when the targets were fixed relative to the sensors. In both camera and lidar datasets, at least 20 measurements of the target at different positions and orientations are collected. Extrinsics are initialized

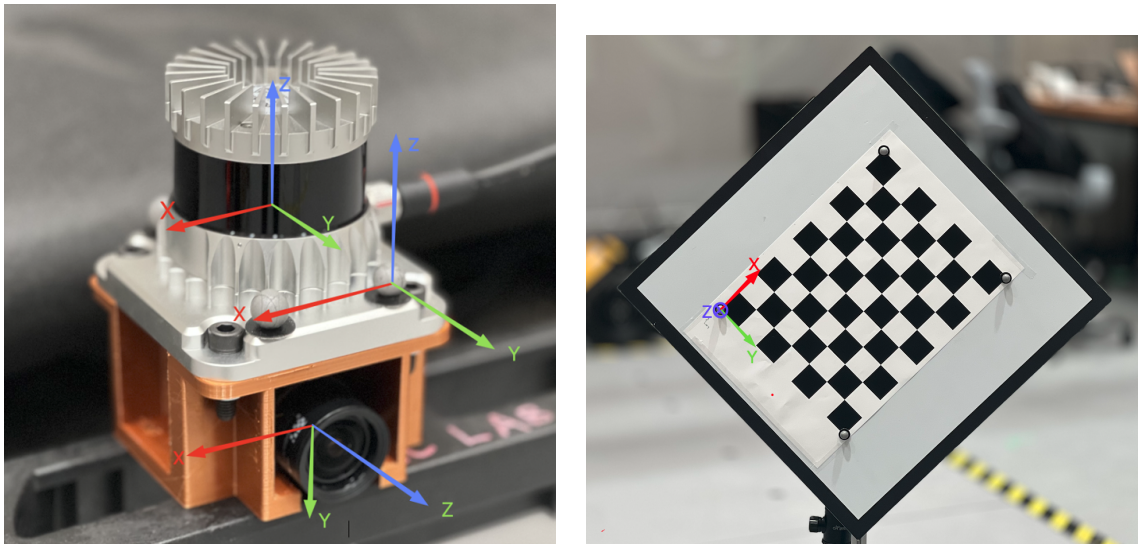


Figure 4.5: Experimental lidar and camera setup (left) and target setup (right)

using rough hand measurements and visually verified in ROS's RVIZ viewer.

Since the simulation results suggest that the calibrations should converge with as few as 15 target measurements, 15 to 23 measurements were used for experimental testing in Table 4.3, where calibration results converge and are consistent across measurements. For lidar calibration using the diamond target, an average Euclidean error of 0.016 m was achieved. According to the manufacturer's datasheet, the OS0-32 lidar shows a typical range error of +/-1.5 cm, suggesting that the +/- error of 1.6 cm to 1.7 cm is well within the error limits that would be expected even with a perfect calibration. This demonstrates promising results for the lidar calibration, given the high level of measurement noise in the sensor data. It should be noted that the experimental data clearly shows significantly higher noise in the lidar data compared to the simulated data. This explains the difference in final error terms between the simulation and experimental results. The main goal of the simulated experiments was to prove that the implementation was correct, thus lower amount of sensor noise was desirable. Future experiments could be performed with similar levels of noise to the real experiments to increase this study's confidence in the experimental results.

Fig 4.6 shows an example of lidar calibration results on the experimental data. Once again, the blue points are the template clouds converted into the map frame given the MCS measurements. The green and red points are the scan points converted into the map

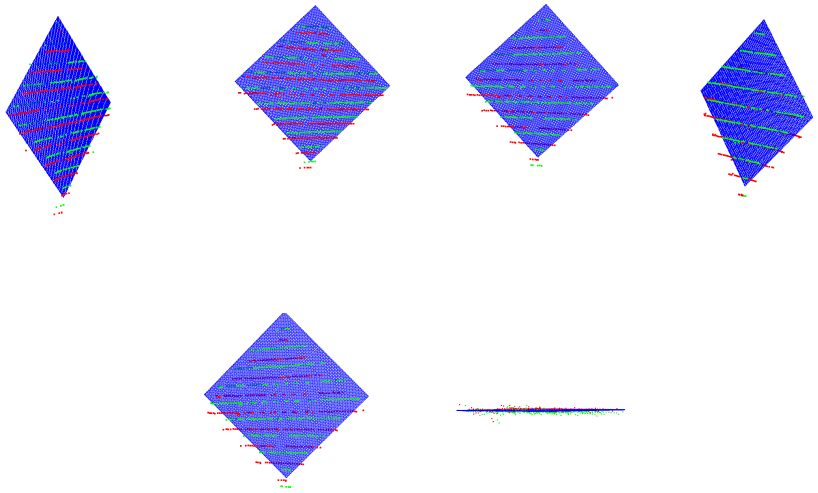


Figure 4.6: Sample lidar results. Top: 4 sample measurements, bottom: single example enlarged

frame using the optimized and initial calibrations, respectively. Despite the high noise in the lidar data, the green points visually align well with the blue template cloud. Fig 4.6 shows more lidar measurements at different angles and distances.

Camera calibration was also thoroughly validated with experimental data, where a similar reprojection error was observed for the experimental dataset relative to the simulation results. As shown in Table 4.3, using 10 to 20 measurements of the diamond target, an average reprojection error of 0.64 pixels was achieved. Notice here that errors get larger with more measurements. First, it is worth pointing out that these are projection errors, not total calibration error as is shown in the simulation experiments, since there is no way to know the true calibration parameters. The reason this error term increases with more measurements is likely because the smaller experiments took the best images with the lowest target velocities and more direct views of the target. Fig. 4.7 shows example camera calibration results. To compare results to the SOTA, the mean reprojection error can be converted to an approximate mean rotational error for each sensor calibration. Dividing the horizontal camera FOV by the image width results in the degrees of rotation per pixel, which can then be multiplied by the pixel error to get an approximate rotational error. Given an averaged reprojection error of 0.64 pixels, the estimated rotational error is com-

Table 4.3: Experiment calibration errors

Lidar Test	No. Meas.	Trans. (m)
1	15	0.0167934
2	20	0.0164762
3	23	0.0162282

Camera Test	No. Meas.	Proj. (deg)	Proj. (pixels)
1	10	0.03142	0.53167
2	15	0.04097	0.69323
3	20	0.04273	0.72292

puted at 0.04 degrees. This outperforms the targetless calibration methodology presented by Taylor et al. [138], who achieved rotational errors between 0.1 degrees and 0.4 degrees on each of the three principle axes. Also, this outperforms the edge-based calibration methodology from Zhou et al. [163] and Yuan et al. [154] with reprojection errors of 0.28 degrees and 0.13 degrees, respectively.

In summary, this work demonstrates experimentally that the proposed approach outperforms SOTA targetless camera-lidar extrinsic calibration methods. This method, therefore, provides the most accurate extrinsic calibration method that can be used to calibrate non-overlapping FOV lidar-camera systems currently available. Being a target-based calibration method, calibration accuracy is repeatable, which other methods relying on the environment cannot provide. Through simulation, it was also proven that this approach is robust to perturbation in the initial extrinsic estimates, and it is proven that this method can be used on various target designs, allowing researchers to experiment with different ways to extract measurements that better suit their specific data.

4.4 Conclusions

In this Chapter, it was shown that a MCS can be used to perform extrinsic calibration of non-overlapping FOV lidar-camera systems without requiring a SLAM implementation. The proposed methodology results in a repeatable and accurate calibration common to traditional target-based approaches without the overlapping FOV constraints. This was validated with experimental and simulation data while also validating that this implementation can be extended to various target shapes or feature extraction methods. This new extrinsic calibration methodology removes any limitations on sensor configuration while

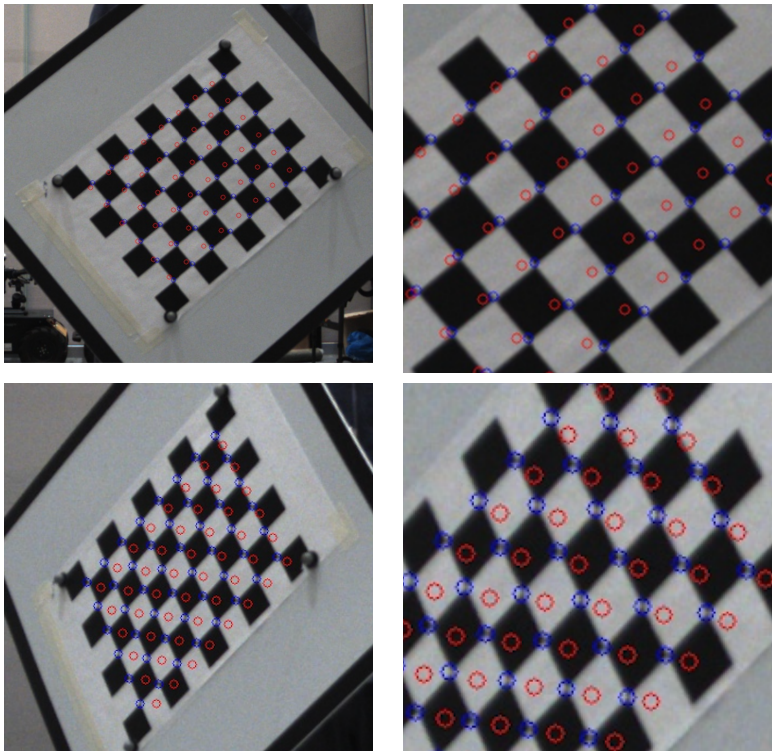


Figure 4.7: Example experimental camera results. Red: initial estimates, blue: estimates after calibration.

ensuring precise data fusion can be achieved to support any downstream data processing application in robotics, including the SLAM-centric inspection system presented in this thesis.

Chapter 5

Inspection Methodology

Mobile robotics has shown tremendous promise for improving how the inspection of infrastructure is done. In Chapter 3, a detailed study described how robotics and associated technology like computer vision have been used to augment visual inspection of infrastructure. This chapter further outlined how the vast majority of robotics for inspection has focused primarily on using robotic platforms as a means to access challenging areas and collect detailed images or pointcloud data of structures. Another significant focus outlined in Chapter 3 is automating information extraction from such data collected via robotic platforms. This includes detecting defects such as cracks, spalls, and corrosion from data such as images and pointclouds. These recent methods use advances in machine learning to reliably and efficiently extract meaningful information. Vision foundation models such as Dinov2 [109], and ViT [37] have shown exceptional abilities to capture intrinsic understandings of image data, which can then be used to build application-specific fine-tuned Machine Learning (ML) models such as infrastructure defect extraction models. However, to most effectively gather inspection data from robotics platforms, it is essential to properly utilize both the 2D and 3D data collected from these platforms. For example, image data has excellent detail, which, combined with SOTA ML models on images, can be a powerful tool to extract information such as visual defects. However, extracting real-world scale from 2D images alone is challenging. Conversely, lidar data is excellent at capturing scale, but it lacks resolution and has fewer advanced ML methodologies to extract useful data due to the additional complexity of learning from 3D information. This key idea motivates the central hypothesis of this thesis: building an inspection system centered around using SLAM to fuse both 2D and 3D data is the most effective solution to robot-aided infrastructure inspection.

Using images and lidar data, coupled with 3D SLAM to perform visual inspection of

infrastructure is not a new idea. There are multiple works that have attempted this, but they all lack key aspects to enable an E2E system that can extract precise and reliable information from inspection data at scale. For example, [62, 61, 69] build robotic systems with cameras and lidars but do not properly design their systems to enable precise calibration and synchronization needed to properly fuse sensor data for reliable SLAM or defect extraction. [40] uses RGBD sensors that do not operate well outdoors or at the long distances required for inspection. Furthermore, all existing methods rely on off-the-shelf SLAM methods that struggle to perform well in infrastructure settings and are not optimized for delivering optimal 3D maps for inspection. Lastly, none of the existing methods demonstrate an entire E2E system for inspection. They miss essential elements like offline pose refinement post-SLAM, how to extract useful raw data from video streams, how to generate dense trajectories and clean maps from SLAM results, how to combine image data and 2D data extracted from images to a 3D map, how to quantify defects in a 3D map, and how to track defect changes over several inspections. The specific contributions of the SLAM-centric inspection pipeline presented herein can be summarized as the following:

- The first E2E SLAM-centric inspection framework for generating precise quantitative visual inspection data that enables change tracking of defects over time
- A new map generation framework specifically designed for precise, dense, and clean 3D maps that uses SOTA multi-sensor SLAM combined with offline pose refinement and independent map generation
- A novel image to 3D pointcloud raytracing method for transferring image data to a 3D pointcloud map that is occlusion-safe, fast, and does not require a mesh

In addition to the contributions outlined above, all the code used to perform inspection, including mapping, is open-sourced for others to use and expand upon. The entire inspection pipeline, including image extraction, map labeling, and defect quantification is available at `github.com/nickcharron/beam_inspection`, the SLAM implementation including offline refinement is available at `github.com/BEAMRobotics/beam_slam`, and the map builder is available at `github.com/nickcharron/3d_map_builder`.

5.1 Related Works

This section reviews the most relevant literature that attempts to perform visual inspections using a combination of lidar and camera data, combined using multi-sensor SLAM.

For a broader review of robotic-aided infrastructure inspections, see Chapter 1, and for a more detailed review of the different robotic platforms used in previous literature, see Chapter 3.

Work by Jung et al. [62, 61] are early examples of researchers using both lidars and cameras for generating inspection data. They use a SOTA visual-inertial SLAM VINS-MONO [117] combined with NDT [17] and Generalized-ICP [126] scan registration. In this work, however, the authors did not properly design their system to effectively run multi-sensor SLAM since their sensors were not synchronized, resulting in poor mapping. In fact, the authors concluded, after comparing to SOTA multi-sensor SLAM solutions, that more robust SLAM is still needed to perform inspection reliably. Furthermore, the authors neglected to address how inspection data can be extracted from such a system.

Yang et al. [150] present the CityFlyer UAV. The CityFlyer is equipped with a structured light RGBD camera. This work proposes an inspection pipeline spanning from SLAM to defect detection and projection to the 3D map. For SLAM, the CityFlyer uses ORB-SLAM2 [105], which is an RGBD-SLAM method that tracks full 6DOF pose at camera rate. To get a higher rate SLAM trajectory, the CityFlyer fuses ORB-SLAM2's trajectory with IMU data using a multi-state EKF [102]. To correct for drift, the CityFlyer uses g2o [51] to perform a pose-graph optimization, which closes loops. Candidate loop closure locations are determined using the RGBD-SLAM's trajectory estimate, and loop closure constraints are generated by matching keypoints in proximate image keyframes. After generating a loop closed trajectory, the CityFlyer then detects concrete cracks and spalls using a Deep Neural Network (DNN) and projects the defects to a 3D voxel grid map. Unfortunately, structured light-based RGBD cameras are limited to close-range measurements (within a few meters), and the IR light they depend on for depth estimation is highly susceptible to corruption by sunlight, which makes this sensor inoperable in direct sunlight. Furthermore, RGBD SLAM still relies on detecting and tracking features. Therefore, regardless of whether they have the depth information or not, these systems will still struggle to accurately track pose in inspection environments with low visual saliency and poor lighting conditions in contrast to lidar SLAM systems, which are invariant to illumination conditions.

Feng et al. [40] use a system with a solid-state lidar and RGB camera precisely synchronized and calibrated. They generate a trajectory, 3D map including pointcloud and mesh using the R^{3}LIVE [83] SLAM software. Using the trajectory and calibration data, they can take crack detections from images and back-project them to the 3D map using raytracing. This work is probably the most similar to the work presented in this Chapter, but their approach is fundamentally tied to the R^{3}LIVE, which limits this solution significantly. Localization, mapping, and map colorization are all fundamentally tied to

the sensors on R^{3}LIVE rigs, which depend on a solid state lidar and camera pointed in the same direction, covering similar FOVs. R^{3}LIVE is known to perform poorly with the more common mechanically rotating lidars. The authors acknowledge that the R^{3}LIVE system requires the users to carefully ensure they are always looking in directions with high environmental structure, whereas lidars with higher FOV (i.e., 360 degree for the Velodyne lidars) are less restricted to this. There is also no mention of how to incorporate additional sensors into the mapping and colorization process, which is not trivial and is required for a more versatile inspection system such as the one proposed in this thesis. Lastly, the key component of transferring inspection data to the 3D map requires a mesh to be generated from their pointcloud map. Since lidars have an accuracy on the order of a few cm, generating meshes for complex shapes is unreliable, which can lead to errors in inspection results. The authors only show results on a single large curved surface, which avoids this problem altogether.

Kim et al. [69] claim they are the first to present a full E2E inspection system from a UAV. Their method includes SLAM, defect detection, and quantification, and they also focus on inspection planning and generating condition assessments from the inspection data. This work puts significant effort into comparing their inspection to human inspection and concludes that such a method does improve upon human inspections. Despite this method claiming to provide an E2E framework, their methodology to generate inspection data is fairly crude and limiting. Defect quantification is done using a 1D lidar connected to a gimballed DSLR camera. The camera data is not actually combined with 3D lidar map data generated from SLAM, significantly restricting the possibilities of this approach. There is also no mention of how the gimbal or camera is calibrated to 1D lidar, nor does it mention how data was time-aligned, which implies that proper data fusion is not possible with their solution.

Yan et al. [149] uses a UAV with a 3D lidar that is mechanically rotated, along with a gimballed camera for images. Their method runs the common LOAM SLAM methodology to generate a lidar map and trajectory. To combine image and map data, they develop a novel raytracing algorithm that uses projection but also accounts for occlusions by separating the image into grids and taking only the points closest to the camera in each grid. This work, however, does not propose a way to combine image and 3D lidar data in a final 3D map. Instead, the opposite action is performed where the 3D map is simply used as a way to inform defect detection in images. Lidar maps are semantically segmented using existing works, and these segments are mapped into the image to better inform image defect detectors. This does not provide inspectors with a 3D map containing all relevant inspection data, which is a significant advantage suggested in this Chapter. [149] also acknowledges that error in the lidar map and poses results in considerable error when

combining the image and lidar data, demonstrating the need for more accurate methods of fusing such data during inspections.

In summary, the research to date has not yet proposed a versatile method for generating high-quality, repeatable inspection results. Offline mapping has not been presented, although it has the potential to improve map quality and inspection results dramatically. Solutions for generating clean inspection maps beyond simply combining 3D sensor data with poses output directly from SLAM are scarce in the literature. A single combined 3D map containing all inspection data in a single data format which inspectors can easily visualize and compare across different inspections for easy defect tracking is lacking in the literature. This chapter will outline the method proposed in this thesis, which attempts to solve the above-stated gaps.

5.2 Datasets

Throughout the remainder of this Chapter, each component of the proposed inspection pipeline will be described along with results generated using the two datasets described here. These datasets show two different inspection applications well suited to the SLAM-centric inspection approach: (1) parking garage inspection, and (2) bridge inspection. Parking garages and bridges are good candidates for this SLAM-centric inspection because they are easily accessible, and they require regular inspections. The following sections will describe the two locations in more detail, followed by a description of the platform used to collect data.

The inspection platform used to process both datasets is the UGV presented in Section 3.3.2.

