Map Point Selection for Hardware Constrained Visual Simultaneous Localisation and Mapping

by
Christiaan Johann Müller

Dissertation presented for the degree of Doctor of Philosophy (Electronic Engineering) in the Faculty of Engineering at Stellenbosch University

Supervisor: Dr Corné E. van Daalen

March 2024

Acknowledgements

I would like to express my heartfelt gratitude to the following individuals and institutions without whom this dissertation would not have been possible:

To my advisor Dr. Corné van Daalen: I would like to express my deepest gratitude for your invaluable support, guidance, and patience throughout the duration of my work on this dissertation. Your advice throughout the years of this project has shaped my academic growth and was instrumental in the completion of this endeavour.

To my parents, Johann and Corlia Müller, and my brother Thinus, thank you for your unwavering love, understanding and constant encouragement. Your belief in me and your sacrifices allowed me to realise this academic achievement.

To my friends and peers: I want to thank my friends and peers who provided a support network and were there to lend an ear or offer advice. In particular, I would like to thank Izaak van Zyl, Francois Lombard, Megan Naidoo, and Michelle Gelderblom for their support during the various stages of my postgraduate journey.

To the Wilhelm Frank Trust: I am thankful for the financial support that allowed me to dedicate myself to this project.

To the ESL Lab: I am thankful to you for providing the facilities necessary to pursue my research, a community of fellow researchers, and fond memories of various social events.

Lastly, I would like to thank all the unnamed individuals who, through support, encouragement, and collective wisdom, contributed in making this journey possible.

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2024

Abstract

Simultaneous localisation and mapping (SLAM) plays a vital role in autonomous robotics. Robotic platforms are often resource constrained, and this limitation necessitates resource-efficient SLAM implementations. While sparse visual SLAM algorithms offer reasonable accuracy for modest hardware requirements, these more scalable approaches still face limitations when applied to large-scale and long-term scenarios. Existing SLAM approaches require manual adjustment to maintain a compact map of the environment.

This dissertation presents an optimisation-based approach to select a good subset of map points for visual SLAM, which discards redundant map points in an automated fashion. Novel information-theoretic approaches are developed as solutions for this optimisation problem. Existing coverage-based approaches for related robotics problems are also adapted to selecting map points for visual SLAM.

The first of the novel information-theoretic approaches developed in this dissertation allows for accurate SLAM estimation by considering the full SLAM estimation problem, which allowed the reduction of the map points by more than 60%. This is in contrast to coverage-based approaches, which do not allow for accurate SLAM estimation. This first information-theoretic approach is too computationally expensive to use online and motivates the development of computationally inexpensive alternatives. To this end, this dissertation also proposes two additional information-theoretic map point selection techniques that approximate the SLAM estimation problem. These approximate approaches allow for similar trajectory accuracy to the original approach while reducing computation times from hours to often less than a second. These approximate approaches enable the practical online application of these techniques to reduce maps, while minimising the impact on SLAM trajectory accuracy. Lastly, this dissertation also presents a combined approach that improves on the approximate information-theoretic approach by including a coverage-based model. This combined approach outperforms all alternative map point selection approaches developed in this dissertation.

Map point selection approaches are first evaluated offline using a modified version of an existing visual SLAM algorithm (ORB-SLAM 2). Different map point selection approaches are evaluated on practical data from both the EuRoC and KITTI datasets. A follow-up experiment also demonstrates the application of map point selection approaches online.

Uittreksel

Gelyktydige lokalisering en kartering (GLEK) speel 'n belangrike rol in outonome robotika. Robotiese platforms is dikwels hulpbron beperk, en hierdie beperking noodsaak hulpbron doeltreffende GLEK-implementerings. Terwyl yl visuele GLEK-algoritmes redelike akkuraatheid bied vir beskeie hardeware vereistes, staar hierdie meer skaalbare benaderings steeds beperkings in die gesig wanneer dit toegepas word op grootskaalse en langtermynscenario's. Bestaande GLEK-benaderings vereis handaanpassing om 'n kompakte kaart van die omgewing te handhaaf.

Hierdie proefskrif bied 'n optimeringgebaseerde benadering om 'n goeie subversameling van kaartpunte vir visuele GLEK te kies, wat oortollige kaartpunte op 'n outomatiese wyse weggooi. Dit bied nuwe inligting teoretiese benaderings om oplossings vir hierdie optimerings probleem te ontwikkel. Bestaande dekkinggebaseerde benaderings vir verwante robotika probleme is ook aangepas om kaartpunte vir visuele GLEK te kies.

Die eerste van die nuwe inligting teoritiese benaderings wat in hierdie proefskrif ontwikkel is, maak voorsiening vir akkurate GLEK-skatting deur die volle GLEK-skattings probleem te oorweeg, wat die vermindering van die kaart punte met meer as 60% moontlik gemaak het. Dit is in teenstelling met dekkinggebaseerde benadering, wat nie voorsiening maak vir akkurate GLEK-beraming nie. Hierdie eerste inligting teoretiese benadering is rekenkundig te duur om aanlyn te gebruik en motiveer die ontwikkeling van rekenkundig goedkoop alternatiewe. Vir hierdie doel stel hierdie proefskrif ook twee bykomende inligting teoretiese kaartpuntseleksietegnieke voor wat die GLEK-beraming probleem benader. Hierdie benaderings maak voorsiening vir soortgelyke trajek akkuraatheid as die oorspronklike benadering, terwyl die berekening tye van ure tot dikwels minder as 'n sekonde verminder word. Gevolglik maak hierdie benaderings die praktiese aanlyn toepassing van hierdie tegnieke moontlik. Laastens bied hierdie proefskrif ook 'n gekombineerde benadering aan wat die vereenvoudige inligting teoretiese benadering verbeter deur 'n dekkinggebaseerde model in te sluit. Hierdie gekombineerde benadering presteer beter as alle alternatiewe kaartpuntseleksie benaderings wat in hierdie proefskrif ontwikkel is.

Kaartpuntseleksie benaderings word eers aflyn geëvalueer deur gebruik te maak van 'n gewysigde weergawe van 'n bestaande visuele GLEK-algoritme (ORB-SLAM 2). Kaartpuntseleksie benaderings word op praktiese data van beide die EuRoC- en KITTI-datastelle geëvalueer. Daarna word die tegnieke aanlyn gedemonsteer in 'n opvolgeksperiment.