5.2.1 Park Street Bridge

The Park Street Bridge is located at the intersection of Park Street and Cherry Street in Kitchener, Ontario, Canada. The bridge is a concrete box girder bridge with one center concrete pier and two concrete abutment walls. The bridge spans a two-lane public road with a clearance of 3.4 m, and it supports an active railway track. The bridge is in very poor structural condition, with significant concrete spalling and cracking throughout the structure. There is also significant discoloration, some paint or patching, as well as some graffiti. The amount of deterioration makes this a good candidate for this research work. Furthermore, there is one sidewalk on each side of the road, and the roadway is not

significantly busy, which makes access easy for research. Fig. 5.1 shows a few different views of the Park Street Bridge, including some of the deterioration.



Figure 5.1: Park Street bridge dataset images

5.2.2 Duke Street Parking Garage

The Duke Street Parking Garage is located at the intersection of Duke Street West and Ontario Street North in Kitchener, Ontario, Canada. This is a public multi-floor parking garage with concrete slabs, girders, and columns providing all the support. At the time of inspection, there was some spalling and cracking, with some orange paint highlighting some of the spalling. The garage also had several parked vehicles at inspection time. The concrete slab is sloped in part of the garage, however, the part that was inspected which

had the deterioration was flat. Fig. 5.2 shows four photos of the parking garage, including some cracking on a column and spalling/cracking on the slab soffit.



Figure 5.2: Duke Street parking garage dataset images

5.3 Inspection Methodology & Results

The goal of the inspection methodology is to generate precise, reliable, and trackable visual inspection data. Precise here means that size estimates in real-world coordinates are at least more accurate than estimates from standard human visual estimates common to existing inspection approaches. Reliable here means that measurements of defects can be repeated with minimal to no human labeling or supervision, regardless of the infrastructure

type or environmental conditions. Data from the above-described platforms should be able to run through this pipeline with few failures, regardless of the conditions, such as the inspection path set by the inspector, or the environment layout, as is commonly an issue with existing literature. Finally, trackable data means that defects need to be located within a 3D map that can be used in reference to previous inspections to track change over time. The power of the proposed SLAM-centric inspection system, is that this is the first approach to enable precise tracking of defect growth over time for an entire structure, allowing quantitative asset degradation models to be generated. This is enabled by the precise, reliable, and automated tracking of defect measurements.

Fig. 5.3 illustrates the overall workflow used in the proposed inspection pipeline. Each step of the pipeline will be discussed in detail next, where the design decisions that relate specifically to inspection data generation will be described in detail. It will be shown how the steps in this pipeline are general to most visual infrastructure inspection types, and thus can be extended to many applications beyond bridge and parking garage inspection shown herein. The ultimate output of the pipeline is to generate a high-quality, dense, and visually accurate 3D map of the structure, with defect information stored and easily viewable to inspectors.

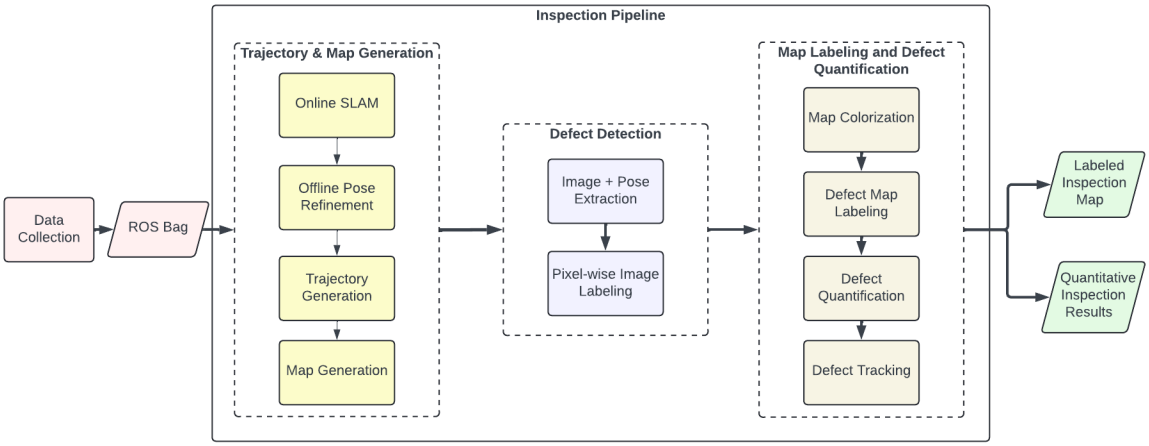


Figure 5.3: Diagram showing overall inspection workflow

From a high level, the pipeline starts with a dataset collected from some mobile inspection rig. The inspection system used in this thesis uses ROS to collect all data, and therefore, the data is stored as a ROS bag. Note that the pipeline is described as an offline process, however, due to the flexibility of ROS, many of these processes can be run in

real-time on data being streamed by the device to a processing computer. For brevity, real-time processing capabilities are neglected here, but none of these processes are too computationally expensive to run in real-time should they be implemented and tuned for this purpose, which would enable in-field evaluation of the collected data. The inspection pipeline is broken into three major steps: (1) trajectory and map generation, (2) defect detection using images, and (3) map labeling and defect quantification, which combines the defects generated in step 1 with the 3D map generated in step 2 and uses these results to generate the final inspection quantities that inspectors need. The following three subsections will dive into the details of those steps. The results of each process will also be presented for the datasets described in Section 5.2. Since there exist no similar works that generate such inspection data in an E2E fashion, it is impossible to compare to SOTA, instead this work relies on qualitative assessment and ablation studies to validate the approach. To quantify the accuracy improvement of each of the mapping steps described below, a new map quality analysis technique was developed and is presented in Appendix B along with some results. For brevity, the results of running the map quality analysis on both datasets for each of the localization and mapping steps described below are left out, but this tool was used heavily in testing and designing the overall approach presented herein.

5.3.1 Trajectory & Map Generation

Fig. 5.4 shows the full mapping workflow, including inputs and outputs of each step in the pipeline. The trajectory and map generation component first starts by running an online SLAM using a combination of lidar, camera, and inertial data. This generates a first estimate of the trajectory, with a low-rate pose synced to camera images and/or lidar scans (depending on the SLAM implementation used), and a high-rate pose that uses the IMU data to estimate the trajectory between the low-rate poses (as is standard in most modern SLAM implementations). Not only does the online SLAM provide initialization for the remaining two steps, but it can also be run in real-time during data collection so that inspectors can see coverage and ensure proper tracking of the inspection device. It should also be noted that this framework is not tied to a specific SLAM implementation, instead the SLAM used in this work is designed to resemble classic SOTA SLAM methods, and the rest of the inspection framework is designed around the typical outputs from any online SLAM. The second step in the mapping pipeline is offline pose refinement which is aimed at optimizing for the best possible trajectory given more compute resources and no restrictions on compute time. Next, the high-rate poses from online SLAM are combined with the refined low-rate poses, and a high-rate refined pose trajectory is generated. This

trajectory is then provided to the map generation, which produces a clean inspection map. The following discussion will dive deeper into each of the four components.

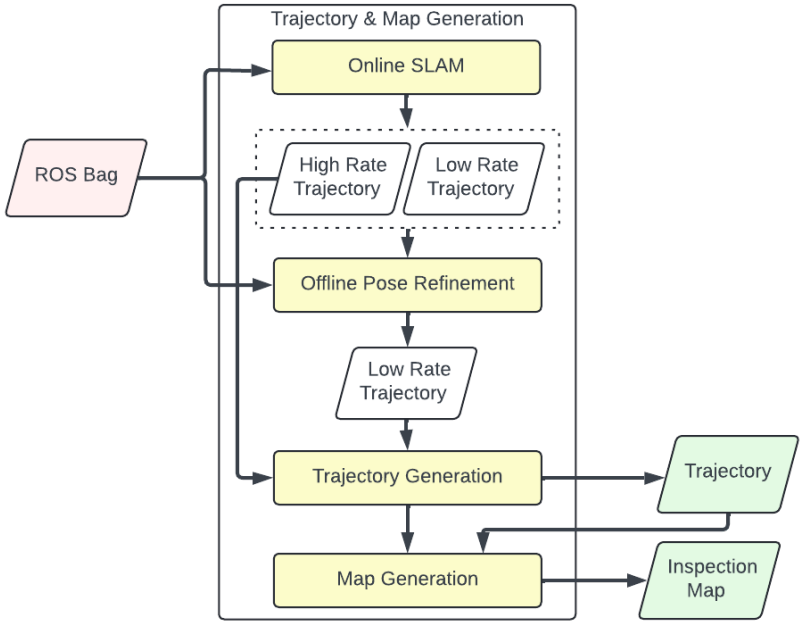


Figure 5.4: Diagram showing the mapping component of the inspection workflow

Online SLAM

Literature shows the most robust and precise SLAM method for high-motion system tracking are the methods that couple data from lidar, cameras, and IMUs [162]. Infrastructure inspection environments often present very challenging SLAM environments such as low visual saliency (e.g., concrete structures), repetitive structures (e.g., steel bridges, background water), low physical structure (e.g., culverts or bridges), poor or highly variable lighting (e.g., bridges), and high dynamic elements (e.g., vegetation, water). Therefore, the combination of the three sensor types becomes crucial, not only for increased precision in challenging environments but also for increased robustness to system failure. Lidar and camera are complimentary sensors: lidar are able to measure precise depth, but with low spatial density, and cameras generate high spatial density, but estimating depth is challenging. Therefore, combining both sensor modalities is critical for robust SLAM systems in

challenging inspection environments. The inclusion of IMU not only further increases accuracy but also provides always-on additional state measurements when the other systems struggle and provides high-rate motion estimates between camera and lidar measurements. The latter being a critical part of the map-building process, which will be discussed later.

The SLAM method used in this thesis is designed after the state-of-the-art Lidar Visual Inertial (LVI) SLAM work named LVI-SAM [130]. The overall workflow is illustrated in Fig. 5.5. First, data is ingested either by playing back a ROS bag or by directly consuming the sensor data being published by the robot over ROS. This data consists of the three SLAM sensors discussed, including lidar data, IMU data, and camera data. Each of the three sensor modalities has its own odometry estimator.

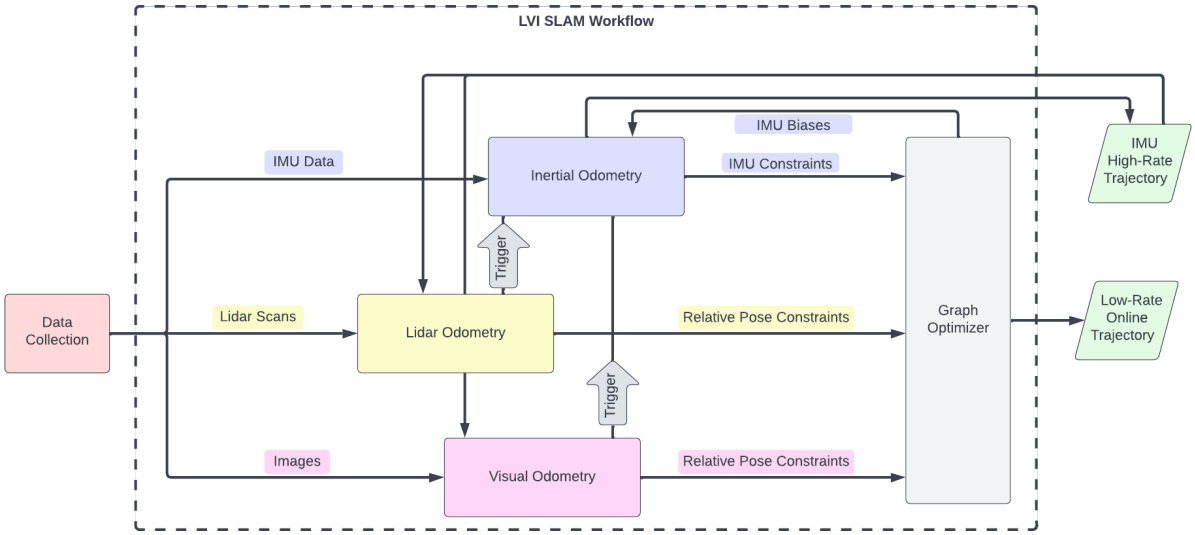


Figure 5.5: Flowchart illustrating the online SLAM implementation

First, there is an Inertial Odometry (IO) module. This module is always on, generating continuous high-rate and high-drift odometry that can be used by the other modules for initializing sensor data poses. IO takes all IMU data, pre-integrates the raw data to estimate poses for each measurement using the most recent bias estimates from the graph, and outputs an odometry estimate. It then waits for triggers from any of the other modules described below, and when a trigger is received, it integrates between the last trigger and the current trigger and sends an IMU constraint to the graph optimizer. The standard IMU pre-integration and relative IMU constraints presented in the seminal work by Foster et al. [44] are used. The IMU states in the graph include **SE3** poses as shown in Eqn. 5.1 augmented to include linear velocity (v), angular velocity (w), and IMU biases (ba, bg) as

described in Eqn. 5.2 below. In Eqn. 5.2 , ba_i are the IMU acceleration biases for state i, and bg_i are the IMU gyroscope biases for state i. For more information on how these biases are used, refer to the IMU motion model presented in Section 2.5.3, or the IMU pre-integration work presented by Forster et al. [44].

$$X_i = \begin{bmatrix} t_i \\ q_i \end{bmatrix}; \quad t_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}; \quad q_i = \begin{bmatrix} qx_i \\ qy_i \\ qz_i \\ qw_i \end{bmatrix} \tag{5.1}$$

$$\bar{X}_i = \begin{bmatrix} t_i \\ q_i \\ v_i \\ w_i \\ ba_i \\ bg_i \end{bmatrix}; \quad v_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{z}_i \end{bmatrix}; \quad w_i = \begin{bmatrix} w_{x,i} \\ w_{y,i} \\ w_{z,i} \end{bmatrix}; \quad ba_i = \begin{bmatrix} ba_{x,i} \\ ba_{y,i} \\ ba_{z,i} \end{bmatrix}; \quad bg_i = \begin{bmatrix} bg_{x,i} \\ bg_{y,i} \\ bg_{z,i} \end{bmatrix} \tag{5.2}$$

Second is the Lidar Odometry (LO) module, which is implemented very similarly to LIO-SAM [129]. In LO, raw scans are ingested and de-skewed using the local IMU trajectory, which should be accurate over small windows. Next, the position of the scan is estimated from the previous scan's optimized pose, combined with the IO output to estimate the motion between the last scan and the current. The scan is then registered against the local lidar map using the standard LOAM [157] registration technique, and finally, this new scan is added to the local lidar map. The lidar map is down-sampled each time a new scan is added, and old scans are removed to keep a fixed map size. LO produces relative pose constraints for each new successfully registered scan, and sends this to the graph to be optimized.

Third is the Visual Odometry (VO) module which is implemented similarly to VINS-MONO [117]. In VO, features are extracted and tracked using KLT [89], and a standard camera reprojection error cost function is employed on all tracked features in the VO graph window. The image poses are also initialized with IO output and include the inertial constraints in the VO graph, which, in fact, makes the VO module a visual-inertial module. VO, however, only outputs relative pose constraints to the mapper's graph.

A final trajectory is optimized for using Levenberg-Marquardt optimization over a sliding window of data. The open-source fuse[1] ROS-based optimization framework is used

[1]https://github.com/locusrobotics/fuse

for the back-end optimization, which provides a lot of flexibility in terms of running online, offline, or running in batch. The SLAM implementation is also open-source and can be found at `https://github.com/BEAMRobotics/beam_slam`. This repository was also a collaboration with Jake McLaughlin, who led the visual odometry component, and Alex Thoms, who helped write the inertial odometry component.

For more information on relative pose constraints in a factor graph framework, refer to Chapter 2, Section 2.2.2. Since the fuse optimization framework is used, which parameterizes poses as translation vector plus unit-quaternion for orientation (Eqn. 5.1) as opposed to using the $\mathbf{SE(3)}$ group, the relative pose constraints used here differ slightly from what is shown in Chapter 2. Eqn. 5.3 shows the relative pose constraints defining an error between two poses (translation and orientation). In Eqn. 5.3, q^{-1} is the quaternion inverse, $(\cdot)$ is the quaternion product, and $\mathcal{F}$ is the transformation from quaternion to axis angle representation. Furthermore, dt_{ij} and dq_{ij} are the measured translation and orientation difference between X_i and X_j output by VO or LO, respectively.

$$e(X_i, X_j) = \begin{bmatrix} e_{t_{ij}} \\ e_{q_{ij}} \end{bmatrix} = \begin{bmatrix} dt_{ij} - (t_i - t_j) \\ \mathcal{F}[dq_{ij}^{-1} \cdot (q_i^{-1} \cdot q_j)] \end{bmatrix} \tag{5.3}$$

To account for imperfect extrinsic calibrations and for any non-rigidities in the sensor mounts that may cause relative movement between sensors, online extrinsics calibration is also implemented. The graph optimizer assumes all poses are in the IMU frames, and both LO and VO operate in their respective sensor frames. Therefore, extrinsic calibrations between lidar, camera, and IMU can easily be optimized for by adding the transform from the lidar and camera to IMU in their respective cost functions and telling Ceres to optimize for these quantities. In practice, this is done by transforming the poses, which are estimated by the optimizer as being relative to the IMU frame, to the sensor frame (i.e., lidar or camera frame) using the extrinsics, and ensuring the LO and VO modules calculate their relative poses (dt_{ij}, dq_{ij}) with respect to the lidar or camera sensor frame. Using the extrinsics, the pose (t and q) in Eqn. 5.3 can be calculated as shown in Eqn. 5.4, where $\mathcal{T}_{WS_i}$ is the transform from the sensor frame (camera or lidar) to the world frame for instance timestamp i, X_i is the pose defined in Eqn. 5.1 and used in Eqn. 5.3, and $\mathcal{T}_{IS}$ is the extrinsic calibration between the sensor frame and the IMU frame.

$$\mathcal{T}_{WS_i} = \begin{bmatrix} R_i & t_i \\ \mathbf{0} & 1 \end{bmatrix} = \mathcal{T}_{WS_i} \cdot \mathcal{T}_{IS} = X_i \cdot \mathcal{T}_{IS} \tag{5.4}$$

Fig. 5.6 shows the lidar map resulting from the online SLAM process on both the parking garage (top) and bridge (bottom) datasets. To better visualize the quality of the

trajectory and map generated in SLAM, these images show the full lidar scans transformed into the world frame using the SLAM estimated pose for each scan. This is not the lidar map generated during SLAM, as this map is very sparse, and it is hard to make out the accuracy of the SLAM by visually inspecting this map.