Contents

Declaration ii

Abstract iii

Opsomming iv

Nomenclature viii

1. Introduction 1
- 1.1. Background . . . 1
- 1.2. Problem Formulation and Project Scope . . . 4
- 1.3. Overview of Existing Work . . . 4
- 1.4. Solution Overview . . . 6
- 1.5. Contributions . . . 7
- 1.6. Document Overview . . . 8

2. Literature Review 10
- 2.1. Visual SLAM Algorithms . . . 10
- 2.2. Coverage-based Map Point Selection for Localisation . . . 12
- 2.3. Information-theoretic Selection Approaches . . . 14

3. Map Point Selection for SLAM 17
- 3.1. Overview of visual SLAM . . . 17
  - 3.1.1. Overview of the Visual SLAM Front-end . . . 17
  - 3.1.2. The Visual SLAM Back-end Estimation Problem . . . 21
  - 3.1.3. Stereo Cameras for Visual SLAM Estimation . . . 24
- 3.2. Submodular Functions and Greedy Algorithms . . . 28
  - 3.2.1. Properties of submodular functions . . . 29
  - 3.2.2. Algorithms for submodular maximisation subject to a cardinality constraint . . . 29
- 3.3. Framework for Map Point Selection . . . 33

4. Map Point Selection Approaches 36
- 4.1. Utility based on SLAM information gain . . . 37
  - 4.1.1. Development of the SLAM-based Utility . . . 38

4.1.2. Alternative Implementation of SLAM utility using the Matrix Determinant Lemma . . . 42
4.1.3. SLAM-based Utility Implementation using Cholesky Updates . . . 44
4.2. Utilities based on Information-Theoretic Approximations . . . 46
4.2.1. Utility based on a Localisation Approximation . . . 47
4.2.2. Utility based on a Stereo Odometry Approximation . . . 49
4.3. Utility based on Coverage and Associated Approaches . . . 52
4.3.1. Existing Weighted Coverage Integer Program for Map Point Selection 53
4.3.2. Utility based on Greedy Weighted Coverage . . . 55
4.3.3. Map Point Selection as Maximising the Minimum Coverage . . . 58
4.4. Combined Utility Formulation for Improved Loop-closure Detection . . . 61

5. Evaluation Setup for Map Point Selection Approaches 65
5.1. Evaluation Metrics for Map Point Selection . . . 66
5.2. Modified ORB-SLAM 2 for evaluating map point selection . . . 69
5.2.1. ORB-SLAM 2 for Map Point Selection . . . 69
5.2.2. Modifications to the ORB-SLAM 2 algorithm . . . 71
5.3. Test Procedure for Evaluating Map Point Selection Approach . . . 72
5.4. Baseline Map Point Selection Approaches . . . 74
5.5. Evaluating Map Point Selection using the KITTI Dataset . . . 75
5.6. Evaluating Map Point Selection using the EuRoC Dataset . . . 80

6. Offline Experimental Results 83
6.1. Evaluation of SLAM Approaches and Random Selection . . . 84
6.1.1. Timing Results for SLAM approaches and Random on the KITTI dataset . . . 84
6.1.2. Trajectory Accuracy for SLAM approaches and Random on the KITTI dataset . . . 87
6.2. Evaluation of Coverage-based Approaches . . . 92
6.2.1. Timing Results for Coverage-based Approaches on the KITTI dataset 92
6.2.2. Trajectory Accuracy for Coverage-based Approaches on the KITTI dataset . . . 93
6.3. Evaluation of Approximate Information-theoretic and Combined Map Point Selection Approaches . . . 97
6.3.1. Timing Results for KITTI and EuRoC Datasets . . . 98
6.3.2. Trajectory Accuracy for the KITTI Dataset . . . 100
6.3.3. Trajectory Accuracy for the EuRoC Dataset . . . 107

7. Online Evaluation of Map Point Selection Approaches 110
7.1. Online Map Point Selection Implementation . . . 110

7.2. Online Results 113

8. Conclusion **118**
8.1. Research Summary 118
8.2. Summary of Contributions 119
8.3. Future Work 121

Bibliography **123**

A. Submodular maximisation subject to a knapsack constraint **129**

B. Supporting Investigations for Approximate Information-theoretic Results **131**
B.1. Visualisation of trajectory errors 131
B.2. Impact of the evaluation frame for the KITTI trajectory 132

C. Lower Threshold Experimental Results **135**

Nomenclature

Acronyms and abbreviations

APE	Absolute Pose Error (typically reported as RMSE)
BRIEF	Binary Robust Independent Elementary Features
EKF	Extended Kalman Filter
EuRoC	European Robotics Challenge
FAST	Features from accelerated segment test
IMU	Intertial Measurement Unit
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
MAP	Maximum a Posteriori
ORB	Oriented FAST and rotated BRIEF
PTAM	Parallel Tracking and Mapping
RANSAC	Random sample consensus
RMSE	Root Mean Squared Error
RPE	Relative Pose Error (typically reported as RMSE)
SLAM	Simultaneous localisation and mapping

Symbol Conventions

x	Scalar
$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
X	Set

List of Symbols

n	The number of map points in the map.
t	The number of robot poses in the trajectory.
i	Index associated with map points.
j	Index associated with robot poses.
$\mathbf{x}_j$	Robot pose (position and orientation) at time step j.
$\mathbf{x}_{1:t}$	Robot poses from timesteps 1 to t.
$\mathbf{m}_i$	Map point position of map point i.
$\mathbf{m}_{1:n}$	Map point positions of map points 1 to n.
$\mathbf{m}_S$	Map point positions of selected set of map points S.
$\mathbf{z}_{1:t}$	Map point observations from timesteps 1 to t.
$\mathbf{z}_{1:t}^S$	Map point observations of selected set of map points S at timesteps 1 to t.
$\mathbf{z}_{1:t}^i$	Map point observations of map point i at timesteps 1 to t.
$\mathbf{y}$	Place-holder for generic vector $\mathbf{y}$.
$\mathbf{\Lambda}_\mathbf{y}$	Information matrix (inverse covariance matrix) associated with the probability distribution over $\mathbf{y}$.
$\mathbf{\Lambda}_{\mathbf{x}_{1:n}}$	Information matrix (inverse covariance matrix) associated with robot poses 1 to t.
$\mathbf{\Omega}$	Measurement noise information matrix (inverse covariance matrix).
$h(\mathbf{x}_j, \mathbf{m}_i)$	The non-linear measurement model that computes the predicted measurement of map point with index i from robot pose with index j.
$\mathbf{J}_{ij}$	Jacobian of $h(\mathbf{x}_j, \mathbf{m}_i)$.
m_i	Map point with index i.
$\{m_i\}$	A set containing only map point with index i.
V	The set of map point indices.
S	Subset of map points indices, $S \subseteq V$.
$f(S)$	Real set function.
$f(m_i\|S)$	Marginal gain, or $f(\{m_i\} \cup S) - f(S)$.
X_i	Set of robot poses indices from which the map point with index i was measured.
V_j	The set of map point indices measured at the timestep with index j.

Chapter 1

Introduction

1.1. Background

Simultaneous localisation and mapping (SLAM) is the joint estimation of the position and orientation, or pose, of a robot and the state of its environment from sensor measurements. SLAM is essential for robustly addressing the autonomous navigation problem for mobile robots in unknown environments [2]. Cameras are a popular choice of sensor for SLAM due to their low cost and power consumption [3]. The visual SLAM algorithms that use cameras can be categorised by map representation as either sparse or dense. While sparse algorithms produce less detailed maps than their dense counterparts, they require only modest hardware. Sparse visual SLAM techniques have been demonstrated in applications of increasing scale over the past three decades (e.g. [4; 5; 6]). However, despite the continued improvement of SLAM algorithms, long-term and large-scale applications remain challenging [7].

Given the importance of the SLAM problem for the field of robotics, it is beneficial to consider this estimation problem in greater detail. It is helpful for the understanding of this problem to consider two simpler related problems: If we have a map of the environment, *localisation* is the problem of determining the position and orientation of the robot using map point measurements. The localisation problem uses measurements to answer the question: "Where is the robot in the environment?" For *mapping*, we instead assume that the pose of the robot is known and use measurements of the environment to build a map. Solving the mapping problem answers the question: "What does the robot's environment look like?" Robustly estimating both the robot trajectory and the map requires solving both these problems simultaneously, hence *simultaneous* localisation and mapping (SLAM).