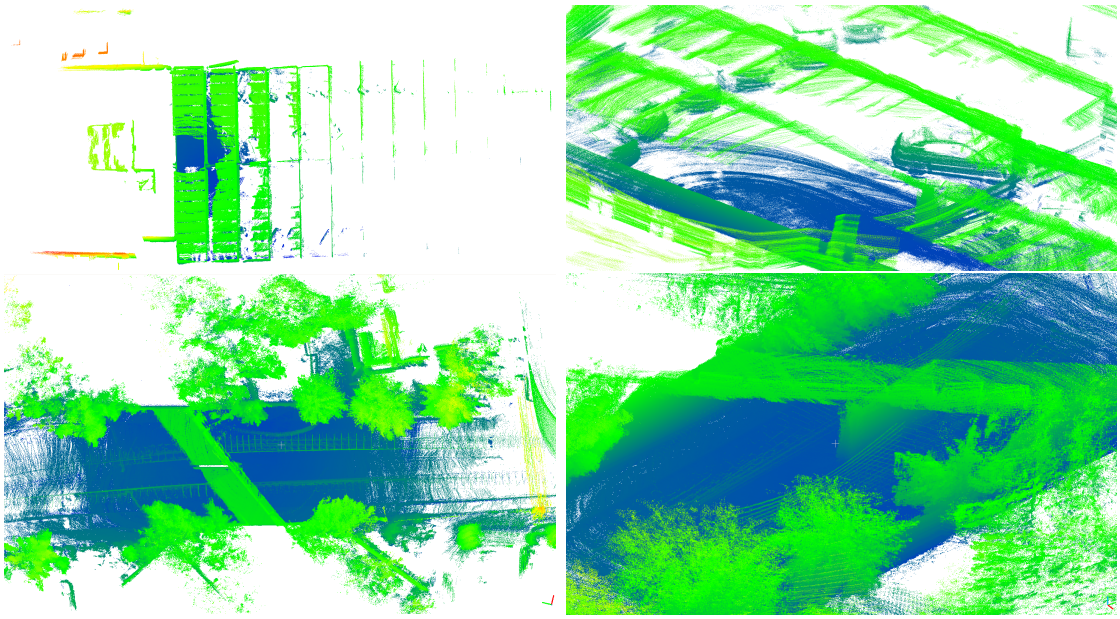


Figure 5.6: Lidar map results from Online SLAM. Top row: parking garage dataset, bottom row: bridge dataset, left column: plan view, right column: oblique view

Offline Pose Refinement

Given an initial trajectory estimated online using SLAM, the trajectory can be further refined given more compute time and resources if available. For an inspection pipeline, full results are not expected to be needed in real-time on-site. Current inspection practices involve collecting images and notes on-site, driving back to the office, and slowly converting such data into a final report for the client. It is, therefore, reasonable to expect that data can be post-processed in the office with larger computers than what was available on the inspection platform. This assumption is leveraged to further improve the trajectory and mapping results.

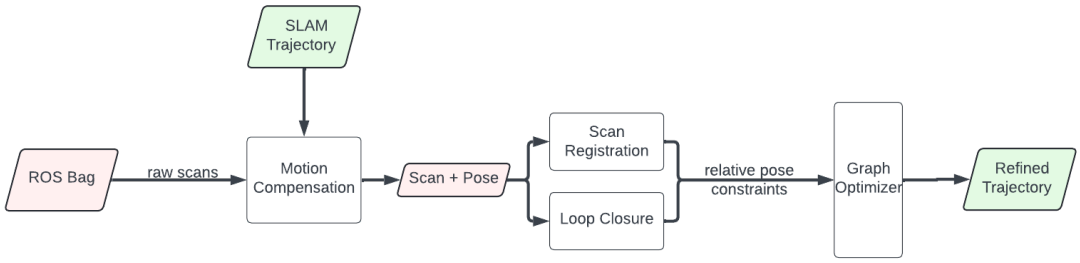
Previously in this chapter, specifically Section 5.3.1, it was discussed that leveraging camera, lidar, and IMU data provides the ideal conditions to enable robust SLAM that

is least susceptible to degenerate SLAM conditions due to the opposing strengths of each sensor modality. However, in ideal conditions, SLAM literature has shown that localization using lidars can consistently outperform localization using cameras. For a good example demonstrating this, refer to the Hilti SLAM Challenges [56]. As discussed in Chapter 2, Section 2.3, ideal conditions for lidar-SLAM methods (which rely on some sort of scan registration) are when good initial pose estimates are provided, and dense scans are used. During offline pose refinement, both these conditions can be met since there are good initial estimates from online SLAM, and scans can be aggregated over short windows using the SLAM results to provide dense combined scans. The offline pose refinement is therefore designed to use only lidar data, and the initial trajectory provided by the online SLAM module.

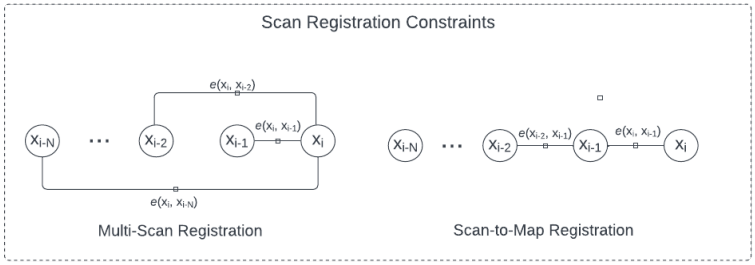
Fig. 5.7 illustrates the offline pose refinement process. The process starts with the raw data from the inspection system, as well as a trajectory from SLAM which should contain a high-rate pose from IMU pre-integration and a lower rate pose at camera or device rate (see Section 5.3.1 for more information on this). The high-rate IMU trajectory is first taken to motion compensate the scans, similarly to what is done in online SLAM described in the previous section. Then the scan is combined with a pose estimate from the lower-rate, more precise SLAM trajectory. Alternatively, the motion-compensated scans with poses can be saved directly during online SLAM so this step does not need to be repeated.

The core of the refinement happens in two steps. First, each incoming scan goes through scan registration and then loop closure. Each of the two steps generates relative pose constraints that get added to the optimization graph with the option of being solved incrementally for each incoming scan. For more information on relative pose constraints in a batch optimizer, refer to Chapter 2, Section 2.2.2. The same relative pose constraints defined in the online SLAM section presented above are used. The offline refinement step is designed to be as versatile as possible, allowing the user to swap between different methods and underlying algorithms in order to experiment to get the best possible trajectories for each dataset. Tests have shown that different implementation details work better on certain datasets than others, therefore setting up a pipeline that can easily experiment between different implementations is beneficial. The following paragraphs will detail the scan registration and loop closure steps.

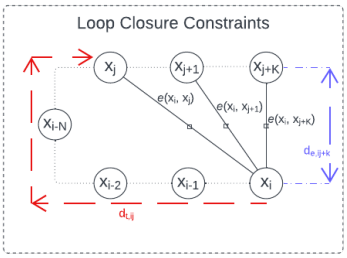
Scan Registration: The overall scan registration process is broken down into two different design spaces. The first is called the matcher methodology, which determines which sets of scans will be aligned to each other. The second is the registration methodology, which describes how two matched scans are aligned to each other in order to be able to estimate the relative pose between them. Two different matcher methodologies are implemented: multi-scan, and scan-to-map. In the case of multi-scan matching, each



(a) Overall block diagram



(b) Scan registration visualization



(c) Loop closure visualization

Figure 5.7: Illustration of the offline pose refinement implementation

new scan is registered against N previous scans. Scan-to-map registration incrementally builds a map with each new scan and matches all incoming scans to this map. Old scans are removed from the map as the map gets too large. Scan-to-map is what is used in the online SLAM implementation since it is less computationally expensive when the map is not too large and can be more robust to outlier measurements. Fig. 5.7-b illustrates how constraints get added to the factor graph for both scan registration methods. The scan registration methodology is what Section 2.3 refers to as scan registration and many different implementations in the literature that can be tried. This work implements a total of 4 different registration methods: LOAM [157] feature-based registration, point-to-point ICP, and GICP. The final results shown here use scan-to-map registration with a map size of 200 scans maximum, and the registration method is LOAM.

Loop Closure: The goal of loop closure is to find locations that have been visited multiple times, and reduce the trajectory drift by aligning the measurements between each visit of that location. Each new scan goes through the loop closure process to see if constraints can be added between that scan and any number of scans already along the trajectory. The first step is to find candidate scans. A scan is considered a candidate if: (1) the traveled distance between the query scan and the candidate scan, $d_{t,ij}$ (red line in Fig.

5.7-c), is greater than some distance parameter, and (2) the Euclidean distance between the query scan and the candidate scan, $d_{e,ij}$ (blue line in Fig. 5.7-c), is less than some distance parameter. The nearest K candidate scans are then taken, and ScanContext [68] is run to determine which scans are collected from the same or very similar viewpoints. ScanContext is a lidar-based global localization technique that compares two scans and estimates their viewpoint similarity by calculating the distance between their geometric descriptors. For better performance using ScanContext, each query, and candidate scan are first densified by aggregating the nearest neighbors. Finally, each of the candidate-query scan pairs is matched using any of the four scan-matching algorithms presented in the previous paragraph on scan registration.

The final graph that is being optimized is the combination of Fig. 5.7-a and Fig. 5.7-c. Similar to online SLAM, the final trajectory is optimized using the open source ROS-based optimization framework fuse which enables re-use of much of the same implementation codebase as is used in online SLAM.

Fig. 5.8 illustrates the results of the pose refinement. Here, the lidar map from online SLAM is shown in greyscale, the top two maps show the parking garage dataset results, the bottom two maps show the bridge dataset. The left column shows the trajectory before (red) and after (green) refinement, and the right column shows a more zoomed-out view of the trajectory with all the loop closure measurements displayed in color, where each color is a different loop closure measurement.

Trajectory Generation

The standard trajectories output from SLAM are generally not sufficient on their own for many of the remaining components of the proposed SLAM-centric inspection. The map generation process, for example, requires a high-rate non-drifting trajectory. Regardless of the SLAM method used for inspection, most SLAM methods (including the one used here) calculate two types of trajectories. First, a high-rate open-loop (OL) trajectory is estimated, and then a refinement is run, which closes loops but is run on a more low-rate trajectory, which is called the closed loop (CL) trajectory. The OL trajectory is commonly at IMU rate, while the CL trajectory is commonly at camera or lidar rate. For this implementation (as discussed in the previous sections), the OL is at IMU rate (200 Hz) while the CL trajectory is at lidar/camera rate (20 Hz). Examples of popular SLAM frameworks that do this are ORB-SLAM [105], LVI-SAM [130], LIO-SAM [129] and VINS-MONO [117]. For generating an inspection map (Section 5.3.1), or extracting image pose data for defect detection (Section 5.3.2), the best of both are needed: a globally consistent trajectory, that is also high rate.

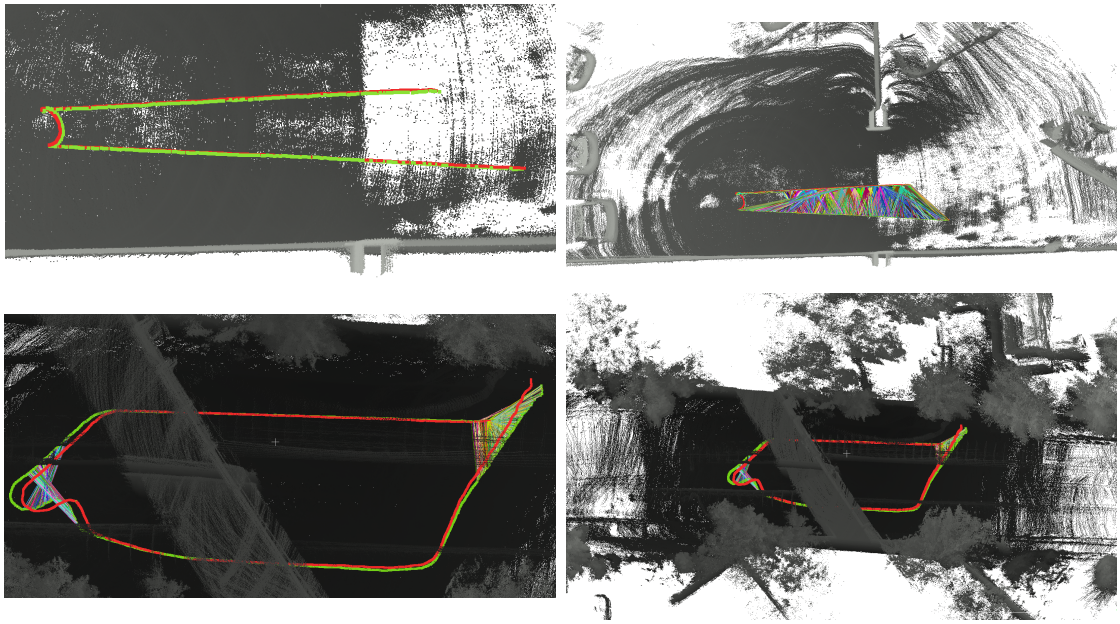


Figure 5.8: Pose refinement results for parking garage (top) and bridge (bottom) datasets

To calculate the optimal high-rate closed loop trajectory, the proposed approach starts with the CL poses. Between each CL pose, the OL poses need to be added to densify the trajectory. This can be done by using only the relative transforms between each OL pose and applying that to the first CL pose in the window. If the poses remained like this, the final trajectory would actually be discontinuous because the OL poses would drift, and therefore, the last pose in the window would not correspond to the end CL pose in the window. To correct for this, a linearly interpolated pose correction is added to each OL pose which is calculated using the start and end CL poses for each segment. Figure 5.9 illustrates this difference. In both figures, the magenta points are the CL poses, and the red points are the uncorrected OL poses. The left image shows the corrected high-rate poses in orange without interpolating the corrections. The right image shows the corrected high-rate poses in green with interpolated corrections.

Map Generation

One of the core novelties of this Chapter is that the proposed method de-couples the map generation from localization. This was first introduced in Thoms et al. [139]. In this collaborative work, the de-coupled map generation was solely my contribution, but it was

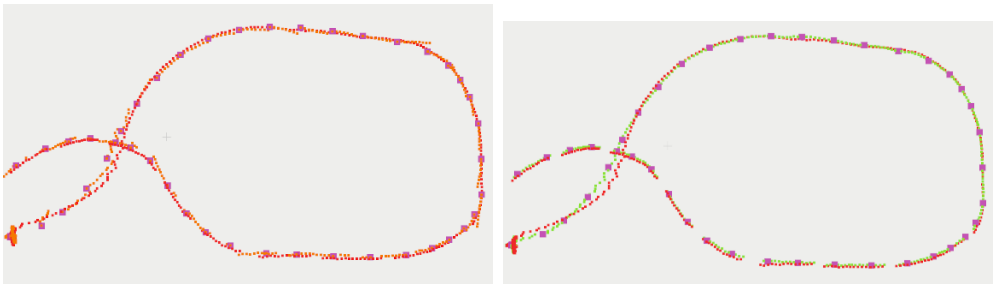


Figure 5.9: Calculating high-rate closed loop poses for the final trajectory generation. Left image: the corrected high-rate poses in orange without interpolating the corrections. Right image: the corrected high-rate poses in green with interpolated corrections.

applied to data from a surface vessel by the other authors. Lidar-based SLAM methods always generate a SLAM map that could theoretically be used for inspection. However, these maps are specially generated for the purpose of localization and not inspection, which would significantly degrade inspection results should they be directly used. To improve scan-registration performance and to remain computationally tractable, lidar-based odometry methods (including feature-based [157, 128, 151, 129] and direct [107, 111] methods) generally down-sample and discard volumetric data. Feature-based methods, which extract and use only the most salient points before scan registration [29], retain a small subset of the total volumetric data collected with environmental structure lost in the process. Direct lidar odometry methods, which attempt scan-registration on dense clouds, suffer from heavy down-sampling, with even state-of-the-art direct lidar odometry [29] voxelizing lidar scans with 0.25 m resolution. Consequently, SLAM map density is reduced, and environmental structure is lost. Inspection maps, in contrast, prioritize high point density and low map noise to improve semantic labeling [92], surface reconstruction [15], and point cloud segmentation and classification [50]. Achieving inspection map performance, thus, directly conflicts with SLAM performance requirements, justifying a separate process for inspection map generation.

The proposed inspection map generation process is referred to as *MapBuilder*, and all code is publicly available at `github.com/nickcharron/3d_map_builder`. *MapBuilder* allows any number or type of volumetric data to be combined into a single map, with noise removal and scan cropping techniques available at various stages along the map-building pipeline. Moreover, *MapBuilder* can be used with any SLAM algorithm, provided the algorithm provides access to estimated poses. Figure 5.10 illustrates the *MapBuilder* pipeline as a block diagram, with details on *MapBuilder*'s input, processing, and filtering following.

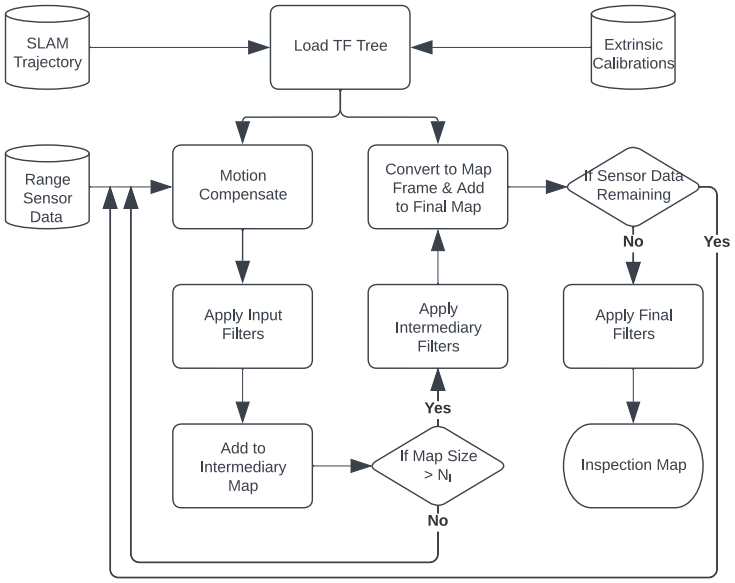


Figure 5.10: Diagram illustrating the overall workflow for the *MapBuilder*

MapBuilder receives input from three separate modules: *Extrinsic Calibrations*, *SLAM Trajectory*, and *Range Sensor Data*. Extrinsic calibrations include the set of rigid body transformations relating all range sensors to a common reference frame. SLAM trajectory includes the full set of time-stamped poses. For this thesis, the output trajectory defined in Section 5.3.1 is used for maximum accuracy; however, any SLAM trajectory can be used.

First, extrinsic calibrations and the SLAM trajectory are combined into a transformation tree (TF Tree) data structure [43]. Conceptually, a TF Tree originates from a single parent node, representing an arbitrarily defined frame common to all sensors, with edges representing time-stamped coordinate transformations between frames. Once constructed, the TF tree may be queried at the time of data capture for any sensor frame. If this time does not coincide with any time-stamped transformations stored in the TF tree, spherical linear interpolation [131] is used, allowing this framework to handle scans that are asynchronous to the SLAM trajectory, which is the case if any 3D sensors are used for map generation but were not used for SLAM.

Prior to transforming any lidar scans to a global frame, scans are optionally motion-compensated if each 3D point contains a time value. Motion compensation again uses the TF tree to query a transformation for each timestamp, aggregating them back into the lidar scan frame. Note this optional processing is only effective when the input trajectory is sufficiently dense (e.g., using pre-integrated IMU poses).

Volumetric data is optionally cropped and filtered using various pointcloud filters defined in PCL [123] as a means of noise removal or map cleanup. Filters may be applied at three stages of the map-building process to suit performance needs. First, filters may be applied directly to each scan, with the *Dynamic Radius Outlier Removal* (DROR) [27] filter specifically designed to remove outliers from a single lidar scan. At this stage, the cropbox filter may also be applied to remove distant, noisy measurements and near measurements of the robot itself or the inspector. In the second stage of filtering, scans are aggregated into intermediary maps for another round of filtering. When applying filters to intermediary maps, the performance of the *Radius Outlier Removal* (ROR) [123] filter improves over its application to a single scan, given the denser set of points to discriminate inliers versus outliers. In the last stage, filters may be applied to the aggregate of all scans, where the voxel grid filter may be used to reduce final surface noise. For a more detailed description of the *MapBuilder* implementation, including full algorithm description, refer to Thoms et al. [139].