The advantage of using a pair of stereo cameras to perform visual SLAM instead of a single camera is their ability to provide depth information and observe the scale of the environment. This results in improved accuracy when compared to using a single camera [8]. Stereo cameras are widely used for visual SLAM, and this dissertation considers the resulting stereo visual SLAM problem.

Practical sensor measurements present various challenges that complicate a unified

approach to solving the SLAM estimation problem. Due to the lack of a unified approach, SLAM algorithms are typically sensor-specific. Cameras produce very high-dimensional measurements in the form of images that are computationally expensive to process and difficult to model robustly. Consequently, practical sparse visual SLAM algorithms are divided into two parts: The first component is the front-end, which uses computer vision techniques to simplify camera frames into pixel position measurements of abstracted map points. The front-end is responsible for identifying if these measurements correspond to any of the existing map points. The second component is the back-end, which uses these abstracted lower-dimensional measurements to estimate the pose of the robot and the map point positions.

Figure 1.1: This figure shows a visualisation of the measured map points produced by the front-end component of a sparse visual SLAM algorithm. The measured pixel locations of abstracted map points are shown in green on top of the camera image. This visualisation is for the left camera of the stereo image pair for the ORB-SLAM 2 algorithm [8].

The front-end of a visual SLAM algorithm uses visual feature techniques to abstract the camera images to a set of map point measurements, as shown in Figure 1.1. These techniques encode the local texture information of salient points in an image using visual feature descriptors. These features are designed to be stable and be consistently detected across different viewpoints (examples of features include SURF [9] and ORB [10]). Each camera image is converted to a collection of such feature measurements at that time step. Different techniques are then used to match these visual feature measurements between the two stereo camera images and existing map points.

The back-end of a visual SLAM algorithm addresses the SLAM estimation problem, where the robot pose and the environment are estimated using the abstracted map point measurements from the front-end. Most modern visual SLAM algorithms jointly estimate the trajectory of robot poses and all map point positions using the full history of map point observations for the entire trajectory [7]. This is in contrast to classical approaches that use recursive estimators that only estimate the robot's current pose, but these approaches scale poorly in visual SLAM problems [11].

The data that a SLAM algorithm stores can be viewed in terms of the front-end and back-end components. The back-end SLAM estimation requires storing the map points

and robot state estimates for the full trajectory and all past observations. The front-end requires additional information, such as the feature descriptors associated with each map point. This required data grows with the length of the trajectory and the size of the environment.

ORB-SLAM 2 is a widely used state-of-the-art sparse stereo visual SLAM algorithm [8], and we now consider a typical run of the ORB-SLAM 2 algorithm on the KITTI [12] dataset as a representative example of visual SLAM algorithms. SLAM datasets include pre-recorded video sequences of different trajectories of the robot through an environment [12; 13]. The videos of these trajectories are then processed in real-time to simulate the camera image measurements of a robot moving through this environment. For a typical run of the ORB-SLAM 2 algorithm on a trajectory in the KITTI dataset, ORB-SLAM 2 extracts approximately 2,000 features for abstracted measurements per frame. The KITTI trajectory with label 00 contains roughly 4,540 camera frames, recorded at ten frames per second. This trajectory takes the robot under 8 minutes to complete, during which the ORB-SLAM 2 generates a map containing over 100,000 map points. ORB-SLAM 2 uses roughly 4 GB of memory for this trajectory.

The trajectories of datasets such as KITTI processed by SLAM implementations are already challenging for desktop-grade hardware from a processing and memory perspective. However, it would be beneficial to scale to even larger environments and have robots performing SLAM over time spans of multiple hours or days. For algorithms to be more applicable to these challenging scenarios, limiting this unbounded growth in map data and, consequently, the hardware requirements of the algorithms that use that map is necessary. This unbounded growth also limits the applicability of SLAM for robotic platforms with limited computational resources, memory, or communication bandwidth (in multi-robot scenarios). As a result, it is desirable to maintain an efficient representation of the environment to avoid exceeding these resource constraints.

SLAM algorithms already employ various heuristics to reduce the size of the map. Some of these heuristics include extracting a limited number of feature measurements per frame, using only a subset of camera frames called keyframes during SLAM estimation [5], or pruning recently added map points with a limited number of observations even though they are in field of view [6]. These existing heuristic approaches to limiting resource usage are typically only capable of limiting the *growth* of the map and are adjusted by experts in advance. These heuristics can be manually adjusted to have a more compact map by adjusting the number of extracted features, the logic for inserting keyframes, or the aggressiveness of discarding newly inserted map points. This manual approach has three drawbacks: It introduces approximations even when surplus resources are available. It does not simplify the map when resources are limited. The requirement of prior adjustment by experts can prove challenging if little information is known about the operational environment or an expert is not available to manually adjust the software implementation.

Instead of manually limiting the growth of the map data, it is desirable to simplify the map in an automated fashion based on the hardware constraints *during* operation. Such an automated approach would introduce fewer approximations when surplus resources are available and only restrict resource usage when necessary. This automated approach would also not require prior adjustment by experts to ensure that the resource constraints are not exceeded.

1.2. Problem Formulation and Project Scope

The goal of this dissertation is to develop an automated approach to limiting the map data for visual SLAM to retain the most useful information that the hardware constraints allow. This dissertation formulates the issue of limiting map data as an optimisation problem, where the problem is to select the most valuable subset of map points and associated data for the visual SLAM algorithm. This dissertation assumes that the relevant hardware constraints, whether memory cost or storage space, can be translated to a budget of map points. Therefore, the map point selection problem, as formulated in this dissertation, considers the problem of selecting a specified number of useful map points for a visual SLAM algorithm.

Visual SLAM algorithms consist of both front-end and back-end components, and the value or utility of a set of map points for a SLAM implementation depends on how the selected subset of map points impacts the functioning of these components. The problem of selecting the best subset of map points for visual SLAM will depend on the specific SLAM implementation. The goal of this dissertation is not to design map point selection algorithms for a specific SLAM implementation but rather to use a representative SLAM implementation as an example of how selection would be integrated with SLAM and develop as generally applicable map point selection algorithms as possible.

Using the number of map points is an abstraction of the underlying hardware constraints and does not consider the encoding scheme of map points and associated data for the specific visual SLAM software implementation. SLAM algorithms also maintain data related to each robot pose in the trajectory. The selection of data related to robot poses in the trajectory is not considered within the scope of this dissertation. Developing solutions for the map point selection problem is an intermediary step to developing optimisation approaches that also consider selecting robot poses or a wider range of resource constraints.

1.3. Overview of Existing Work

The problem of reducing the map by selecting a specified budget of map points for visual SLAM is unexplored in literature. However, we identified two important branches of

research that consider optimisation problems related to selecting map points for SLAM. These two branches are coverage-based approaches and information-theoretic approaches.

The first branch of approaches considers selecting map points for problems related to the visual SLAM problem, but does not select map points for the visual SLAM problem. These related problems include localisation, visual odometry, and visual-inertial navigation. Localisation is the problem of estimating the current robot pose given a known map, whereas both the robot pose and map are estimated for the SLAM problem. Visual odometry is the problem of estimating the incremental motion of the camera using map point measurements from a limited subset of camera frames, whereas visual SLAM estimates the full trajectory using all the map point measurements. Lastly, the visual-inertial navigation problem is similar to the visual odometry problem but also includes measurements from an inertial measurement unit (IMU), which are not typically available for visual SLAM.

The first branch of research uses *coverage-based* approaches, where map points are selected based on the number of map points visible in each frame. These coverage-based approaches are motivated by the requirement to have sufficient map points available for visual feature techniques at each frame. The selection of map points is formulated in terms of variations of the set multi-cover problem or maximum coverage problem. These coverage-based approaches have been applied to selecting map points for localisation (e.g. [14; 15; 16]) or visual-inertial navigation (e.g. [17; 18; 19]). A visual SLAM algorithm's front-end also uses feature techniques that suggest these coverage-based approaches are applicable to the SLAM problem. However, using coverage-based approaches for visual SLAM will not consider the back-end SLAM estimation problem.

The second branch of research that was identified is *information-theoretic* approaches used for various selection problems related to the map point selection problem for SLAM. Information-theoretic approaches are typically more expensive than the previously mentioned coverage-based approaches but explicitly optimise for estimation uncertainty.

A line of research investigates information-theoretic approaches for selecting a limited number of map point *measurements* for only the last timestep during SLAM (e.g. [20; 21]). Selecting map point measurements for the most recent pose in SLAM estimation only influences what new map point observations are added. In contrast, this dissertation aims to select *map points* for visual SLAM where selecting a limited number of map points from the total map points for the entire trajectory removes all present and past map point measurements of the map points that were not selected.

Carlone and Karaman [22] investigated selecting map points for visual-inertial navigation, which is equivalent to visual SLAM in a limited window, but with an inertial measurement unit added. Information-theoretic approaches have yet to be applied to the problem of selecting map points for the full visual SLAM problem. However, information-theoretic approaches are expensive and require further modifications to be feasibly applied

to the increased scale of selecting map points for SLAM.

This section reviewed existing information-theoretic approaches and coverage-based approaches for related problems. These approaches have yet to be applied to the visual SLAM problem. Information-theoretic approaches will be challenging to apply to the SLAM problem due to the computational cost, and while they consider the SLAM estimation, they do not consider the potential impact of map point selection on the front-end. In contrast, coverage-based approaches are motivated by the requirements of the front-end to have a sufficient number of feature matches. However, when applied to the SLAM problem, these coverage-based approaches will not consider the back-end SLAM estimation problem. Both the front- and back-end components are necessary considerations for developing novel map point selection approaches for visual SLAM.

1.4. Solution Overview

Before developing solutions for the map point selection problem, it is necessary to identify suitable optimisation algorithms to select a good subset of points given a utility function that quantifies the value of any subset of points. This dissertation identifies that coverage-based and information-theoretic utility functions used for related problems can be approximated with the same greedy optimisation algorithms with strong guarantees. Consequently, this dissertation uses these computationally inexpensive optimisation algorithms, which allows this research to focus on developing suitable utility functions for the map point selection problem.

Based on existing information-theoretic ideas for related problems, this dissertation develops a novel utility function based on selected map points to minimise the uncertainty of the SLAM trajectory estimation. This approach allows selecting subsets of map points for accurate SLAM trajectory estimation, but is too computationally expensive to use online. In order to address this issue, this dissertation also develops two novel information-theoretic utility functions that approximate the previously mentioned SLAM utility. These approaches are computationally inexpensive while often yielding comparable trajectory accuracy to the offline SLAM approach when selecting map points for SLAM.

Experimental results show that the ability of these information-theoretic approaches to reduce maps is often limited by the performance of the front-end component of the visual SLAM algorithm, specifically, the front-end task of loop-closure detection. Loop closures play a critical role in SLAM and occur when a robot revisits an existing location in the map. Loop closures allow for large corrections in the estimated trajectory, and loop-closure detection is the task of the visual SLAM front-end of identifying when loop closures have occurred.

Coverage-based alternatives are also adapted to the map point selection problem for visual SLAM. These approaches are motivated by the requirement of sufficient map points

for feature-matching front-ends. Results show that these coverage-based approaches often compare favourably to information-theoretic approaches based on the performance of the front-end components responsible for loop-closure detection. However, as shown in this dissertation, the maps from these coverage-based approaches do not generally allow for accurate SLAM estimation.

This dissertation also develops a novel approach that combines one of the approximate information-theoretic approaches with a coverage-based utility term to improve map point selection for loop-closure detection. This combined approach assumes knowledge about loop-closure locations. However, this approach outperforms all alternative map point selection approaches developed in this dissertation, both in terms of trajectory accuracy and loop-closure detection.

Map point selection approaches are first compared offline using a modified implementation of the existing visual SLAM implementation ORB-SLAM 2. This offline evaluation was necessary to compare map point selection approaches unsuitable for online applications. This research also finds that modifications to ORB-SLAM 2 improve performance using map point selection approaches. This offline evaluation is performed on both the KITTI and EuRoC datasets. After comparing approaches offline, a separate experiment is used to demonstrate the proposed approximate information-theoretic approaches online.

1.5. Contributions

This dissertation's main contribution is the development of novel information-theoretic map point selection approaches. The most notable of these are the approximate information-theoretic approaches for SLAM. These approximate techniques can reduce maps by more than 60% for the tested datasets without significantly impacting trajectory accuracy. This contrasts with the adapted coverage-based approaches, which require significantly more map points to achieve comparable trajectory accuracies. These approximate approaches now enable the practical application of map point selection algorithms on kilometre-scale environments, often with minimal impact on the accuracy of the visual SLAM algorithm.

A second contribution of this dissertation is the development of the novel combined approach. This combined approach outperforms all alternative map point selection approaches and allows for excellent trajectory accuracy and loop-closure detection. However, this approach requires additional loop-closure information to formulate the utility function. For applications where this information is not available, this combined approach serves as a point of comparison for techniques that do not use this information.

The third contribution of this dissertation is the development of a modified version of ORB-SLAM 2 to evaluate map point selection approaches. Future researchers can use this implementation to compare newly developed map point selection approaches to those proposed in this work. These modifications also highlight future directions towards an

optimised integration of map point selection with visual SLAM algorithms.

The fourth contribution of this dissertation is adapting coverage-based approaches previously used for other problems to the SLAM problem. The application of these techniques for the SLAM problem is unexplored in previous research. This dissertation demonstrates that these coverage-based approaches offer good recall performance for loop-closure detection for visual SLAM, but do not generally allow for accurate SLAM estimation.

The highlights of the contributions of this dissertation have been published in the Elsevier journal *Robotics and Autonomous Systems* [1].

1.6. Document Overview

This dissertation is structured as follows: Literature for problems related to the map point selection problem for visual SLAM is discussed in Chapter 2. While the map point selection problem is novel, approaches to existing problems inform the development of solutions for the map point selection problem in this work.

Chapter 3 provides an overview of the components of a visual SLAM algorithm, presents existing general-purpose greedy algorithms to maximise utility functions in this dissertation, and presents the framework for how map point selection approaches are used alongside a visual SLAM implementation. The overview of the components of visual SLAM algorithms in this chapter provides background information and informs the development of different utility functions for map point selection.

Chapter 4 develops different utility functions for map point selection to use with the greedy optimisation algorithms. This chapter details the development of an information-theoretic utility function for SLAM and approximations of the SLAM utility function based on localisation and visual odometry problems. This chapter also adapts existing coverage-based approaches to the SLAM problem. Lastly, this chapter presents a combined information-theoretic and coverage-based utility function using loop-closure information.