Fig. 5.11 illustrates the maps generated using the *MapBuilder* for both test datasets.

5.3.2 Defect Detection

Fig. 5.12 shows the full defect detection workflow, including inputs and outputs of each step in the pipeline. There are two major components to this. The first is extracting images and metadata from various camera streams. The second is labeling these images with defects of interest in such a way that they can be used for map labeling and quantification (see Section 5.3.3). The next sections will dig into the proposed solution for accomplishing both these tasks.

Image and Pose Extraction

The goal of this step is to identify a versatile method to extract image information that will be useful for gathering inspection data from a moving robot that is collecting continuous image data from several cameras. Ideally, high-quality images of defects of interest are desired, as well as full image coverage of the environment so maps can be completely colorized and visually realistic 3D maps can be generated for inspectors. Duplicate images showing similar views are to be minimized since this will result in excessive data processing and storage requirements. This, therefore, creates a trade-off between the quantity of data and view coverage. There are various works that tackle the problem of coverage planning which could be useful to solve this problem, or machine learning can be used to select

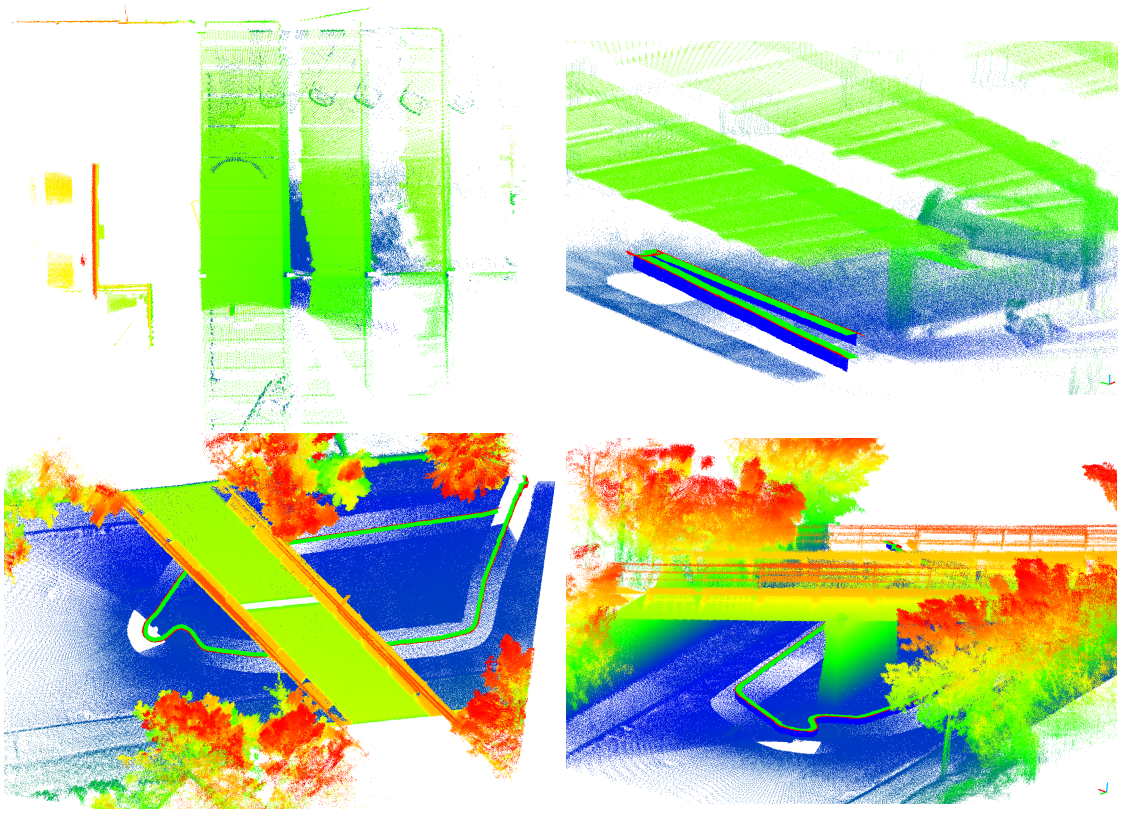


Figure 5.11: Lidar map results using the MapBuilder with the vertical lidar. Top row: parking garage dataset, bottom row: bridge dataset, left column: plan view, right column: oblique view

photos where defects are centered in the image. Since the core contribution of this chapter is to define an overall inspection workflow, these options are not investigated, instead a simple solution is selected that works well enough for a proof of concept.

The image selection solution is to use the trajectory to extract images given a user specified minimum rotation or translation in between each consecutively selected image. This can be done on each camera stream, allowing the user to select appropriate spacing for each camera, which may be used for different purposes. For example, the low-resolution and high FOV camera used in this work is mostly used for colorizing the 3D map, and therefore, images should be extracted at a higher frequency compared to the high-resolution camera that is mainly for capturing defects. Furthermore, the trajectory can be used to filter out images that may have high distortion. To do this, rotation rates are computed

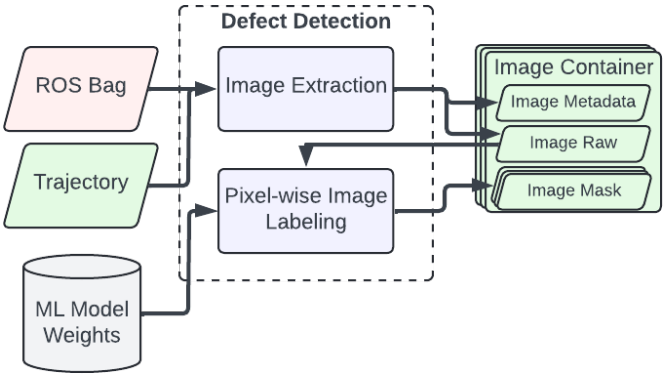


Figure 5.12: Diagram showing the defect detection component of the inspection workflow

using surrounding poses and the user can set defined thresholds, which can be tuned based on the exposure time and resolution of your camera.

For each extracted image, poses are extracted from the input trajectory and stored if the pose of that camera has changed more than the input thresholds since the last image for that camera. For each raw image that is stored, an image container is created which holds all information relevant to that image. This includes a metadata file that stores information like the capture timestamp (needed to lookup pose), the frame ID (needed to lookup calibrations), and if any image manipulations were done, such as rectification. This container is also used to store any image masks that were extracted from that raw image (see Section 5.3.2).

Fig. 5.13 shows example data after the pose extraction. The top two images are the parking garage dataset, where the left shows an example image from the front-facing camera, which is displayed in the map frame showing the true location and orientation relative to the map. The upper right image shows all images displayed in the map frame from all four cameras onboard the robot. The bottom image shows a similar view from the bridge dataset, where all four images extracted from the same timestamp are displayed, and the pose of the front camera is shown in the map. These are the images that will be used to extract defects on the bridge and their poses shown here will then be used to transfer the defect information onto the map.

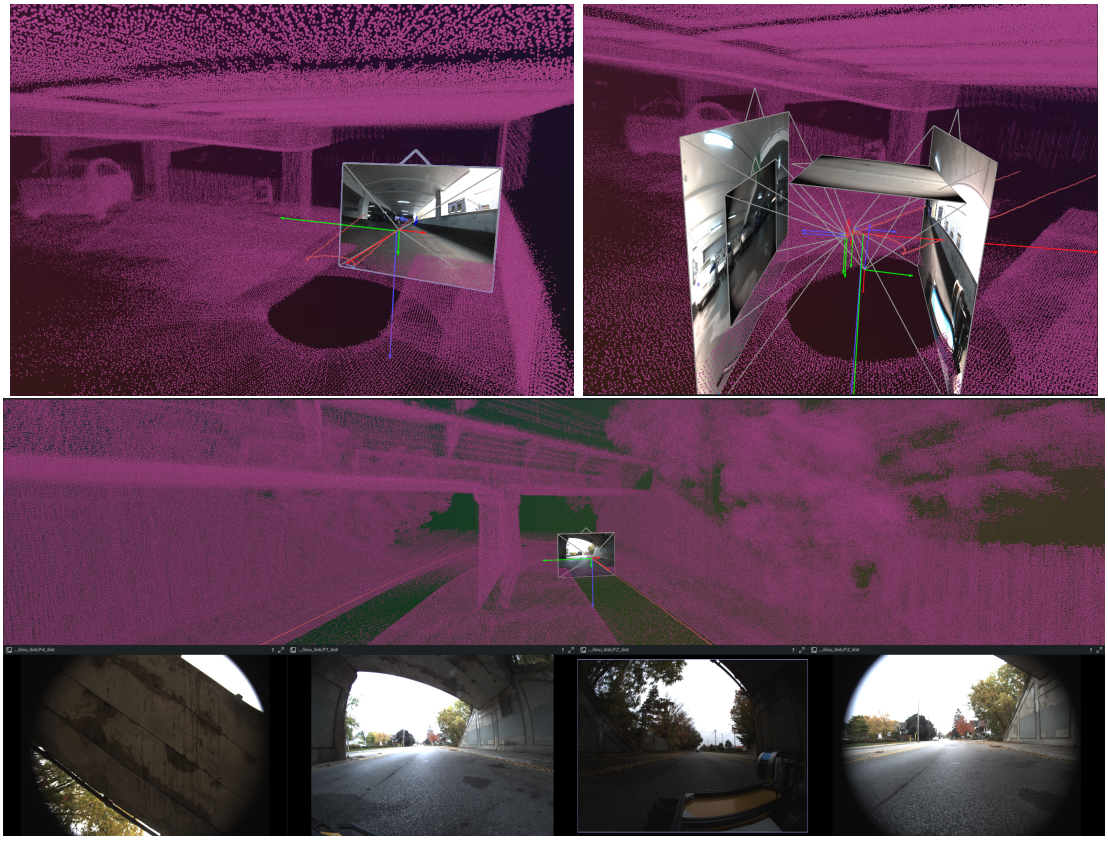


Figure 5.13: Example pose and image extraction from parking garage (top) and bridge (bottom) datasets

Pixel-wise Image Labeling

This section refers to defect labeling as the process of labeling defects in images extracted during the image extraction step. Although this step is a key component in the inspection pipeline, this component is designed to be relatively general in order to ensure the pipeline can be generalized to various inspection tasks. For example, defect labeling for bridge and parking garage inspection would mostly consist of detecting cracks, spalls, and delamination of concrete. For new construction projects, perhaps the defect labeling would be quantifying the number of piping in order to calculate percent complete for assembling piping. The only requirement for the labeling is that pixels be labeled on the image to quantify some class type.

There exists ample work in the space of defect labeling in images. This thesis does not

claim any novelty in defect labeling, instead common tools are used or tools generated by collaborators. Three defect detection methods will be highlighted in this work.

First, during some collaborative work with Evan McLaughlin, it was shown that concrete delamination outlines can be extracted from infrared images using a specially trained ML model [92, 91]. In this work, DeeplabV3 [30] was trained with a small number of images collected from a mobile platform, using a MobileNetV2 [57] as a feature extractor. For results, please refer to [91]. Since the datasets presented in this Chapter did not have infrared cameras, results from this work are not shown, but the reader is encouraged to see the work presented in [92] and [91] to understand how such results can be integrated into this workflow.

Second, for a more general and automatic defect extraction technique that can run on the datasets used in this chapter, Defect Instance Segmentation YOLO, or DIS-YOLO [156] is used. DIS-YOLO is a fully convolutional model that uses the YOLOv3 [119] backbone and is trained to simultaneously detect and segment three concrete surface defects, including cracks, spalling, and exposed rebar. The model was trained on the combination of three open datasets for bridge inspection and one dataset from the Hong Kong Highways department, resulting in 1,433 labeled images.

Third, for a more human-in-the-loop approach, some of the images are also processed with the popular Segment Anything Model (SAM) [71]. SAM is the largest segmentation model to date, trained with over 1 billion masks on 11 million images. SAM is also designed to work using a prompt, making this model extremely versatile for any sort of human-in-the-loop segmentation task. With a small amount of human work, any sort of defects or objects in an image can be quickly segmented.

Regardless of the methods used, all defect labeling methods must generate one or more segmentation maps. These segmentation maps get added to the image container for the respective raw image as illustrated in Fig. 5.12. Once all segmentation masks are generated, the pixel labels can be back-projected for each mask, as well as the raw images, to the 3D map presented in Section 5.3.1. This process will be discussed in the following section.

Fig. 5.14 and Fig. 5.15 show example defects extracted from the bridge dataset. In this dataset, SAM was used to semi-automatically select defect areas to be extracted and quantified. This is a good example of an inspection where it is beneficial for labeling to be done with a human in the loop. Since the bridge is in such poor condition, explicit defects are not obvious and likely would be up to an inspector or the owner to decide on what exactly needs to be repaired. In this dataset, SAM is used with a few manual positive and negative labels input by the inspector. This is used to segment spall areas, as well as a repaired section which is being called a delamination. In Figures 5.14 and 5.15, the left

column shows the raw image, and the right shows the raw image with the defect labeled in red, which will then be used in the defect map labeling and quantification discussed next.



Figure 5.14: Bridge dataset example defect labels from the front-facing high FOV camera. Image IDs from top to bottom: C1I10, C1I26. The left side shows the raw images, and the right shows the image with the defect labeled in red.

5.3.3 Map Labeling and Defect Quantification

Given a 3D pointcloud map made specifically for inspection, a precise and dense trajectory, and a set of labeled images, all such information can be combined into a single data structure that enables the following key features:

- A to-scale map allowing all data to be geographically cataloged for easy localization relative to the structure.
- An easy means to visualize all inspection data in one place from a computer or in 3D using a VR device.

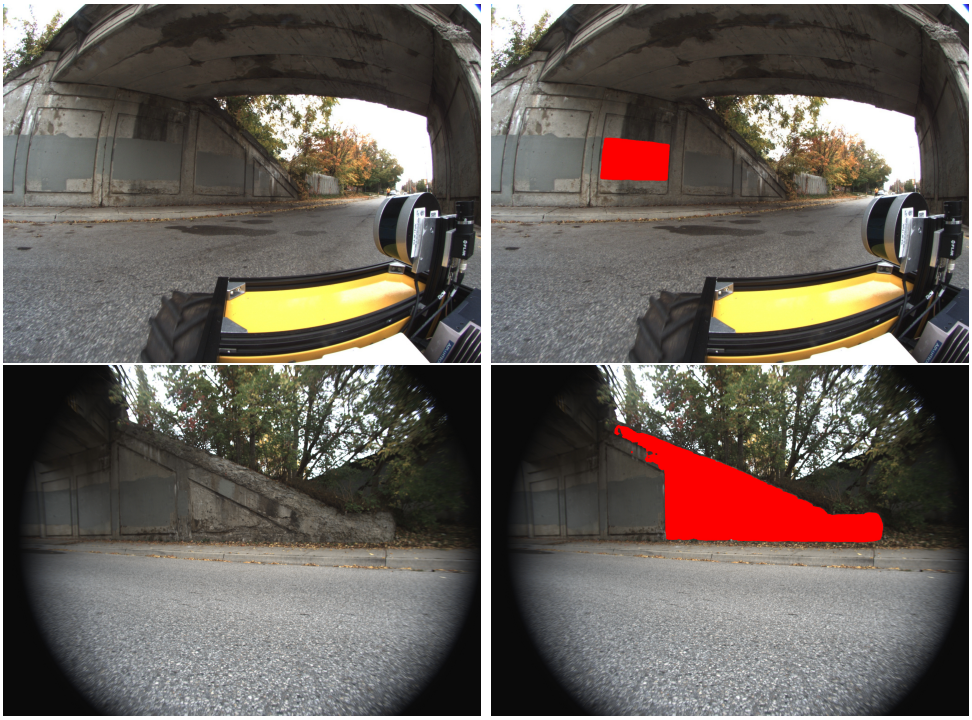


Figure 5.15: Bridge dataset example defect labels from the rear-facing camera (top) and the front-facing high-resolution camera (bottom). Image IDs from top to bottom: C2I20, C3I27. The left side shows the raw images, and the right shows the image with the defect labeled in red.

- A way to combine several observations of the same objects from different viewpoints.
- A way to quantify defects that may have been partly observed from different images.
- A way to identify specific defects between consecutive inspections.

These key features are what make the proposed SLAM-centric inspection system so valuable to inspectors. This section describes all the sub-components that allow such data to be combined into a map in an efficient manner and how to extract important inspection data from such a map, including defect/object sizes and changes in size over time.

Fig. 5.17 shows the full defect detection workflow, including inputs and outputs of each step in the pipeline. The first two steps, which can run in parallel, are Map Colorization and Defect Map labeling. Since defects are stored as image masks, the same exact algorithms can be used to transfer defect information to the map as regular RGB information. Both

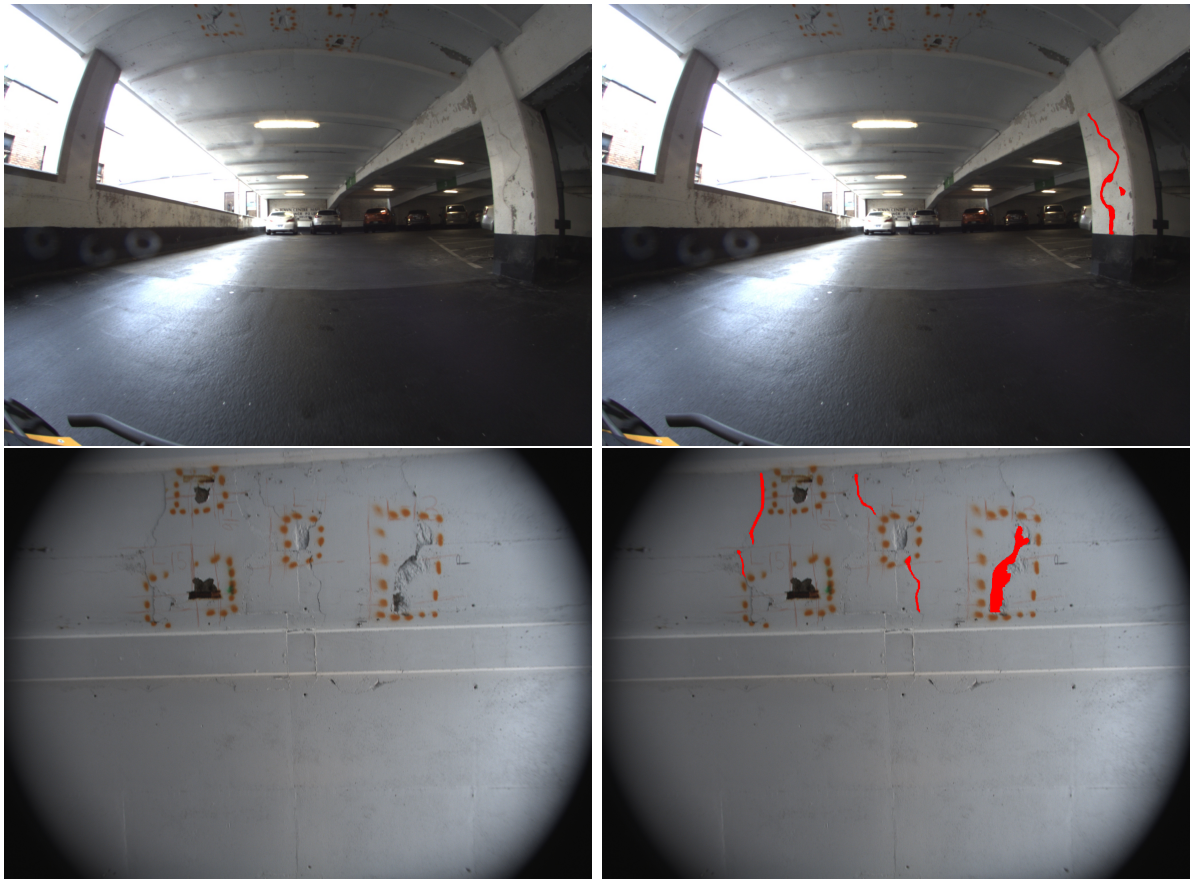


Figure 5.16: Garage dataset example defect labels. Image IDs from top to bottom: F1I2, F4I4

these tasks are grouped into the Section 5.3.3. The output of these two sub-components is a 3D inspection map where points have their own labels describing their color and whether they belong to some defect or other class from the input masks. Given a labeled map, points can be grouped into objects/defects, and their sizes can be quantified (Section 5.3.3). Finally, if previous inspection maps exist of that same structure, these maps can be aligned, and objects can be tracked over time to detect change in sizes or to quantify new objects not existing in previous inspections (Section 5.3.3).