Chapter 5 modifies the existing ORB-SLAM 2 implementation to facilitate the offline evaluation of the different map point selection approaches. These modifications allow a fair comparison between approaches and address some limitations when using ORB-SLAM 2 to evaluate different map point selection approaches. This chapter also presents a test procedure to evaluate map point selection approaches and presents details for this test procedure for both the KITTI and EuRoC datasets.

Chapter 6 details the offline experimental evaluation of the different map point selection approaches using the modified SLAM implementation from Chapter 5.

Chapter 7 evaluates the approximate information-theoretic map point selection approaches suitable for online evaluation. This online evaluation demonstrates how to use map point selection online and validates the offline results for these approaches from

Chapter 6.

Lastly, Chapter 8 presents concluding remarks and avenues for future research.

Chapter 2

Literature Review

This chapter reviews existing work related to the map point selection problem for sparse visual SLAM. Section 2.1 provides an overview of sparse visual SLAM algorithms. While selecting map points for visual SLAM remains unexplored, this chapter provides an overview of existing research into related optimisation problems. These optimisation approaches can be categorised into two sections based on the techniques used to formulate the associated optimisation problem. Section 2.2 provides an overview of coverage-based approaches developed for localisation, while Section 2.3 provides an overview of information-theoretic selection approaches used for problems related to the map point selection problem for SLAM.

2.1. Visual SLAM Algorithms

In early approaches to SLAM, the SLAM problem was often solved using recursive estimators such as the extended Kalman filter (EKF), the unscented Kalman filter (UKF) or particle filters. Thrun [2] provides a comprehensive overview of classical approaches to SLAM. Davison *et al.* [4] proposed the first monocular SLAM algorithm using an extended Kalman filter. This work considers the map point selection problem for visual SLAM. Due to the poor asymptotic computational complexity of recursive estimators, these approaches have mainly been superseded by graph-based SLAM approaches, especially for visual SLAM problems.

Graph-based approaches use maximum likelihood estimation using the information matrix, that is, the inverse of the covariance matrix that corresponds with the linear approximation of the SLAM problem [2; 23]. An important insight is that this matrix is typically sparse for the full SLAM problem. Each map point observation adds non-zero entries between the estimated robot pose and map points in the information matrix. Map points are typically only visible from a limited number of robot poses. The non-zero entries of this matrix can equivalently be interpreted as a bipartite graph where the variables to be estimated are represented as graph nodes. Each map point observation at a timestep introduces an edge between the estimates of the map point and the robot pose variables. For typical sensors with a limited perceptive range, the map points observed at each

timestep are a small subset of all the map points in the map. The resulting graph is, therefore, commonly sparse. This graph provides essential insight into efficient SLAM estimation algorithms that exploit the problem's sparsity and the design of approximations for online SLAM applications.

Before graph-based approaches were applied to visual SLAM problems, these approaches were applied offline in the computer vision domain to perform estimation for the closely related problem of structure from motion. Structure from motion deals with the problem of reconstructing 3D scenes from an unordered set of 2D images [24]. As a final step for this problem, it is common to perform bundle adjustment, which is the joint estimation problem of the set of camera poses and map point observations, typically solved using non-linear least squares methods [25]. This estimation problem is largely equivalent to the visual SLAM estimation problem, where the trajectory of robot poses and map points are estimated. Subsequently, the term bundle adjustment in SLAM literature is often used to refer to the visual SLAM estimation problem of the full trajectory and map (e.g. [5; 6]). However, visual SLAM is an online problem that assumes sequential motion through the environment and known camera calibration. These assumptions do not necessarily apply to the structure from motion problems and bundle adjustment in general.

Klein and Murray [5] identified that not all camera images are useful for estimating the map and introduced keyframe-based SLAM techniques with the parallel tracking and mapping (PTAM) algorithm. This approach approximates the localisation of the camera in real-time by visual tracking, ignoring uncertainty in the map point positions. The SLAM estimation is done in parallel at a lower frequency, using bundle adjustment with a spatially distributed subset of the camera frames called keyframes. Performing bundle adjustment with only a limited number of non-redundant camera frames and not at frame rate was essential to allow the use of bundle adjustment in an online application. Strasdat *et al.* [11] showed that bundle adjustment with a subset of camera frames results in more accurate trajectory estimates for visual SLAM than filtering at the same computational budget.

Strasdat *et al.* [26] proposed approximating the bundle adjustment problem of the full trajectory with two simpler problems: Using the currently observed map points and past poses from which those map points were observed, it is possible to define a local region of map points and camera frames. Bundle adjustment is then only applied to this local region. Since this does not allow for large-scale estimations, a different heuristic approach is used for these corrections. Pose graph optimisation approximates the marginal distribution over camera poses with approximate heuristic factors based on the co-visibility or visual overlap between different camera poses. The pose-graph heuristically approximates the bundle adjustment problem and allows propagating large-scale corrections in the map in scenarios where bundle adjustment is considered prohibitively expensive.

Mur-Artal *et al.* [6] then developed the ORB-SLAM algorithm, which built on these

initial approaches. ORB-SLAM further separates the problem into different specialised sub-components running in parallel. Loop closures are detected using the visual large-scale localisation technique proposed by Galvez-López and Tardos [27]. SLAM estimation is based on the keyframe approach by Klein and Murray [5], which uses a separate tracking thread to localise the robot. A different sub-component is responsible for local mapping, which performs bundle adjustment in a small local window. Lastly, large-scale loop closures were optimised in parallel using an initial pose-graph heuristic followed by an additional step of bundle adjustment. Mur-Artal and Tardós [8] extended the baseline ORB-SLAM algorithm in ORB-SLAM 2 to support stereo cameras. Campos *et al.* [28] developed ORB-SLAM 3, which added support for inertial measurement units, revised the loop-closure algorithm and added improved support for multi-session mapping.

As an alternative to the aforementioned feature-based (or indirect) methods, direct methods have also been proposed to estimate the map by optimising directly on image intensities. Engel *et al.* [29] proposed LSD-SLAM, a direct method for doing visual SLAM. However, this approach still requires feature-based techniques for loop-closure detection and was shown to be less accurate than feature-based methods by Mur-Artal and Tardós [8]. While feature-based methods are less robust than direct methods in scenarios where camera images exhibit a lack of texture or motion blur in images, Campos *et al.* [28] shows ORB-SLAM 2 and 3 remain state of the art in terms of trajectory accuracy. This dissertation chooses to use ORB-SLAM 2 in this work as a representative SLAM implementation to evaluate the effects of map point selection algorithms. This choice is discussed in greater detail in Chapter 5.

2.2. Coverage-based Map Point Selection for Localisation

This section considers coverage-based map point selection approaches used for feature-based maps. Various works have investigated selecting map points for the large-scale maps produced by sparse feature-based structure-from-motion algorithms. These structure-from-motion algorithms can be seen as an offline version of visual SLAM, without ordered camera images or real-time processing constraints. The resulting reduced maps were then used for localisation instead of SLAM.

Li *et al.* [30] identified that many map points are redundant for localisation and proposed making a sub-selection of the map to improve localisation efficiency. They propose to model this problem as a set multi-cover problem, which involves selecting the smallest set of map points where each camera view retains at least a specified number of map points. The set multi-cover problem is a generalisation of the set cover problem. The set cover problem is known to be NP-hard; therefore, efficient solutions have yet to be discovered for this class of problems. Fortunately, set cover problems can be approximated using greedy algorithms in polynomial time.

Cheng *et al.* [16]; Cao and Snavely [31] and Camposeco *et al.* [32] improved baseline set multi-cover approaches for large-scale localisation. These improved approaches emphasise more unique descriptors, higher observation counts, or well-distributed points in the image plane. Emphasis on these properties improved localisation performance over the baseline set multi-cover approach. While often without strong optimisation guarantees, these heuristics result in scalable approaches for map point selection for large-scale localisation.

The map point selection problem for localisation has also been modelled as an integer programming problem. Integer programming considers optimisation problems over integer variables with linear or quadratic objective functions and constraints. These methods recover optimal solutions instead of greedy algorithms, but integer programming techniques typically have a non-polynomial computational complexity. Formulating the map point selection problem as an integer programming problem has been proposed by Van Opdenbosch *et al.* [15]; Park *et al.* [33] and Dymczyk *et al.* [34]. Park *et al.* [33] proposed both linear and quadratic integer programs for reducing maps for localisation. Dymczyk *et al.* [34] adapted the formulation by Park *et al.* [33] to use the number of map points visible in each frame as a soft constraint while limiting the total number of map points. Soft constraints is a technique where an objective function is penalised for not satisfying a constraint instead of typical constraints, which are required for a solution to be valid. Therefore, this approach favours solutions where each camera frame has a specified number of selected map points, but this is not guaranteed. This approach proves remarkably scalable in practice and in follow-up work by Lynen *et al.* [19], who demonstrates this approach on city-scale problems by dividing the city into blocks of 150 meters per side and selecting map points for each block independently. Additionally, Van Opdenbosch *et al.* [15] proposed a feature coding scheme and revised the formulation of Dymczyk *et al.* [34] to account for different storage costs associated when used along a proposed scheme for compressing feature descriptors.

A few approaches also included visual-inertial tracking to allow high-frequency updates when localising in reduced maps after map point selection. Greedy heuristics have been used by Lynen *et al.* [17] for this problem, while Schneider *et al.* [35] and Lynen *et al.* [19] investigate integer programming techniques. Schneider *et al.* [35] developed Maplab and the associated robust visual-inertial odometry with localisation integration (ROVIOLI) algorithm. This algorithm augments the baseline direct vision odometry technique with offline tools for bundle adjustment and uses the resulting map to assist localisation. The localisation and map-building approaches are feature-based and use the integer program by Dymczyk *et al.* [34] to reduce the offline map.

This overview of existing work demonstrates that coverage-based models can simplify maps efficiently for large-scale localisation but have not yet been applied to SLAM. Visual SLAM implementations employ similar localisation algorithms to perform large-scale loop closures (e.g. [6; 29]). Due to the success of these coverage-based algorithms for

localisation, we expect this approach would also do well for the localisation subsystem of a SLAM implementation. However, these coverage-based approaches will not necessarily perform as well for the SLAM problem.

2.3. Information-theoretic Selection Approaches

The previous section provided an overview of one branch of research related to the map point selection problem based on coverage-based approaches. Another relevant branch of research related to map point selection is information-theoretic approaches for related problems. This section reviews related optimisation problems where optimisation is performed with respect to a statistical measure (typically entropy or information gain). The approaches in this section formulate information-theoretic problems related to the map point selection problem for visual SLAM. Existing approaches do not select map points for graph-based visual SLAM.

Early approaches to SLAM used Kalman filtering to estimate both the map and the current robot position. A Kalman filter requires the maintenance of a dense covariance matrix of the robot pose and landmarks and it is therefore essential to limit the number of landmarks to ensure real-time operation. Early approaches by Dissanayake *et al.* [36] and Zhang *et al.* [37] prioritise creating new landmarks based on information gain while discarding landmarks no longer being observed.

Choudhary *et al.* [38] proposes a heuristic approach to remove map points for 2D graph SLAM. This formulation iteratively removes the map point that minimises the difference between the landmark's information gain and its memory cost, or $f_{\text{IG}}(l) - \lambda f_{\text{mem}}(l)$, until this metric is positive. The heuristic weight λ controls the landmark removal rate. In contrast, this work considers the budgeted map point selection problem for 3D visual SLAM, where we specify a set number of map points, which is incompatible with this landmark removal heuristic.

Kopitkov and Indelman [20] showed the similarity between different decision-making problems under uncertainty and proposed a general algorithm for optimisation algorithms defined over the posterior distribution where the selection or actions impact the factors in the posterior distribution. Within this framework, future observations are assumed to be known, and this assumption results in an equivalence between problems such as selecting map point measurements that maximise information gain or selecting actions for a robot to maximise information gain. Among other problems, they applied this general algorithm to the problem of selecting map point measurements at the current timestep for SLAM.

Zhao and Vela [21] selected map point measurements to reduce the computation time of ORB-SLAM to localise the camera during tracking. Similar to PTAM [5], ORB-SLAM [6] (the monocular version of ORB-SLAM 2) localises the camera at framerate, while the other sub-components of the SLAM implementation is done in parallel. This approach

uses the stochastic greedy algorithm by Mirzasoleiman *et al.* [39].

Carlone and Karaman [22] proposed an information-theoretic technique for selecting map points during visual-inertial odometry to limit computation time. This selects map points based on the predicted feature measurements. Visual-inertial odometry is equivalent to the local SLAM estimation problem that only considers the most recent set of poses and the associated map points. Visual-inertial odometry involves the use of an inertial measurement unit (IMU) in addition to a camera. This approach made two important assumptions appropriate for this setting: the orientation uncertainty is negligible due to the IMU, and the future states are known due to access to the robot's commands. This approach used greedy algorithms to optimise the proposed utility function with guarantees.

Selecting map points for visual SLAM given a specified budget still needs to be explored. The work by Carlone and Karaman [22], where map points were selected for visual-inertial odometry is the closest to the problem investigated by this dissertation, but selecting map points for SLAM has four crucial differences from selecting map points for visual-inertial odometry. The first is that it is feasible for visual odometry approaches to marginalise out past poses which allows for maintaining past information. For SLAM problems, the dense information matrix that results from marginalisation renders this approach unsuitable when selecting map points for SLAM. In the map point selection problem for visual SLAM, map point selection will also discard past map point measurements. The second is that SLAM presents a multiple-order of magnitude increase in the number of map points and poses that are jointly estimated. Carlone and Karaman [22] selects ten map points from the roughly hundreds and considers a few nearby poses for the visual-inertial odometry. SLAM maps can easily contain hundreds of thousands of map points and thousands of poses. Different approximations are necessary to select map points in problems of this scale. The third difference between selecting map points for SLAM instead of visual odometry is that SLAM implementations involve multiple interacting sub-modules to address the SLAM problem. These sub-modules address different components of the online SLAM problem, such as updating the local SLAM estimation, detecting loop closures and large-scale estimation after loop closures. Selecting map points for a visual SLAM implementation will influence these respective sub-modules. Visual inertial odometry does not include loop-closure detection or large-scale SLAM estimation. Lastly, visual-inertial odometry includes an inertial measurement unit which Carlone and Karaman [22] used to predict future poses, which will not be present for vision-only SLAM problems.