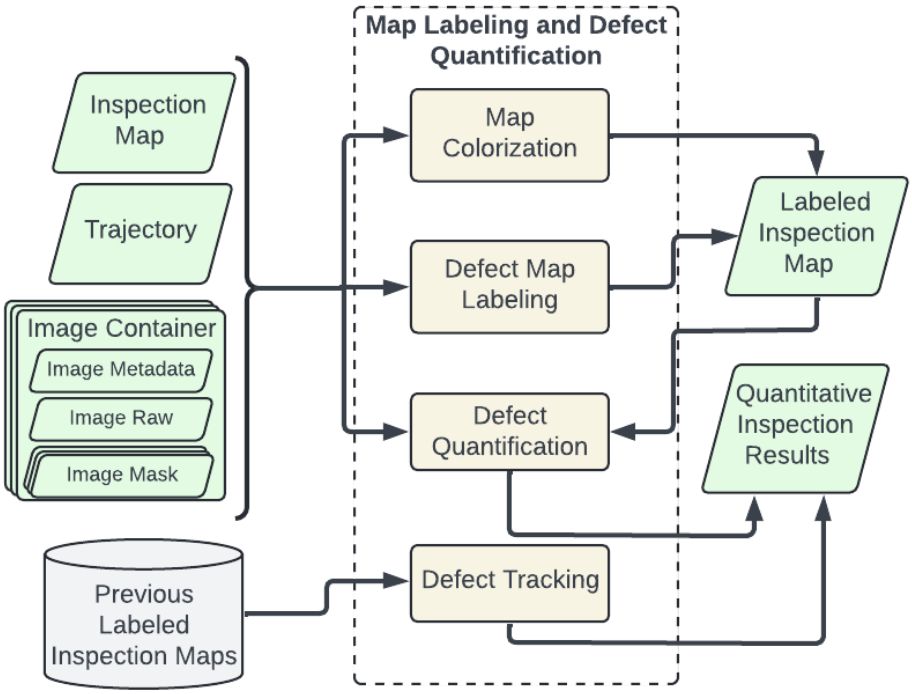


Figure 5.17: Diagram showing the map labeling component of the inspection workflow

3D Map Labeling from Images

Map labeling is defined here as the process of taking images or image masks with known poses and transferring the information from the image pixels to the 3D pointcloud map. For regular images, transferring RGB image fields to the map to provide visual cues for the inspectors or even intensity fields from an infrared image that may be used to visually locate subsurface defects. For image masks, this includes transferring any defect or object classification field added in the defect labeling described in Section 5.3.2. In either case, the map labeling process is identical since it can be applied to any image type, regardless of the pixel fields that exist in the image data.

In computer vision and graphics, the task of colorizing a scene from an image can be broken down into two general approaches: projection, and raytracing. Projection involves taking 3D points and projecting them into an image plane using the camera model and then assigning the color of the projected pixel to that point. Raytracing does the inverse and

back-projects a pixel using the camera model and extends that ray to find the intersection with the map.

The projection method is computationally inexpensive and simple to implement; however, these methods cannot account for occlusions as there is no sense of 3D geometric relationships. Therefore projection can only work when an image and its corresponding 3D points have been captured from the same viewpoint. Autonomous vehicle applications usually implement this method since they color lidar scans from a short time window with its associated image [120]. Since the approach proposed in this thesis decouples the map generation from the remainder of the inspection pipeline, occlusions become a significant issue since the map points may have been generated from a wide range of viewpoints that differ from the image's viewpoint. These map points also may have been altered during the map creation, such as averaging points in each voxel during down-sampling. This means that it cannot be known which pixel captured which point during data collection since a point may be a combination of many other points captured at different times.

Raytracing eliminates the occlusion issue by computing the first part of the scene that comes into contact with the ray extending from the camera center. This problem is more common in the graphics community when rendering scenes given a predefined complex scene model, and some light source that reflects off this scene. Raytracing methods are better at understanding the geometric relationships of the objects in the scene and how light reflects off the scene, however, this process can be quite computationally expensive even for large GPUs. To accelerate the raytracing process, the graphics community relies on defining the scene geometry using complex data structures such as the bounding volume hierarchy (BVH) [94]. BVH works by breaking the scene into a hierarchy of relatively simple shapes, such as axis-aligned bounding boxes, and recursively finding intersections with these objects. As such, the scene must be divided into objects that have pre-defined volumes. Since inspection maps are an unorganized set of points, such methods cannot be applied without doing some point segmentation or clustering, which presents challenges of their own when applied to noisy inspection data. Using a brute-force raytracing algorithm that extends a ray and looks for contact with a pointcloud map would be too computationally expensive for the inspection use case, since inspection images need to be high-resolution and pointcloud maps can be quite large when inspecting large structures.

There is, therefore, a gap in the literature for a method to quickly label unordered pointclouds from images in an occlusion-safe manner. Yan et al. [149] attempt to solve this problem, motivated by the same inspection use case, by creating a custom projection method. In this work, the images being used to label the map are separated into grids, and only the points closest to the camera in each grid are labeled with that image grid. This method failed to provide promising results for the datasets in this thesis for a few

reasons. First, noisy maps cause issues because a few outlier points can cause an entire grid of pixels to be neglected. Second, inconsistent density in the maps means the grid size cannot be tuned to work well for the entire map. Third, edges of objects, specifically small objects, are not well labeled due to the discretization of the image into grids.

This work instead focuses on creating a highly efficient custom raytracing approach. The approach uses a few key concepts to drastically increase computation time without degrading accuracy. First, for each image, all points in the map are projected into the image. All points that do land on the image plane are removed from the map, resulting in a map that is orders of magnitude smaller. This decreased map size significantly decreases the time for each volume search performed during raytracing. Second, a hit map is created with equal dimensions to the image. This is a binary image where each pixel is true if at least one map point has been projected into that pixel. Raytracing then only needs to be performed for all pixels in the hit-map labeled as true. This significantly reduces the amount of raytracing steps needed per image. Raytracing is then performed normally, where a ray is extended from the camera center, passing through each pixel in the hit-map and extended until hitting the map. The final improvement which increases speed without degrading accuracy is how the ray is extended. In each increment, the ray is extended by a length equal to the distance to the next closest point in the map. This ensures the ray does not pass by any point during increments, while also not performing unnecessarily small increments when there are no map points nearby. A ray is considered to have hit the map if the end of the ray at that specific increment is within some threshold distance from a point on the map. When a hit has occurred, that pixel's color is applied to all points within the splatter threshold distance.

Figures 5.18, 5.19 and 5.20 show the results of running the above map labeling process on the defects from the bridge and garage datasets, respectively. These defects correspond to the same ones shown in Figures 5.14, 5.15 and 5.16 using the same inspection maps shown above. The left column on these figures show the map labeled using the RGB images, and the right column shows the defect maps labeled on the map. In the left column, only the points that are seen by the camera at that image time are labeled, and all other points are left black (note that these can later be combined to color the entire map). The right column shows the points labeled as defects in red, and all other points are in blue. In the bottom row of Fig. 5.20, the remainder of the map is also shown in gray to get a perspective as to where that section of the map is on the entire structure.

Multiple images

The above sections have only discussed how to label the map given a single image or image mask. Another important component is the process of combining data from

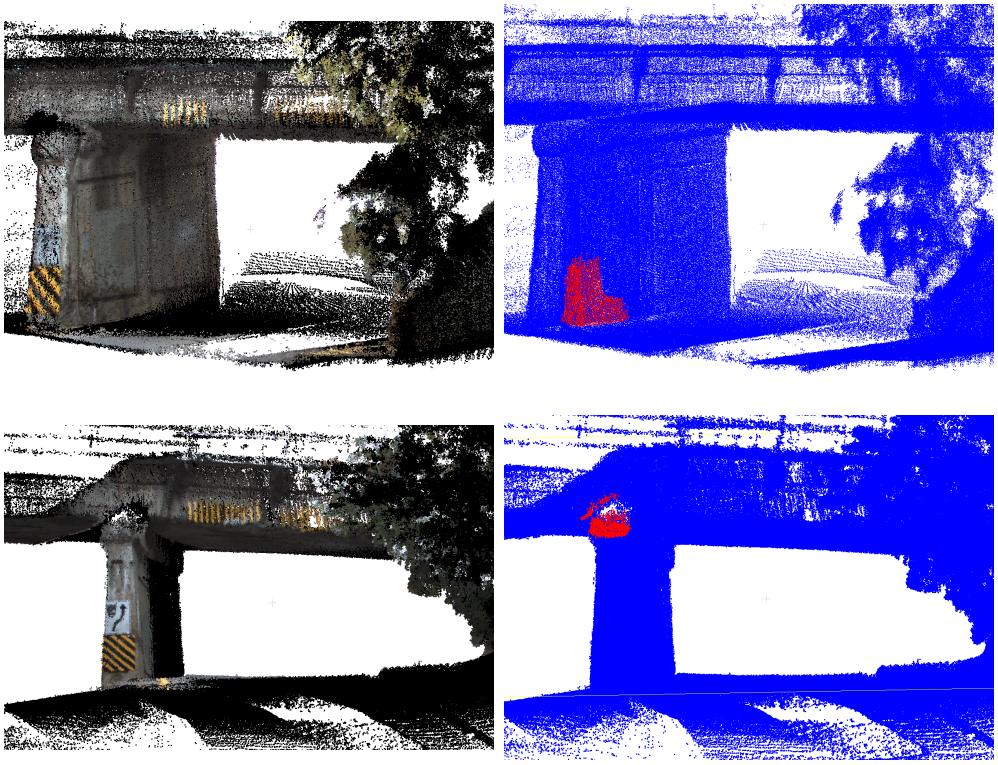


Figure 5.18: Bridge map labeling results from the front-facing high FOV camera. Image IDs from top to bottom: C1I10, C1I26. The left column shows the map colorized with the RGB camera, and the right column shows the map colorized with the defects

multiple overlapping images. The proposed implementation does this step differently for RGB images versus image masks. However, this may also differ based on the specific inspection task. For RGB image labeling, each time a point is labeled with RGB color, the distance from the camera is logged. If another image covers that same point, then the color is overridden if the new pixel was captured from a closer camera viewpoint. Other strategies could be used here such as blending, however, since the goal is not to get a photo-realistic rendering of the structure this method provided satisfactory results.

For labeling image masks, on the other hand, a more careful solution is preferred. Since models trained to detect defects or objects are never perfect, intelligently combining multiple detections from different viewpoints can help improve the certainty on that object. To take advantage of this idea, this implementation proposes to use a Bayesian log odds updating method on all classes defined in the image masks. Eqn. 5.5 presents the update rule for each point in the map when it gets labeled by a single image mask. In this

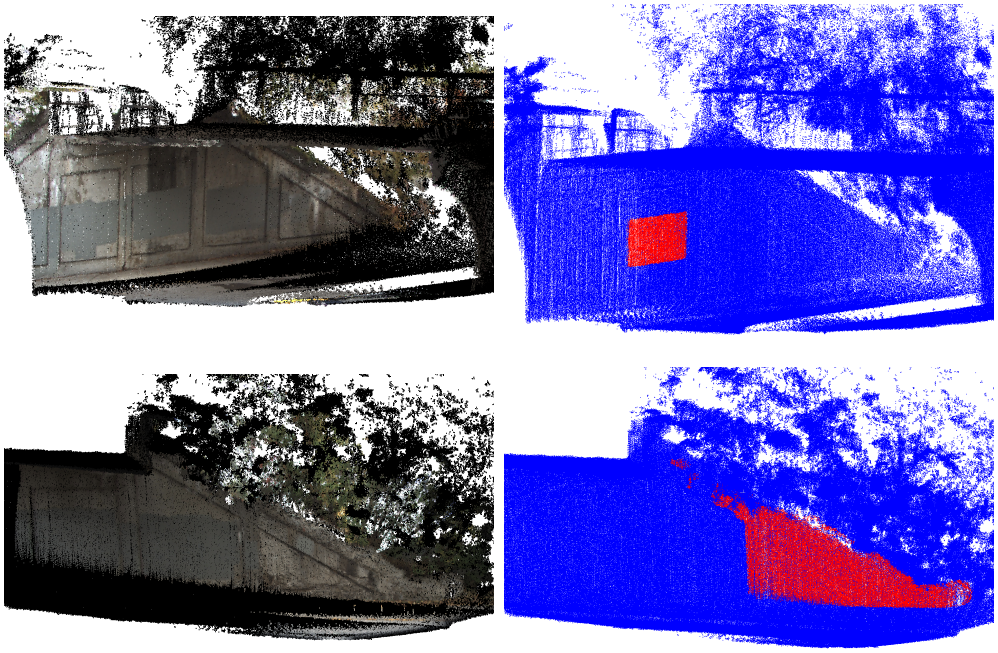


Figure 5.19: Bridge map labeling results from the rear-facing camera (top) and the front-facing high-resolution camera (bottom). Image IDs from top to bottom: C2I20, C3I27. The left column shows the map colorized with the RGB camera, and the right column shows the map colorized with the defects

formulation, the probability of a point belonging to a class is tracked. $P'_{k,i}$ is the update probability of a point i belonging to a particular class k. P_{dk} is the probability of a detection model correctly detecting a class with a single image, and P_0 is the probability of a point belonging to any class given no measurements, which is set to $P_0 = 0.5$. Since no ground truth labels were obtained for the datasets presented in this work, accuracy improvement from combining multiple images cannot be proven and is therefore not shown.

$$P'_{k,i} = \mathcal{L}^{-1}[\mathcal{L}(P_{dk}) + \mathcal{L}(P_{k,i}) - \mathcal{L}(P_0)]; \quad \mathcal{L}(x) = \log \frac{x}{1-x}; \quad \mathcal{L}^{-1}(x) = \frac{\exp x}{1+\exp x} \tag{5.5}$$

Defect Quantification

Given a single labeled map, it is possible to quantify the sizes of objects or defects belonging to each class. This implementation was first introduced in collaboration with Evan

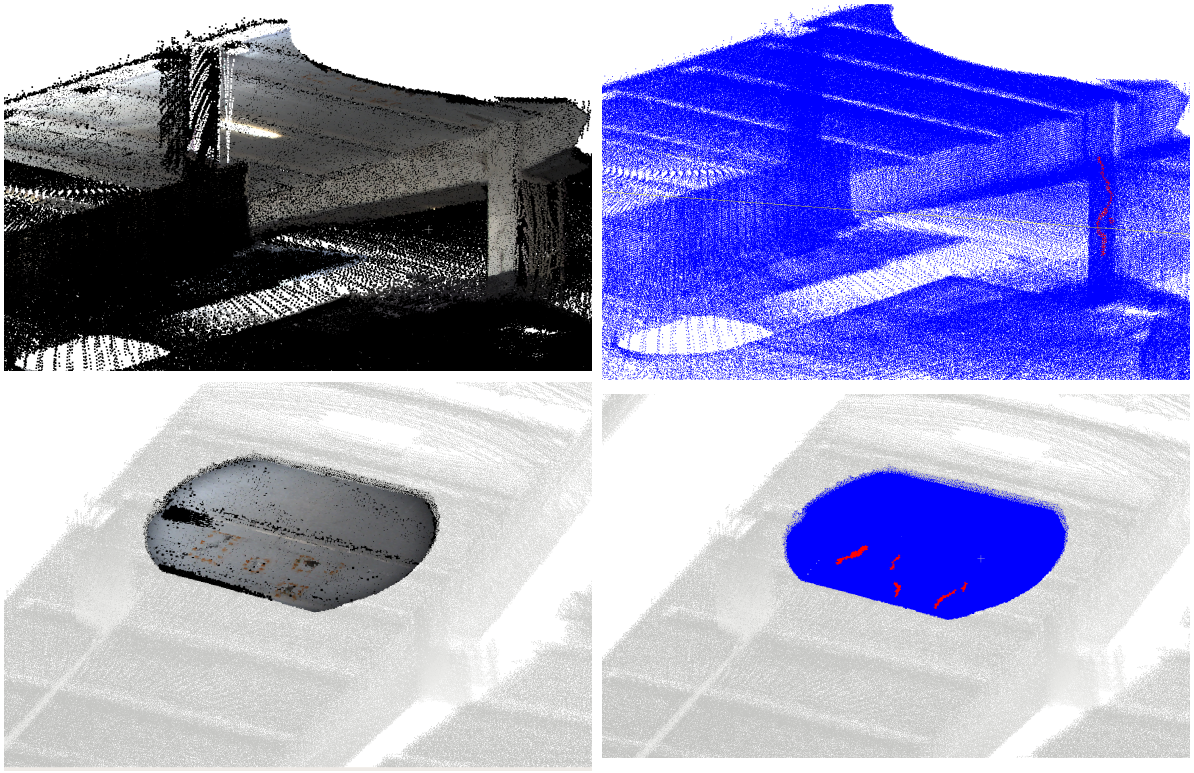


Figure 5.20: Garage map labeling results. Image IDs from top to bottom: F1I2, F4I4. The left column shows the map labeled with the RGB images, and the right column shows the map labeled with the detected defects.

McLaughlin in [92]. The approach can be summarized as follows. First, all points in the 3D map not belonging to the class of interest are removed. Next, points are separated into groups (or objects) using Euclidean distance segmentation. Assuming each defect sits along a single plane (which is the case for the defects detected in this thesis), the total area is quantified for each object by projecting points to the best-fitting plane and calculating the convex hull, and using this to estimate the area. For a detailed explanation of this process, refer to [92].

Fig. 5.21 shows the segmented defects for the bridge dataset, while Fig. 5.22 shows these defects situated on the entire pointcloud map. Having the ability to display defects on a complete 3D map can be extremely helpful for inspectors to visualize and get a perspective of all defects, including their sizes and locations relative to each other. Table 5.1 presents all defect quantification results for the bridge dataset. In this dataset, all

defects are marked as either spalls or delaminations. Quantification metrics in Table 5.1 include the concave hull area and 2D bounding box dimensions (width, height, area), which can be useful to try to estimate overall patch size at repair time.