This section provided an overview of information-theoretic approaches for problems related to the map point selection for visual SLAM. An information-theoretic approach to the map point selection for the visual SLAM problem still needs to be explored. The techniques for developing computationally efficient solutions for these related problems can be used for the problem of selecting map points for SLAM. However, given the increased scale and complexity of selecting map points for SLAM, we expect it will be necessary to

develop novel techniques to allow an information-theoretical approach to be applicable when the computational cost of the approach is a consideration.

Chapter 3

Map Point Selection for SLAM

This chapter provides the necessary background required for the development of map point selection approaches for visual SLAM and describes the approach of this dissertation for developing solutions to this optimisation problem. Visual SLAM algorithms can be described in terms of two components: the front-end, which pre-processes camera images into a simplified form and the back-end, which uses these simplified measurements to perform the resulting estimation problem. Selecting map points will impact both of these components of the visual SLAM algorithm.

Section 3.1 provides an overview of the front-end and back-end components of a visual SLAM implementation. In literature, both coverage-based and information-theoretic utility functions are used to model the utility of map points for related problems. The approach of this dissertation is first to assume we are given a general utility function and consider relevant optimisation algorithms. Section 3.2 presents algorithms compatible with these functions. Lastly, Section 3.3 describes the framework for using map point selection alongside a visual SLAM algorithm. Chapter 4 develops different utility functions to use within this framework and with the algorithms in Section 3.2.

3.1. Overview of visual SLAM

This section first provides an overview of these front-end techniques in Subsection 3.1.1. These techniques generate the simplified measurements used by the back-end. Subsection 3.1.2 then provides an overview of the back-end and relevant information-theoretic quantities for measuring the impact of map point selection on the back-end. Information-theoretic approaches for evaluating the impact of map point selection on the back-end estimation problem require the measurement model to formulate relevant quantities. Consequently, Subsection 3.1.3 presents an overview of the stereo camera measurement model used by the back-end estimation.

3.1.1. Overview of the Visual SLAM Front-end

This subsection provides an overview of the process by which the front-end of a visual SLAM implementation abstracts stereo images into map point measurements. These

abstracted map point measurements not only include the measured pixel locations in the camera frames, but also the identity of the map points. The back-end estimation uses these abstracted map point measurements for the SLAM estimation problem. Selecting map points will impact the map points available during feature matching. Subsequently, it is necessary to review the front-end processes to identify stages in these processes affected by map point selection. This subsection describes both a general approach to these problems and the specific implementation of the ORB-SLAM 2 algorithm. After this overview, this subsection identifies potential points in this process impacted by map point selection.

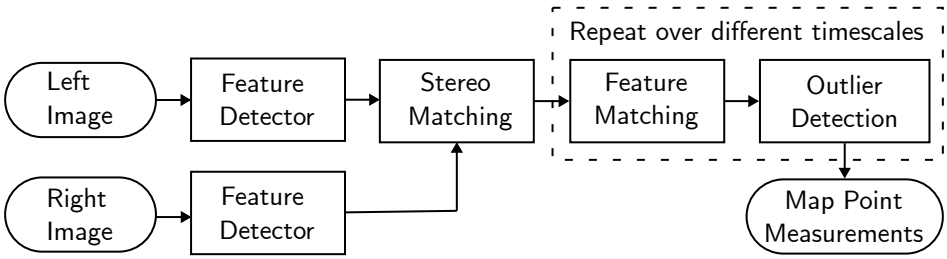

Figure 3.1: A flowchart of the front-end processing for stereo camera images into map point measurements. A feature detector is used to describe texture at salient points in the image using feature descriptors. Stereo matching finds the corresponding points in the image between the left and right images using these descriptors. These features are then matched to existing map points. Feature matching is error-prone and typically requires a further outlier removal step before these abstracted map point measurements are used for SLAM estimation in the back-end. SLAM algorithms such as ORB-SLAM 2, employ different feature matching and outlier detection algorithms over different timescales. This diagram is adapted from Mur-Artal and Tardós [8].

Figure 3.1 provides an overview of the process of the front-end to convert camera images into map point measurements with known correspondence. The first step is to use a “Feature Detector”. Feature detectors identify salient points in images (called key points) and use feature descriptors to encode information about the local texture, making it possible to match these points across different images or frames. Feature descriptors are designed to be stable and be consistently detected across changes in scale, rotation, illumination and viewpoints. Examples of features include SURF [9] and ORB [10]. Each feature descriptor also defines a distance by which features can be compared. For example, binary features such as ORB use hamming distance to compare feature descriptors.

Using a feature detector is followed by ”Stereo Matching”. Since visual descriptors are designed to be robust to changes in viewpoint, these descriptors can be used to identify which pixel location corresponds to measurements of the same point between the left and right cameras. For a pair of cameras, the epipolar constraint restricts the search for corresponding points to a line in the camera image (known as an epipolar line). For rectified stereo cameras, the image planes are parallel and observations of the same point are on the same vertical pixel height [24]. During stereo matching, the features in the left camera image are matched to the right camera image with a similar height using the

descriptor distances. These stereo-matched features are potential measurements of new map points or existing map points. The next step is to identify whether these correspond to existing map points. Suppose there are no map points in the map (or the following steps found that these stereo-matches features do not correspond to existing map points). In that case, new map points are created using these stereo-matched features by triangulating their position, and their associated feature descriptors are stored.

Stereo matching is followed by "feature matching" where matches are found between stereo-matched features from the current set of camera images and the map points in the map. Unlike during stereo matching, these feature matches are not limited to the same vertical pixel height and therefore, different matching algorithms are needed. The most straightforward matching algorithm is brute force matching. Brute force matching iterates over all features in the current image and assigns each feature measurement to the closest feature descriptor. Brute force matching requires comparing each feature descriptor against all candidates and is expensive. SLAM algorithms employ approximate nearest-neighbour techniques to approximate brute-force matching. These approaches reduce the number of descriptor comparisons by first grouping similar feature descriptors together (as defined by the technique) and using these groupings to find similar feature descriptors efficiently. These accelerated matching approaches include fast local approximate nearest neighbour (FLANN) matching, which uses a KD-tree [40] to discretise vector spaces or local sensitivity hashing for binary features [41]. ORB-SLAM 2 uses a different clustering-based technique for approximate nearest-neighbour matching [27].

ORB-SLAM 2 uses both the previous approximate nearest neighbour feature matching approach and a matching approach, which we label "guided matching". This guided matching algorithm is used when the transformation between camera poses is known, as well as the map point positions (or can be estimated accurately). Map points are projected into the camera frame using the estimate of the relative motion and map point positions. These expected pixel locations are then used to search for nearby stereo-matched features in the image plane with similar feature descriptors to perform feature matching.

While feature descriptors are designed to be robust to changes in viewpoints, feature matching can be prone to errors due to occlusions, lighting changes, or repetitive texture in the images that lead to ambiguous matches. After feature matching, it is common to employ an outlier detection algorithm. The random sample consensus (RANSAC) algorithm [42] is a popular outlier detection algorithm employed by SLAM algorithms. RANSAC is an iterative method that aims to robustly estimate model parameters from a set of data points that may contain outliers. In the context of feature matching, RANSAC helps to identify and discard incorrect matches by iteratively selecting a minimal set of feature matches and fitting a model (e.g., the rigid transformation between camera coordinate frames) to them. The RANSAC algorithm then determines how many other feature matches are consistent with this model. It is assumed that the map underwent

a rigid transformation, and random subsets of map points are selected to estimate this transformation. This random selection is repeated multiple times, and the model with the highest number of consistent matches or inliers is considered the best estimate. After outlier detection, the matched feature measurements are treated by the SLAM algorithm as map point measurements.