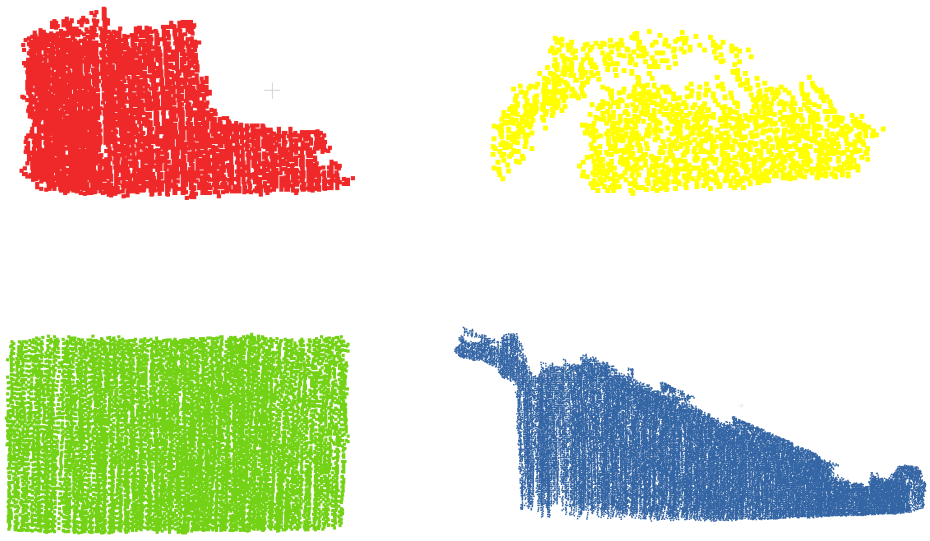


Figure 5.21: Bridge dataset segmented defects from each image shown in Fig. 5.16. See Table 5.1 for color labels

Table 5.1: Defect quantification summary for bridge dataset

Image Label	Defect Id	Type	Hull Area(m^2)	BBox Width(m)	BBox Length(m)	BBox Area(m^2)	Color
C1110	0	Spall	1.95	1.39	2.45	3.40	Red
C1126	0	Spall	0.39	0.73	0.99	0.72	Yellow
C2120	0	Delam	2.61	1.35	2.26	3.05	Green
C3127	0	Spall	7.91	2.51	6.88	17.26	Blue

Fig. 5.23 and Table 5.2 below show the same information as above but for the parking garage. For the parking garage dataset, however, the defects labeled were cracks. For crack quantification, the bounding box area is also quantified, but crack length is also crudely estimated by simply taking the largest distance between any two points in the defect cloud.

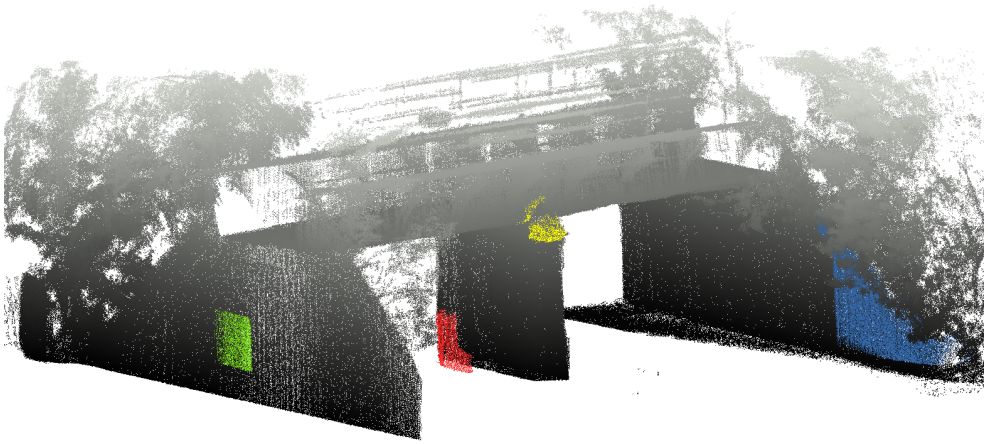


Figure 5.22: Showing all bridge defects labels on the map using different colors

Since these datasets were collected in public spaces, unfortunately, no ground truth defect sizes were collected to compare results. Instead, this chapter provides a proof of concept for the overall approach to prove the viability of adopting such a technology in practice and highlights the challenges of doing so. For more analysis on quantification accuracy for such an approach, refer to the original work that first proved the approach in [28].

Table 5.2: Defect quantification summary for garage dataset

Image Label	Defect Id	Type	Length (m)	BBox Width(m)	BBox Length(m)	BBox Area(m^2)	Color
C112	0	Crack	1.39	0.32	1.35	0.43	Red
C112	1	Crack	1.4	0.33	1.36	0.44	Green
C414	0	Crack	0.65	0.41	0.5	0.20	Green
C414	1	Crack	0.38	0.22	0.3	0.06	Yellow
C414	2	Crack	0.26	0.09	0.24	0.02	Blue
C414	3	Crack	0.37	0.09	0.36	0.03	Purple
C414	4	Crack	0.19	0.08	0.17	0.01	Brown



Figure 5.23: Garage dataset segmented defects from each image shown in Fig. 5.16. See Table 5.2 for color labels

Defect Tracking

Having a labeled map with defects and sizes unlocks the ability to automatically and precisely track defect growth over time if defects can be associated between maps generated using different inspections. This can be especially challenging for the inspection use-case because (1) inspections present challenging scenarios such as lack of texture, poor lighting, and reflective or moving objects in the scene, and (2) measuring defects over several years may include significant changes in the environment.

Multi-map alignment methods have been proposed in the literature, by running re-localization in a prior map, or multi-session SLAM [93]. In both cases, the existing solutions rarely take advantage of lidar and camera data to align the maps. Re-localization is explicitly known to be challenging under significant environmental change, rendering this method unlikely to be suitable for inspection over several years. Furthermore, multi-session SLAM requires the SLAM front-ends to be rerun on all datasets that are to be aligned which requires all original data to be kept and presents challenges if hardware or data formats change between the years as SLAM is often tightly coupled with the sensor specifics as shown in the SLAM work with Project Aria [134].

To solve this problem, collaborative work led by Jake McLaughlin takes a SLAM-agnostic approach to map alignment while combining both lidar and camera data for map alignment. This method presents a non-rigid map alignment that takes into account different levels of localization drift between the SLAM maps generated potentially years apart. Though the design was Jake's contribution, the ideation was born from this work and is briefly discussed here to close the loop in the E2E inspection system design. For more details on map alignment and defect tracking, see [93].

5.4 Conclusions

This chapter presents, for the first time in literature, a full E2E system for visual inspection of infrastructure from a mobile mapping lidar-camera system. While previous chapters motivate the requirement for such a SLAM-centric inspection system, this chapter outlines a viable solution to perform this E2E. Each system component is described in detail, with results presented for each step on two example datasets: a parking garage and a bridge.

The proposed system starts with a trajectory and map generation step. This first runs SOTA multi-sensor online SLAM to generate a first trajectory. This trajectory is then refined offline to get the best possible trajectory, which is globally consistent, accurate, and high-rate. Finally, an inspection map is generated using this trajectory, which is explicitly designed for generation inspection maps, which differ in requirements compared to SLAM maps.

The second major component in the proposed inspection system is the defect detection. First, images must be extracted from the dataset, ensuring adequate coverage, and then images need to be labeled with defects. Two example approaches were shown for image defect labeling: (1) a human-in-the-loop method that utilizes SAM to detect spalls, and (2) a fully automated labeling method DIS-yolo to detect cracks.

The third and final major component of the proposed inspection system is map labeling and defect quantification. Image masks and raw RGB images are used to label the final inspection map using a novel raytracing technique that is accurate, occlusion-aware, and computationally inexpensive. Given a single labeled map, defects are then segmented into individual instances and quantified in real-world coordinates. These quantities can then be compared to previous inspection results to estimate defect growth over time accurately.

Given the use cases and results presented in this chapter, it can be concluded that the full E2E SLAM-centric inspection system presented can adequately solve the goals set at the start of this chapter. That is, high-quality 3D maps can be generated and used to track and quantify various defects of interest to the infrastructure inspection community. Implementing such a technology in practice has the unprecedented capabilities of accurately tracking all defects in a structure at scale with little human effort. This can unlock many new opportunities, such as significantly more sophisticated asset degradation models generated using rich quantitative data collected at frequent intervals.

Chapter 6

Conclusions

The research presented in this thesis explores the transformative potential of mobile robotics for automating the visual inspection of infrastructure, specifically through a SLAM-centric approach. This work contributes significant advancements across multiple domains within civil engineering and robotics by addressing the challenges of efficiency, accuracy, and repeatability in infrastructure maintenance practices. The following sections will summarize this study's key contributions and implications, highlighting the limitations and recommended directions for future work.

6.1 Key Conclusions

The following is a list of key conclusions that can be drawn from the work presented in this thesis:

- **A SLAM-centric inspection system is viable and effective**: The thesis demonstrates that SLAM can be the core of an E2E robotic infrastructure inspection system, enabling precise 3D mapping and localization necessary for visual inspections.
- **Multi-sensor system design is crucial for dual goals**: Effective inspection requires a sensor configuration optimized not just for SLAM performance, but also for capturing high-quality inspection data.
- **Non-overlapping sensor calibration was successfully addressed**: A novel target-based extrinsic calibration technique using motion capture allows for accu-

rate calibration of sensors with non-overlapping fields of view—a common challenge in real-world inspection setups.

- **Inspection maps must be decoupled from SLAM maps**: While SLAM maps are good for localization, inspection tasks require high-quality, dense, and low-noise maps, which are not necessarily prioritized when building maps for SLAM. The system includes a separate mapping module that refines the SLAM outputs for this purpose.
- **3D defect labeling, quantification, and tracking is achievable**: Detected defects from 2D images are projected into 3D space, enabling the tracking of defects over time and precise measurement of quantities such as crack length or spall area.
- **The system works in real-world scenarios**: The system was validated on infrastructure assets like bridges and parking garages, demonstrating its practical effectiveness and repeatability.
- **The pipeline enhances repeatability, objectivity, and safety**: By integrating modern machine learning with precise SLAM and map alignment, the system eliminates many of the subjectivities and inconsistencies of manual inspections. By creating a platform-agnostic sensor suite, any mobile platform type can be used to perform inspections remotely and therefore eliminate inspector dangers.

6.2 Significant Contributions

Broadly, this thesis presents an original system-level contribution that attempts to improve the way inspections are performed in regard to several metrics, including cost, safety, and precision. This thesis presents the first work that solves the E2E visual inspection task using a SLAM-centric approach. The SLAM-centric approach has been demonstrated to be the key element that enables a defect to be located in 3D, quantified precisely, and tracked over time in a scalable manner that unlocks the ability to perform more frequent inspections of large pieces of infrastructure.

Several specific contributions are presented in this thesis that are deemed necessary to unlock such an E2E SLAM-centric system. The following are the main areas of scientific contributions, along with several minor contributions that are described in the Chapters themselves:

Inspection System Design: A robust multi-sensor inspection system was developed, emphasizing sensor layout optimization for SLAM and inspection. The design ensures adaptability across a wide range of mobile platforms and infrastructure asset types, showcasing the versatility of the proposed solution.

Calibration Innovations: The development of a target-based extrinsic calibration method for non-overlapping camera-lidar systems resolves a significant bottleneck in multi-sensor platforms. This method ensures repeatable, measurable, and high-precision calibration, regardless of sensor field-of-view overlap.

Mapping For Inspection: The research advanced multi-sensor SLAM by combining real-time online localization with offline map refinement and a decoupled inspection map generation process. This dual approach ensures immediate usability in the field while maintaining the capacity for high-quality data post-processing and a resulting map that can be specifically tailored for optimal inspection results without degrading SLAM performance.

Automated Defect Assessment Pipeline: By integrating machine learning-based defect detection with the 3D mapping results, the automated pipeline enables the detection, localization, quantification, and temporal tracking of defects. This is enabled through a novel raytracing map labeling technique, which is both precise and computationally efficient. Viable solutions to several key components required to generate quantitative inspection data are also presented.

6.3 Sources of Uncertainty

Due to the complexity of the proposed inspection system, there are many sources of uncertainty that can contribute to errors in the final inspection results (i.e., defect sizes). This thesis does not attempt to quantify how much these errors affect the final inspection results, but it is worth listing these sources of uncertainty in order to help inspectors and future researchers understand the limits of such a system and where improvements can be made. The following are the most significant sources of uncertainty:

- **Lidar accuracy**: The scale of the defects are determined by the map generated from the lidar data. Since lidars typically have errors on the order of a few cms, this error can propagate to the final defect sizes.
- **Map density**: lidar maps can be relatively sparse, depending on the data collection details. Since defect areas and dimensions are measured by outlining points on the

lidar map, the granularity of the outline is affected by the density of the map which can affect defect measurement accuracy.

- **Trajectory accuracy**: the lidar maps are built given the trajectory estimate, which will have error that will propagate to the inspection results.
- **Calibration**: any errors in the extrinsic or intrinsic calibrations will result in errors when performing ray-tracing to combine the defect information with the lidar map.
- **Time synchronization**: Any error in the time synchronization strategy will not only degrade the SLAM performance, but it will also introduce error in the ray-tracing when trying to combine defects with the map.
- **Defect labeling**: Any error in the defect labeling will also propagate to error in the defect quantification.

6.4 Limitations and Directions for Future Study

SLAM: Robust localization and mapping is still the main challenge to such an inspection approach. The entire inspection system presented in this thesis falls apart if SLAM cannot be reliably generated for a dataset. As discussed in this thesis, inspection environments often present a very challenging case for SLAM. Current literature does imply that lidar-visual-inertial SLAM has the best chances at creating a robust SLAM system, however, this is still a challenge to implement in practice. SLAM, in general, needs a lot of engineering to tailor the SLAM implementation to the specific hardware being used. Combining data from three sensor modalities (lidar, IMU, camera) only increases the engineering burden of creating a great SLAM system. This explains why no single SLAM tool exists in academia or industry that can be successfully applied to any sensor suit. In industry, such as the Virtual Reality space, each new device requires significant engineering effort to build up a reliable SLAM system. This thesis has proven the concept of a SLAM-centric inspection device by demonstrating the design shown here can work well on a few example datasets. However, having something that can be reliably deployed to any inspection environment and produce high-quality results will require significant engineering beyond what typical academic labs are capable of.

Inspection Map Renderings: This thesis shows that a 3D map can be colored with images collected during the inspection. This can be useful for inspectors to view the inspections in 3D off-site or months or years after the inspection. However, the results

from this process are far from visually realistic renderings. Pointclouds are relatively sparse and noisy, and images collected at different exposure levels create an unrealistic color pattern on the map. Clearly, a visually realistic digital twin of the asset would be very beneficial for inspectors, and the direct combination of pointclouds and images likely is not the best solution. However, recent advancements in neural rendering techniques such as NERFs [97] and Gaussian Splatting [64] show tremendous potential to solve this problem. These solutions are also capable of using the data provided by the inspection system presented in this thesis (images + pointclouds), and therefore, the exploration of this space is highly recommended for future work.

6.5 Implementation in Practice

This section briefly discusses how the proposed SLAM-centric inspection methodology can and should be implemented in practice given the studies conducted over the course of this research. There are two key considerations that have been discovered over the years of testing such an approach and meeting with practicing inspectors.

First, infrastructure inspection generally requires licensed engineers to conduct the inspections and take responsibility for the accuracy of the results. As such, using a tool such as the one proposed in this thesis should allow inspectors to have better confidence in their measurements and not obscure the decision-making process. Measurements should be easy to visualize and validate manually. Inspectors should also be able to adjust the measurements to better align with their own observations. Having a final 3D map that the Engineer could use to visualize all the defects and compare to images is critical. It is also critical to allow a more human-in-the-loop defect detection methodology, which is also demonstrated in this work.

Second, the less disruptive the inspection tool is to the inspector's usual tasks, the more likely the tool is to be implemented in practice. Inspectors are not necessarily tech-savvy, nor are they proficient at operating robots. In the studies conducted in this thesis, inspectors often gave the feedback that a handheld device is likely the best form-factor since it does not require learning to operate a robot. Furthermore, inspectors already carry cameras to capture images of defects. Having a handheld scanner replacing the camera in their hand provides a very low barrier to entry for this technology.

Given these two considerations, this thesis recommends that the inspection methodology should be first implemented as a low disruption supplementary tool. This tool should be easy to operate and should provide additional verifiable information for the inspector

to make decisions. This tool should not be used as a means to fully automate inspections without inspectors in the loop.

6.6 Implications for Infrastructure Management

This research addresses pressing societal needs by reducing traditional inspection methods' cost, time, and variability. The proposed SLAM-centric framework empowers infrastructure managers with a quantitative, repeatable, and scalable approach to asset assessment, enabling improved resource allocation and prioritization of maintenance tasks. The ability to track changes in defects over time fosters a proactive rather than reactive maintenance culture, ultimately enhancing the safety and longevity of critical infrastructure components like bridges and parking garages.

This research represents a significant step forward in automating infrastructure inspection using mobile robotics. By adopting a SLAM-centric approach, this work lays a foundation for more efficient, accurate, and repeatable inspection processes that are poised to disrupt current industry practices. The innovations and findings herein not only advance the state of the art but also hold promise for tangible societal benefits through safer, longer-lasting infrastructure. This work hopes to inspire further research and adoption of mobile robotics in civil engineering academics and industry. In the long term, society ultimately needs such advancements to emerge in order to continue to build infrastructure at such a high rate without weakening the economy with outrageous maintenance and repair costs.

References

[1] Ikhlas Abdel-Qader, Osama Abudayyeh, and Michael E Kelly. Analysis of edge-detection techniques for crack identification in bridges. *Journal of Computing in Civil Engineering*, 17(4):255–263, 2003.

[2] R. S. Adhikari, O. Moselhi, and A. Bagchi. Image-based retrieval of concrete crack properties for bridge inspection. *Automation in Construction*, 39:180–194, 2014.

[3] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org.

[4] Habib Ahmed, Hung Manh La, and Nenad Gucunski. Review of non-destructive civil infrastructure evaluation for bridges: State-of-the-art robotic platforms, sensors and algorithms. *Sensors*, 20(14):3954, 2020.

[5] Hatem Alismail, L. Douglas Baker, and Brett Browning. Automatic calibration of a range sensor and camera system. *Proceedings - 2nd Joint 3DIM/3DPVT Conference 2012*, pages 286–292, 2012.

[6] Zahra Ameli, Yugandhar Aremanda, Wilhelm A Friess, and Eric N Landis. Impact of uav hardware options on bridge inspection mission capabilities. *Drones*, 6(3):64, 2022.

[7] American Society of Civil Engineers. Infrastructure Report Card. Technical report, American Society of Civil Engineers, 2021.

[8] Nicolas Andreff, Radu Horaud, and Bernard Espiau. Robot Hand-Eye Calibration Using Structure-from-Motion. *The International Journal of Robotics Research*, 20(3):228–248, 2001.

[9] Stéphanie Aravecchia, Marianne Clausel, and Cédric Pradalier. Comparing metrics for evaluating 3d map quality in natural environments. *Robotics and Autonomous Systems*, 173:104617, 2024.

[10] Josep Aulinas, Yvan Petillot, Joaquim Salvi, and Xavier Lladó. The SLAM problem: A survey. *Frontiers in Artificial Intelligence and Applications*, 184(1):363–371, 2008.

[11] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics and Automation Magazine*, 13(3):108–117, 2006.

[12] Timothy D Barfoot. *State Estimation for Robotics*. Cambridge University Press, Cambridge, U.K., 2018.

[13] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[14] Rene Bergelt, Owes Khan, and Wolfram Hardt. Improving the intrinsic calibration of a velodyne lidar sensor. In *IEEE Sensors*, pages 1–3. IEEE, 2017.

[15] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. A survey of surface reconstruction from point clouds. In *Computer Graphics Forum*, volume 36, pages 301–329. Wiley Online Library, 2017.

[16] Paul Besl and Neil McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[17] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3, pages 2743–2748. IEEE, 2003.

[18] Michael Bloesch, Marco Hutter, Christian Gehring, Mark A Hoepflinger, and Roland Siegwart. Kinematic batch calibration for legged robots. In *2013 IEEE International Conference on Robotics and Automation*, pages 2542–2547. IEEE, 2013.

[19] Jean-Yves Bouguet. Camera calibration toolbox for matlab. *Computational Vision at the California Institute of Technology.*, 2015.

[20] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[21] Duane Brown. Decentering Distortion of Lenses - The Prism Effect Encountered in Metric Cameras can be Overcome Through Analytic Calibration. *Photometric Engineering*, 32(3):444–462, 1966.

[22] 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: Towards the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[23] Guy Campion, Georges Bastin, and Brigitte Dandrea-Novel. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 12(1):47–62, 1996.

[24] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 2021.

[25] Canadian Infrastructure Report Card. Informing the Future: Assessing the Health of our Communities' Infrastructure. Technical report, Canada Infrastructure, 2019.

[26] Juan Castorena, Ulugbek S. Kamilov, and Petros T. Boufounos. Autocalibration of lidar and optical cameras via edge alignment. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 2862–2866, 2016.

[27] N. Charron, S. Phillips, and S.L. Waslander. De-Noising of Lidar Point Clouds Corrupted by Snowfall. *Computer and Robot Vision, Toronto, ON, Canada, May 9-11*, 2018.

[28] Nicholas Charron, Evan McLaughlin, Stephen Phillips, Kevin Goorts, Sriram Narasimhan, and Steven L Waslander. Automated bridge inspection using mobile ground robotics. *Journal of Structural Engineering*, 145(11):04019137, 2019.

[29] Kenny Chen, Brett Lopez, Ali-akbar Agha-mohammadi, and Ankur Mehta. Direct lidar odometry: Fast localization with dense point clouds. *IEEE Robotics and Automation Letters*, 2022.

[30] Liang-Chieh Chen. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[31] G.G. Clemena and W. T. McKeel. Detection of Delamination in Bridge Decks with Infrared Thermography. In *Transportation Research Record No. 664, Bridge Engineering, Volume 1*, pages 180–182, 1978.

[32] CloudCompare. (version 2.11.1) [GPL software]. Technical report, GPL software, 2016.

[33] Saleh Abu Dabous and Sainab Feroz. Condition monitoring of bridges with non-contact testing technologies. *Automation in Construction*, 116:103224, 2020.

[34] Frank Dellaert. Factor Graphs and GTSAM: A Hands-on Introduction. Technical Report September, Georgia Institute of Technology, 2012.

[35] Ankit Dhall, Kunal Chelani, Vishnu Radhakrishnan, and K Madhava Krishna. Lidar-camera calibration using 3d-3d point correspondences. *arXiv preprint arXiv:1705.09785*, 2017.

[36] Joris Domhof, Kooij F.P. Julian, and Kooij M. Dariu. An extrinsic calibration tool for radar, camera and lidar. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May:8107–8113, 2019.

[37] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[38] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping (SLAM): part I The Essential Algorithms. *Robotics & Automation Magazine*, 2:99–110, 2006.

[39] Andrew Ellenberg, Ahsas Kontsos, Franklin Moon, and Ivan Bartoli. Bridge deck delamination identification from unmanned aerial vehicle infrared imagery. *Automation in Construction*, 72:155–165, 2016.

[40] Chu-Qiao Feng, Bao-Luo Li, Yu-Fei Liu, Fu Zhang, Yan Yue, and Jian-Sheng Fan. Crack assessment using multi-sensor fusion simultaneous localization and mapping (SLAM) and image super-resolution for bridge inspection. *Automation in Construction*, 155:105047, 2023.

[41] Chuncheng Feng, Hua Zhang, Shuang Wang, Yonglong Li, Haoran Wang, and Fei Yan. Structural damage detection using deep convolutional neural network and transfer learning. *KSCE Journal of Civil Engineering*, 23(10):4493–4502, 2019.

[42] Dongming Feng and Maria Q Feng. Computer vision for shm of civil infrastructure: From dynamic response measurement to damage detection–a review. *Engineering Structures*, 156:105–117, 2018.

[43] Tully Foote. tf: The transform library. In *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–6. IEEE, 2013.

[44] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Robotics: Science and Systems XI*, 2015.

[45] Paul Furgale, Joern Rehder, and Roland Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1280–1286. IEEE, 2013.

[46] Andreas Geiger, Frank Moosmann, Ömer Car, and Bernhard Schuster. Automatic camera and range sensor calibration using a single shot. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3936–3943, 2012.

[47] Spencer Gibb, Hung Manh La, Tuan Le, Luan Nguyen, Ryan Schmid, and Huy Pham. Nondestructive evaluation sensor fusion with autonomous robotic system for civil infrastructure inspection. *Journal of Field Robotics*, 35(6):988–1004, 2018.

[48] Matthew N Gillins, Daniel T Gillins, and Christopher Parrish. Cost-effective bridge safety inspections using unmanned aircraft systems (uas). In *Geotechnical and structural engineering congress 2016*, pages 1931–1940, 2016.

[49] Kevin Goorts, Stephen Phillips, Ali Ashasi-Sorkhabi, and Sriram Narasimhan. Structural control using a deployable autonomous control system. *International Journal of Intelligent Robotics and Applications*, 1(3):306–326, 2017.

[50] Eleonora Grilli, Fabio Menna, and Fabio Remondino. A review of point clouds segmentation and classification algorithms. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:339, 2017.

[51] Giorgio Grisetti, Rainer Kümmerle, Hauke Strasdat, and Kurt Konolige. g2o: A general framework for (hyper) graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*, pages 9–13, 2011.

[52] Nenad Gucunski, Basily Basily, Jinyoung Kim, Jingang Yi, Trung Duong, Kien Dinh, Seong-Hoon Kee, and Ali Maher. RABIT: implementation, performance validation and integration with other robotic platforms for improved management of bridge decks. *International Journal of Intelligent Robotics and Applications*, 1(3):1–16, 2017.

[53] Carlos Guindel, Jorge Beltrán, David Martín, and Fernando García. Automatic extrinsic calibration for lidar-stereo vehicle sensor setups. In *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*, pages 1–6. IEEE, 2017.

[54] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, 1997.

[55] Jan Heller, Michal Havlena, Akihiro Sugimoto, and Tomas Pajdla. Structure-from-motion based hand-eye calibration using L infinity minimization. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3497–3503, 2011.

[56] Hilti slam challenges. `https://hilti-challenge.com/`.

[57] Andrew Howard, Andrey Zhmoginov, Liang-Chieh Chen, Mark Sandler, and Menglong Zhu. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. In *Proc. CVPR*, pages 4510–4520, 2018.

[58] McKinsey Global Institute. Mckinsey infrastructure practice. Technical report, McKinsey & Partners, 2013.

[59] Mohammad R. Jahanshahi, Sami F. Masri, Curtis W. Padgett, and Gaurav S. Sukhatme. An innovative methodology for detection and quantification of cracks through incorporation of depth perception. *Machine Vision and Applications*, 24(2):227–241, 2013.

[60] Jinyong Jeong, Younghun Cho, and Ayoung Kim. The road is enough! extrinsic calibration of non-overlapping stereo camera and lidar using road information. *IEEE Robotics and Automation Letters*, 4(3):2831–2838, 2019.

[61] Sungwook Jung, Duckyu Choi, Seungwon Song, and Hyun Myung. Bridge inspection using unmanned aerial vehicle based on hg-slam: Hierarchical graph-based slam. *Remote Sensing*, 12(18):3022, 2020.

[62] Sungwook Jung, Seungwon Song, Sanghyeon Kim, Junho Park, Jongbeom Her, Kyungshik Roh, and Hyun Myung. Toward autonomous bridge inspection: A framework and experimental results. In *2019 16th International Conference on Ubiquitous Robots (UR)*, pages 208–211. IEEE, 2019.

[63] Juho Kannala and Sami S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1335–1340, 2006.

[64] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[65] Ali Khaloo and David Lattanzi. Hierarchical dense structure-from-motion reconstructions for infrastructure condition assessment. *Journal of Computing in Civil Engineering*, 31(1):04016047, 2017.

[66] Ali Khaloo, David Lattanzi, Keith Cunningham, Rodney Dell'Andrea, and Mark Riley. Unmanned aerial vehicle inspection of the placer river trail bridge through image-based 3d modelling. *Structure and Infrastructure Engineering*, 14(1):124–136, 2018.

[67] Eung Su Kim and Soon Yong Park. Extrinsic calibration of a camera-LIDAR multi sensor system using a planar chessboard. *International Conference on Ubiquitous and Future Networks, ICUFN*, 2019-July:89–91, 2019.

[68] Giseop Kim and Ayoung Kim. Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4802–4809. IEEE, 2018.

[69] In-Ho Kim, Sungsik Yoon, Jin Hwan Lee, Sungwook Jung, Soojin Cho, and Hyung-Jo Jung. A comparative study of bridge inspection and condition assessment between manpower and a uas. *Drones*, 6(11):355, 2022.

[70] Pileun Kim, Bullet Jingdao Chen, and Bullet K Yong Cho. Robotic sensing and object recognition from thermal-mapped point clouds. *International Journal of Intelligent Robotics and Applications*, 1(3):243–254, 2017.

[71] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[72] C. Koch, K. Georgieva, V. Kasireddy, B. Akinci, and P. Fieguth. A review on computer vision based defect detection and condition assessment of concrete and

asphalt civil infrastructure. *Advanced Engineering Informatics*, 29:196–210, 2015. DOI: 10.1016/j.aei.2015.01.008.

[73] Kenji Koide, Shuji Oishi, Masashi Yokozuka, and Atsuhiko Banno. General, single-shot, target-less, and automatic lidar-camera extrinsic calibration toolbox. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11301–11307. IEEE, 2023.

[74] Julius Kummerle, Tilman Kuhner, and Martin Lauer. Automatic Calibration of Multiple Cameras and Depth Sensors with a Spherical Target. *IEEE International Conference on Intelligent Robots and Systems*, pages 5584–5591, 2018.

[75] H M La, R S Lim, B B Basily, N Gucunski, J G Yi, A Maher, F A Romero, and H Parvardeh. Mechatronic Systems Design for an Autonomous Robotic System for High-Efficiency Bridge Deck Inspection and Evaluation. *IEEE-ASME Transactions on Mechatronics*, 18(6):1655–1664, 2013.

[76] Hung Manh La, Nenad Gucunski, Seong-Hoon Kee, Jingang Yi, Turgay Senlet, and Luan Nguyen. Autonomous robotic system for bridge deck data collection and analysis. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1950–1955. IEEE, 2014.

[77] Debra F. Laefer, Linh Truong-Hong, Hamish Carr, and Manmeet Singh. Crack detection limits in unit based masonry with terrestrial laser scanning. *NDT and E International*, 62:66–76, 2014.

[78] David Lattanzi and Gregory Miller. Review of robotic infrastructure inspection systems. *Journal of Infrastructure Systems*, 23(3):04017004, 2017.

[79] David W. Law, David Silcock, and Lucas Holden. Terrestrial laser scanner assessment of deteriorating concrete structures. *Structural Control and Health Monitoring*, 25(5):1–15, 2018.

[80] Bin Lei, Ning Wang, Pengcheng Xu, and Gangbing Song. New crack detection method for bridge inspection using uav incorporating image processing. *Journal of Aerospace Engineering*, 31(5):04018058, 2018.

[81] Jesse Levinson and S. Thrun. Automatic online calibration of cameras and lasers. In *Robotics: Science and Systems*, 2013.

[82] Ronny Salim Lim, Hung Manh La, and Weihua Sheng. A robotic crack inspection and mapping system for bridge deck maintenance. *IEEE Transactions on Automation Science and Engineering*, 11(2):367–378, 2014.

[83] Jiarong Lin and Fu Zhang. R3live: A robust, real-time, rgb-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10672–10678. IEEE, 2022.

[84] W. Liu, S. Chen, and E. Hauser. LiDAR-based bridge structure defect detection. *Experimental Techniques*, 35(6):27–34, 2011.

[85] Xiyuan Liu, Chongjian Yuan, and Fu Zhang. Targetless extrinsic calibration of multiple small fov lidars and cameras using adaptive voxelization. *IEEE Transactions on Instrumentation and Measurement*, 71:1–12, 2022.

[86] Yu-fei Liu, Soojin Cho, and Jian-sheng Fan. Concrete Crack Assessment Using Digital Image Processing and 3D Scene Reconstruction. *Journal of Computing in Civil Engineering*, 30(1):1–19, 2016.

[87] Hans-Andrea Loeliger. Intro to Factor Graph. *IEEE Signal Processing Magazine*, January 2004.

[88] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

[89] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, 1981.

[90] Pablo Martinez, Mohamed Al-Hussein, and Rafiq Ahmad. A scientometric analysis and critical review of computer vision applications for construction. *Automation in Construction*, 107:102947, 2019.

[91] E McLaughlin, N Charron, and S Narasimhan. Combining deep learning and robotics for automated concrete delamination assessment. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 36, pages 485–492. IAARC Publications, 2019.

[92] Evan McLaughlin, Nicholas Charron, and Sriram Narasimhan. Automated defect quantification in concrete bridges using robotics and deep learning. *Journal of Computing in Civil Engineering*, 34(5):04020029, 2020.

[93] Jake McLaughlin, Nick Charron, and Sriram Narasimham. Visual-lidar map alignment for infrastructure inspections. *arXiv preprint*, 2024.

[94] Daniel Meister, Shinji Ogaki, Carsten Benthin, Michael J Doyle, Michael Guthe, and Jiří Bittner. A survey on bounding volume hierarchies for ray tracing. In *Computer Graphics Forum*, volume 40, pages 683–712. Wiley Online Library, 2021.

[95] Dominik Merkle, Annette Schmitt, and Alexander Reiterer. Concept of an autonomous mobile robotic system for bridge inspection. In *Remote Sensing Technologies and Applications in Urban Environments V*, volume 11535, page 115350A. International Society for Optics and Photonics, 2020.

[96] Pierre Merriaux, Yohan Dupuis, Rémi Boutteau, Pascal Vasseur, and Xavier Savatier. A study of vicon system positioning performance. *Sensors (Switzerland)*, 17(7):1–18, 2017.

[97] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[98] Ministry of Transportation Ontario, St. Catharines, ON. *Ontario structural inspection manual (OSIM)*, 2008.

[99] Paul Moller. CALTRANS Bridge Inspection Aerial Robot Report No. CA08-0182. *Advanced Highway Maintenance and Construction Program (AHMCT).*, 2008. University of California at Davis Davis, CA.

[100] M. Moore, B. Phares, Graybeal B., D. Rolander, and G. Washer. Reliability of visual inspection for highway bridges. Struct. Engrg. Rep. No. FHWA-RD-01-020, Federal Highway Administration, McLean, VA, USA, 2001.

[101] T. Moore and D. Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

[102] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572. IEEE, 2007.

[103] Naveed Muhammad and Simon Lacroix. Calibration of a rotating multi-beam lidar. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5648–5653. IEEE, 2010.

[104] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[105] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.

[106] Ashley Napier, Peter Corke, and Paul Newman. Cross-calibration of push-broom 2D LIDARs and cameras in natural scenes. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3679–3684, 2013.

[107] Erik Nelson. B(erkeley) l(ocalization) a(nd) m(apping)! [Online]. Available: `https://github.com/erik-nelson/blam`.

[108] Tarek Omar, Moncef L. Nehdi, and Tarek Zayed. Infrared thermography model for automated detection of delamination in RC bridge decks. *Construction and Building Materials*, 168:313–327, 2018.

[109] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[110] Nobuyuki Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[111] Matteo Palieri, Benjamin Morrell, Abhishek Thakur, Kamak Ebadi, Jeremy Nash, Arghya Chatterjee, Christoforos Kanellakis, Luca Carlone, Cataldo Guaragnella, and Ali-akbar Agha-Mohammadi. Locus: A multi-sensor lidar-centric solution for high-precision odometry and 3d mapping in real-time. *IEEE Robotics and Automation Letters*, 6(2):421–428, 2020.

[112] Gaurav Pandey, James R. McBride, Silvio Savarese, and Ryan M. Eustice. Automatic targetless extrinsic calibration of a 3D lidar and camera by maximizing mutual information. *Proceedings of the National Conference on Artificial Intelligence*, 3:2053–2059, 2012.

[113] Steven G. Parker, Greg Humphreys, Morgan McGuire, Martin Stich, Heiko Friedrich, David Luebke, Keith Morley, James Bigler, Jared Hoberock, David McAllister,

Austin Robison, and Andreas Dietrich. GPU ray tracing. *Communications of the ACM*, 56(5):93, 2013.

[114] Harriet Anne Peel. *Robotic navigation and inspection of bridge bearings*. PhD thesis, University of Leeds, 2019.

[115] Nhan H Pham and Hung M La. Design and implementation of an autonomous robot for steel bridge inspection. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 556–562. IEEE, 2016.

[116] Prateek Prasanna, Kristin J. Dana, Nenad Gucunski, Basily B. Basily, Hung M. La, Ronny Salim Lim, and Hooman Parvardeh. Automated Crack Detection on Concrete Bridges. *IEEE Transactions on Automation Science and Engineering*, 13(2):591–599, 2016.

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

[118] J. Rebello, A. Fung, and S. L. Waslander. Ac/dcc : Accurate calibration of dynamic camera clusters for visual slam. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6035–6041, 2020.

[119] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[120] Jean Pierre Richa, Jean-Emmanuel Deschaud, François Goulette, and Nicolas Dalmasso. Adasplats: Adaptive splatting of point clouds for accurate 3d modeling and real-time high-fidelity lidar simulation. *Remote Sensing*, 14(24):6262, 2022.

[121] Sergio A. Rodriguez F., Vincent Frémont, and Philippe Bonnifait. Extrinsic calibration between a multi-layer lidar and a camera. *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 214–219, 2008.

[122] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. *Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM*, pages 145–152, 2001.

[123] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1 – 4, 2011.

[124] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[125] T. Scott, A. A. Morye, P. Piniés, L. M. Paz, I. Posner, and P. Newman. Exploiting known unknowns: Scene induced cross-calibration of lidar-stereo systems. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3647–3653, 2015.

[126] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

[127] Junwon Seo, Luis Duque, and Jim Wacker. Drone-enabled bridge inspection methodology and application. *Automation in Construction*, 94:112–126, 2018.

[128] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[129] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142. IEEE, 2020.

[130] Tixiao Shan, Brendan Englot, Carlo Ratti, and Daniela Rus. Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping. *arXiv preprint arXiv:2104.10831*, 2021.

[131] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, 1985.

[132] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[133] Aaron Sirken, Gedaliah Knizhnik, Jessica McWilliams, and Sarah Bergbreiter. Bridge risk investigation diagnostic grouped exploratory (bridge) bot. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6526–6532. IEEE, 2017.

[134] Kiran Somasundaram, Jing Dong, Huixuan Tang, Julian Straub, Mingfei Yan, Michael Goesele, Jakob Julian Engel, Renzo De Nardi, and Richard Newcombe. Project aria: A new tool for egocentric multi-modal ai research. *arXiv preprint arXiv:2308.13561*, 2023.

[135] Satoshi Suzuki and Keiichi A. Be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics and Image Processing*, 30(1):32–46, 1985.

[136] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[137] Zachary Taylor and Juan Nieto. Automatic calibration of lidar and camera images using normalized mutual information. *IEEE Conference on Robotics and Automation (ICRA)*, 2013.

[138] Zachary Taylor and Juan Nieto. Motion-Based Calibration of Multimodal Sensor Extrinsics and Timing Offset Estimation. *IEEE Transactions on Robotics*, 32(5):1215–1229, 2016.

[139] Alexander Thoms, Gabriel Earle, Nicholas Charron, and Sriram Narasimhan. Tightly coupled, graph-based dvl/imu fusion and decoupled mapping for slam-centric maritime infrastructure inspection. *IEEE Journal of Oceanic Engineering*, 2023.

[140] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y. Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 23(7-8):693–716, 2004.

[141] Matthew M. Torok, Mani Golparvar-Fard, and Kevin B. Kochersberger. Image-Based Automated 3D Crack Detection for Post-disaster Building Assessment. *Journal of Computing in Civil Engineering*, 28(5):A4014004, 2014.

[142] Yelda Turkan, Simon Laflamme, and Liangyu Tan. Terrestrial Laser Scanning-Based Bridge Structural Condition Assessment. *InTrans Project Reports*, 199, 2016.

[143] Vladyslav Usenko, Nikolaus Demmel, and Daniel Cremers. The double sphere camera model. In *2018 International Conference on 3D Vision (3DV)*, pages 552–560. IEEE, 2018.

[144] J. Valença, I. Puente, E. Júlio, H. González-Jorge, and P. Arias-Sánchez. Assessment of cracks on concrete bridges using image processing supported by laser scanning survey. *Construction and Building Materials*, 146:668–678, 2017.

[145] A. Velas, M., Spanel, M., Materna, Z., Herout. Calibration of RGB Camera With Velodyne LiDAR. In *Conference on Computer Graphics, Visualization and Computer Vision*, 2014.

[146] Velodyne LiDAR, Inc., 345 Digital Drive, Morgan Hill, CA 95037. *Velodyne LiDAR Pick^{TM} real-time 3D LiDAR sensor*, 2017.

[147] C Wang, C Harper, Y Lee, R Harris, and C Berryman. Autonomous Mobile Robot Localization and Mapping for Unknown Construction Environments. In *Construction Research Congress*, pages 491–500, 2018.

[148] David Wisth, Marco Camurri, Sandipan Das, and Maurice Fallon. Unified multi-modal landmark tracking for tightly coupled lidar-visual-inertial odometry. *IEEE Robotics and Automation Letters*, 6(2):1004–1011, 2021.

[149] Yujie Yan, Zhong Mao, Jiahao Wu, Taskin Padir, and Jerome F Hajjar. Towards automated detection and quantification of concrete cracks using integrated images and lidar data from unmanned aerial vehicles. *Structural Control and Health Monitoring*, 28(8):e2757, 2021.

[150] Liang Yang, Bing Li, Wei Li, Howard BrandH, Biao Jiang, and Jizhong Xiao. Concrete defects inspection and 3d mapping using cityflyer quadrotor robot. *IEEE CAA J. Autom. Sinica*, 7(4):991–1002, 2020.

[151] Haoyang Ye, Yuying Chen, and Ming Liu. Tightly coupled 3d lidar inertial odometry and mapping. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[152] Chul Min Yeum and Shirley J. Dyke. Vision-Based Automated Crack Detection for Bridge Inspection. *Computer-Aided Civil and Infrastructure Engineering*, 30(10):759–770, 2015.

[153] Chul Min Yeum, Shirley J Dyke, and Julio Ramirez. Visual data classification in post-event building reconnaissance. *Engineering Structures*, 155:16–24, 2018.

[154] Chongjian Yuan, Xiyuan Liu, Xiaoping Hong, and Fu Zhang. Pixel-level extrinsic self calibration of high resolution lidar and camera in targetless environments. *IEEE Robotics and Automation Letters*, 6(4):7517–7524, 2021.

[155] Chaobo Zhang, Chih-chen Chang, and Maziar Jamshidi. Simultaneous pixel-level concrete defect detection and grouping using a fully convolutional model. *Structural Health Monitoring*, 20(4):2199–2215, 2021.

[156] Chaobo Zhang, Chih-chen Chang, and Maziar Jamshidi. Simultaneous pixel-level concrete defect detection and grouping using a fully convolutional model. *Structural Health Monitoring*, 20(4):2199–2215, 2021.

[157] Ji Zhang and Sanjiv Singh. LOAM : Lidar Odometry and Mapping in Real-time. *Robotics: Science and Systems*, 2014.

[158] Ji Zhang and Sanjiv Singh. Laser–visual–inertial odometry and mapping with high robustness and low drift. *Journal of Field Robotics*, 35(8):1242–1264, 2018.

[159] Ji Zhang and SShewarmaanjiv Singh. Visual-lidar Odometry and Mapping: Low-drift, Robust, and Fast. *IEEE International Conference on Robotics and Automation*, pages 2174–2181, 2015.

[160] Qilong Zhang and Robert Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3:2301–2306, 2004.

[161] Zichao Zhang, Henri Rebecq, Christian Forster, and Davide Scaramuzza. Benefit of large field-of-view cameras for visual odometry. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:801–808, 2016.

[162] Shibo Zhao, Hengrui Zhang, Peng Wang, Lucas Nogueira, and Sebastian Scherer. Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments. *arXiv preprint arXiv:2104.14938*, 2021.

[163] Lipu Zhou, Zimo Li, and Michael Kaess. Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5562–5569. IEEE, 2018.

[164] Jennifer Zink and Barritt Lovelace. Unmanned aerial vehicle bridge inspection demonstration project. Technical report, Minnesota Department of Transportation, 2015.

APPENDICES

Appendix A

List of Publications

A.1 Publications in Referred Journals

- McLaughlin, E., **Charron, N.**, & Narasimhan, S. (2020) "Automated defect quantification in concrete bridges using robotics and deep learning". Journal of Computing in Civil Engineering, 34(5)
- **Charron, N.**, McLaughlin, E., Phillips, S., Goorts, K., Waslander, S., and Narasimhan, S. (2018) "Automated Bridge Inspection using Mobile Robotics" Journal of Structural Engineering
- Thoms, A., Earle, G. **Charron, N.**, and Narasimhan, S. (2023) "Tightly Coupled, Graph-Based DVL/IMU Fusion and Decoupled Mapping for SLAM-Centric Maritime Infrastructure Inspection" IEEE Journal of Oceanic Engineering

Contributions Summary:

In McLaughlin et al. (2020), my contributions to this work are as follows: design and integration lead of the mobile platforms used to collect data, leading the implementation of the localization and mapping, equal contribution in the data collection, and partial contributions in the map labeling and defect quantification design and implementation.

In Thoms et al. (2023), my contributions are as follows: lead design, calibration, and integration of sensors used for data collection, and ideation and full implementation of the de-coupled mapping approach.

A.2 Publications in Conference Proceedings

- McLaughlin, J., Thoms, A., **Charron, N.**, & Narasimhan, S. (2021) "Depth Completion from Sparse 3D Maps for Automated Defect Quantification". ASCE Lifelines Conference, Los Angeles, CA, USA

- Thoms, A., Earle, G., **Charron, N.**, Malama, S., & Narasimhan, S. (2021) "Combined Lidar and Sonar Mapping for Partially Submerged Infrastructure ". In 14th International Workshop on Structural Health Monitoring (IWSHM)

- Phillips, S., **Charron, N.**, McLaughlin, E., & Narasimhan, S. (2019) "Infrastructure Mapping and Inspection using Mobile Ground Robotics". In 12th International Workshop on Structural Health Monitoring (IWSHM).

- McLaughlin, E., **Charron, N.**, & Narasimhan, S. (2019) "Combining Deep Learning and Robotics for Automated Concrete Delamination Assessment". In 36th International Symposium on Automation and Robotics in Construction (ISARC) (pp. 485-492)

- **Charron, N.**, Phillips, S., & Waslander, S. L. (2018) De-noising of Lidar Point Clouds Corrupted by Snowfall. In 15th Conference on Computer and Robot Vision (CRV) (pp. 254-261). IEEE.

Contributions Summary:

In McLaughlin et al. (2021), my contributions were minor. I helped in the ideation of the depth completion as a way to improve defect quantification. I also contributed to the design and implementation of the defect and map labeling.

In Thoms, et al. (2021), my contributions are as follows: lead design, calibration, and integration of sensors used for data collection, and ideation and full implementation of the de-coupled mapping approach.

In Phillips et al. (2019), my contributions were minor in data collection, data processing, and paper writing.

In McLaughlin et al. (2019), my main contributions included: leading the system design and integration, significant contribution to the data collection, and minor support in data analysis and paper writing.

A.3 Publications Under Review

- **Charron, N.**, Waslander, S. L., & Narasimhan, S.(2025) "A Target-Based Extrinsic Calibration Framework for Non-Overlapping Camera-Lidar Systems Using a Motion Capture System". Computer and Robot Vision (CRV)
- **Charron, N.**, McLaughlin, J., & Narasimhan, S. (2025) "SLAM-VII - SLAM-centric Inspection of Infrastructure". Journal of Automation in Construction.
- **M**cLaughlin, J., **Charron, N.**, & Narasimhan, S. (2025) "Visual-Lidar Map Alignment for Infrastructure Inspections". IEEE Robotics and Automation Practice.

Contributions Summary:

In McLaughlin et al. (2025), my contributions are as follows: lead the design, integration, and calibration of the sensors used in data collection, lead contributor to the data collection, and minor contributions to the methodology ideation and data analysis approach.

Appendix B

Map Quality Analysis

Prior to discussing the way that inspection maps are generated, it is important to come up with a way to validate the map quality. This will enable the algorithms here to be compared in a way that better reflects the end goals of this work, that is, optimal inspection maps. Contrary to the majority of mapping research including the field of SLAM, this work focuses on map quality over location accuracy. All code for the proposed map quality analysis methodology is publicly available at `github.com/nickcharron/map_quality_analysis`.

B.1 Existing Quality Analysis Approaches

Traditional SLAM works compare different quality of results in one of two ways: (1) trajectory validation, or (2) comparing metrics minimized during the mapping optimization. Trajectory validation is either performed by comparing to ground truth such as RTK GPS or surveyed locations as is done in [56], or using SLAM techniques such as comparing offline loop closure errors. On the other hand, metric based analysis uses things such as total reprojection error for visual SLAM, or point-to-point distance error from scan registration in lidar SLAM.

Since this work focuses on optimizing for map quality, not trajectory accuracy like most SLAM works, trajectory validation is not an ideal approach for validation. Furthermore, these datasets were recorded on real sites and therefore gathering ground truth trajectories or surveyed anchor points was not realistic. Other metrics such as comparing reprojection errors or point-to-point registration errors also do not provide an appropriate metric for final inspection map quality, especially with the decoupled mapping methodology presented in this chapter.

There are also many works within and outside of the SLAM research area that attempt to come up with metrics for estimating map quality itself. However, all these methods rely on having some sort of ground truth map (e.g., collected with a TLS). For a full literature review on map quality analysis research, specifically related to robotics in natural environments, refer to [9]. Since no ground truth maps were available for the datasets described here, this work studies how to come up with metrics that measure the overall quality of a map on its own.

Having a map quality analysis methodology that does not require ground truth poses or maps is especially useful for inspection of new structures in practice since generating such ground truth is time consuming and defeats the purpose of building a quick mobile mapping solution. Inspectors can use these a metric for automatically determining the quality of the reconstruction or for tuning the mapping parameters. The following section describes the new map quality analysis tool and compares to some other metrics available in open source tools.

B.2 Proposed Distance To Planes Map Quality Metric

The proposed methodology is designed based on how humans generally inspect pointcloud maps for overall quality. Figure B.1 shows three examples of the same map with three different quality levels: (1) worst quality shown in red - online SLAM only, (2) medium quality shown in blue - refinement using few loop closure measurements, (3) best quality shown in green - best refinement using maximum loop closure measurements. The left image shows the full map from a top down view, the middle image shows a zoomed in top down view of a car next to a wall, and the right image shows a zoomed top down view of another wall that was more distant from the robot. The red map clearly shows a lot of drift in the map as can be seen with the double surfaces in some areas where the robot re-visited. The points in the blue map sit between the red and green points which is expected. Visually comparing these three maps, you can clearly see the difference in map quality by looking at how thick or noisy the surfaces are. The more accurate the map, the cleaner the surfaces which results in thinner surfaces. The proposed approach attempts to quantify the surface thicknesses and use this as the map quality metric.

This section proposes a metric which extracts the N largest planes in the dataset, and then computes the average point distance to the plane for the points that belong to this plane. The idea here being that calculating metrics on a subset of points with a high

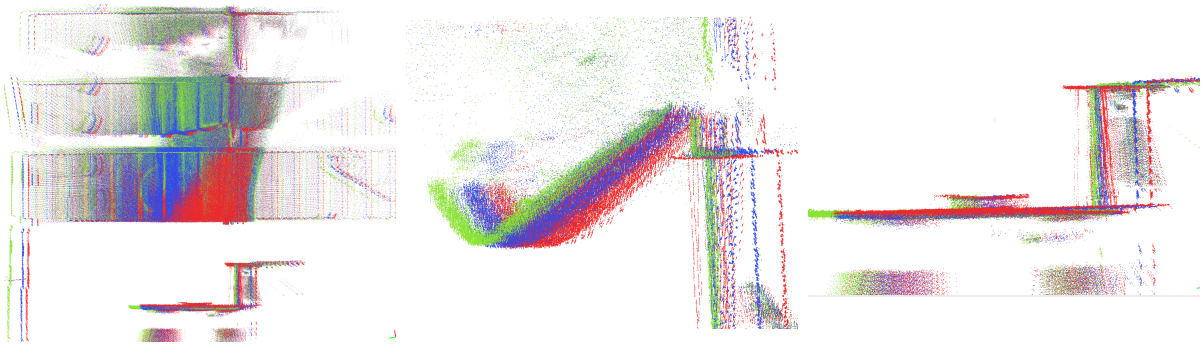


Figure B.1: Example map with 3 different map qualities on garage dataset: red - poorest with only online SLAM results, blue - medium quality with slight refinement, green - best quality with maximum refinement

confidence of their underlying structure would give better results than trying to estimate metrics on all points with unknown surface shapes. It is expected that inspections datasets will contain a given number of large planar surfaces which can be approximately estimated ahead of time given the dataset. Figure B.2 illustrates example results of running the plane segmentation procedure on the parking garage dataset.

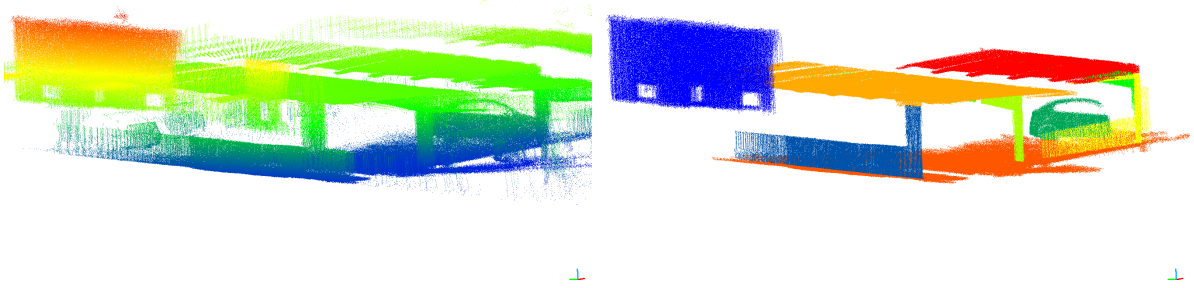


Figure B.2: Parking garage dataset plane extraction results. Left: raw pointcloud, right: final planes

The steps in computing point-to-plane metrics are as follows:

1. Extract K horizontal planes and J vertical planes: if the maps are gravity aligned, as is the case with many SLAM methods that have access to IMU data, then planes can

be extracted in both the horizontal and vertical direction separately. This allows the user to specify how many horizontal planes are expected which can often be known ahead of time. The planes are extracted using [123], with a minimum distance of 15 cm for a point be be considered part of a plane, and an angular threshold of 15 deg relative to horizontal or vertical.

2. Filter based on Euclidean clustering: The plane fitting in the previous steps attempts to fit a plane to the entire map to maximize the points along that plane. This causes a lot of outlier points which are often disconnected from the main plane. Instead, the biggest connected plane is more desired. Reminder that this prioritizes precision over recall since a higher confidence that the points belong to an underlying planar structure is preferred over capturing all points. To do this, Euclidean clustering is run using a clustering distance threshold of 6 cm. All clouds are then sorted based on the number of points in each cluster and the K and J largest clusters are taken in the horizontal and vertical directions, respectively. This gives a much cleaner set of planes with very few outlier points.

3. Compute mean distance to plane: each segment is then taken on its own and a new plane is fit to the segment to estimate the plane parameters. Next, each point belonging to that plane is projected onto the normal of the plane and the distance to the plane is recorded. The final metric is then determined by taking the average of all these distances over all planes.

Table B.1 shows the results of running the proposed map quality metric on both datasets presented in this work. These metrics are also compared to other pointcloud quality metrics freely available which do not require a ground truth map.

In Table B.1, the first metric labeled *point-to-plane* is the proposed metric described above. *Roughness* and *Density* are metrics available in the popular open-source CloudCompare [32]. *Roughness* is computed by taking the distance between each point and the best fitting plane around its nearest N neighbors. The idea of this metric is that better quality maps will have points more concentrated and therefore the distance from each point to the plane defined by its neighbors should be smaller for better maps. This is similar to the proposed metric, except it assumes all points belong to a plane and runs over all points. The *Density* metric is computed by iterating over all points and calculating the number of neighbors found within a fixed search radius and dividing by the area to approximate a surface density. The idea behind this metric is that higher quality maps should be more dense as points are more concentrated along the surfaces. Note that both of the CloudCompare metric have hyperparameters (search radius, and number of neighbors) that need

Table B.1: Comparing different pointcloud map quality metrics on bridge and parking garage datasets

Dataset	Refinement Level	point-to-plane (cm) (proposed)	Roughness (cm) (CloudCompare)	Density ($\frac{pt}{cm^2}$) (CloudCompare)
Garage	None	4.164	1.511	0.493
	light	3.904	1.521	0.512
	max	3.647	1.538	0.543
Bridge	None	6.175	1.419	0.301
	light	5.183	1.531	0.367
	max	4.231	1.529	0.396

to be set, whereas there is not a clear intuition of what these need to be set to for your specific maps which makes these metrics a bit hard to use properly. On the other hand, the proposed metric has more hyper parameters but these can be set pretty intuitively. For example, the plane segmentation threshold should be set based on how thick typical planes are in the maps which can be estimated by looking at a few maps. Furthermore, the planes that are segmented out can be visualized and the parameters can be tuned to make sure we do in fact segment out the largest planes for that dataset.