ORB-SLAM 2 uses multiple iterations of matching algorithms and outlier detection and switches between matching algorithms based on heuristics. ORB-SLAM 2 first uses a motion model to predict the camera position and orientation using the estimated velocity and performs guided matching to find features that match with the camera frame at a previous timestep. The camera pose is approximately estimated using these feature matches, and outliers are removed. Suppose the guided matching does not produce enough feature matches. In that case, feature matching is performed using nearest neighbour techniques followed by the RANSAC outlier detection algorithm. If guided matching or feature matching finds sufficient matches, the current pose is estimated using the matched features. A new round of guided matching follows this initial feature matching to match map points in the current camera frame to the local map that includes all the map points in co-visible camera frames.

Another important role of the front-end is to detect loop closures; that is, when the robot revisits a previously explored area. Before feature matching can be applied in these scenarios, potential candidates for loop-closures are first identified using appearance-based methods that encode the appearance of an image to allow retrieval of locations in the map with similar appearance. It is common for visual SLAM implementations [29; 8] to use bag-of-word models, e.g. FAB-MAP [43] or Galvez-López and Tardos [27]. These models are classical approaches used for document retrieval in natural language processing, but these approaches have been adapted to retrieving images. When applied to images, these models represent the appearance of an image using a histogram of the feature descriptor content of an image. These models are first trained on features extracted from separate datasets. The features are clustered into a vocabulary of different "words", and infrequent words are assigned higher weights. When processing frames for SLAM, the feature descriptors associated with each frame are assigned to the closest words and a bag-of-words histogram is generated. Different camera frames are matched using the difference in these bag-of-word histograms. Similar to feature matching, this appearance-based matching is prone to errors.

ORB-SLAM 2 requires several consecutive frames to match the same loop-closure candidate using appearance-based matching before proceeding. After identifying camera frames with similar appearance, feature matching is used between the two camera frames to identify the matches between features. If an insufficient number of feature matches are found, this loop-closure candidate is rejected. Otherwise, the camera pose is estimated using these initial feature matches. This is followed by an additional stage of guided

matching, which is applied to match the stereo-matched features within the current frame with those found in the local map of the loop-closure candidate. The loop closure candidate is also rejected if an insufficient number of local map point matches is found after using guided matching.

SLAM algorithms typically cannot recover from incorrect loop closures. This limitation requires loop-closure detection algorithms to be very selective when allowing detected loop closures. As a consequence, algorithms such as ORB-SLAM 2 choose a high number of matched features before accepting a loop closure. This choice improves the ability of the loop-closure detection algorithm to reject incorrect loop closures at the cost of potentially rejecting correct loop closures.

This subsection provided an overview of the visual SLAM front-end. We now consider how map point selection will impact the front-end of a visual SLAM algorithm. The selection of map points will result in fewer map points being available for the various stages of feature matching and outlier detection. Notably, stable points in the environment that yield consistent feature matches can still be used by the SLAM algorithm at future timesteps, even if map point selection removes the associated map points. However, since the associated feature information and observations of these map points have been removed by selection, feature matches of these points are instead treated as feature matches of potential new map points, rather than observations of existing map points. Consequently, map point selection will reduce the number of longer-term feature matches. We expect that this reduction will result in fewer map point matches for loop-closure detection and negatively impact loop-closure performance.

3.1.2. The Visual SLAM Back-end Estimation Problem

The visual SLAM back-end uses measurements generated by the front-end (as described in Subsection 3.1.1) to formulate the simplified SLAM estimation problem using these abstracted measurements. These abstracted measurements include not only the pixel locations where map points are measured in the pair of stereo cameras, but also the identity of the map points, which is assumed to be accurate. Map point selection will impact the map points and associated measurements included in the SLAM estimation problem. This subsection provides an overview of the SLAM estimation problem and relevant information-theoretic quantities to measure this impact.

The back-end formulates the SLAM problem as the joint estimation of the robot poses, or $\mathbf{x}_{1:t}$, a set of uniquely identifiable positions of map points, $\mathbf{m}_{1:n}$, and the respective pixel locations $\mathbf{z}_{1:t}$, where those points were observed, or equivalently as

$$P(\mathbf{x}_{1:t}, \mathbf{m}_{1:n} | \mathbf{z}_{1:t}). \tag{3.1}$$

This dissertation uses n for the total number of map points and t for the total number of

estimated robot poses. Ranges, such as $1{:}n$, indicate the range of indices from 1 to n. This dissertation generally uses i as an index associated with map points and j as an index associated with robot poses.

Maximum a posterior (MAP) estimation is commonly used in SLAM to reduce the estimation to a non-linear least-squares problem. We present an overview of this approach similar to the approach presented in Thrun [2]. The posterior distribution in Equation (3.1) can be rewritten into a factorised form after the common assumptions that map point observations are independent and by applying Bayes' rule. Additionally, assuming an informative prior over both map point locations and robot states yields the factorised posterior distribution, or

$$P(\mathbf{x}_{1:t}, \mathbf{m}_{1:n}|\mathbf{z}_{1:t}) = \eta P(\mathbf{x}_{1:t}) \prod_{j=1}^{t} \prod_{i=1}^{n} P(\mathbf{z}_j^i|\mathbf{x}_i, \mathbf{m}_j), \tag{3.2}$$

where η is a normalising constant.

The maximum a posteriori probability (MAP) estimate is found by finding the maximum of the posterior distribution, which is equivalent to minimising the negative log posterior. To simplify the notation, we define the vector $\mathbf{y}$ as the joint vector of robot poses and map points, or

$$\mathbf{y} = \begin{pmatrix} \mathbf{x}_{1:t} \\ \mathbf{m}_{1:n} \end{pmatrix}. \tag{3.3}$$

The negative log of the posterior distribution of Equation (3.2) yields the sum of the factors,

$$\begin{aligned} l(\mathbf{y}) &= -\log P(\mathbf{y}|\mathbf{z}_{1:t}) \\ &= \text{const.} + \frac{1}{2} \sum_{j=1}^{t} \sum_{i=1}^{n} \left(\mathbf{h}_{ij}(\mathbf{y}) - \mathbf{z}_j^i\right)^{\mathrm{T}} \mathbf{\Omega}_{ij} \left(\mathbf{h}_{ij}(\mathbf{y}) - \mathbf{z}_j^i\right), \end{aligned} \tag{3.4}$$

where for each observed map point measurement $\mathbf{z}_j^i$ the function $\mathbf{h}_{ij}$ describes the predicted measurement, given the map point $\mathbf{m}_i$ and camera pose $\mathbf{x}_j$. We assume Gaussian noise for the measurement of map point i at timestep j, with an information matrix (or inverse covariance matrix) of $\mathbf{\Omega}_{ij}$.

We can approximate the non-linear posterior distribution in Equation (3.4) around the current estimate of $\mathbf{y}$ represented by $\boldsymbol{\mu}_k$. We define the Jacobian of the measurement function $\mathbf{h}_{ij}$ as $\mathbf{J}_{ij}$, which yields the Taylor approximation of the measurement function, or

$$\mathbf{h}_{ij}(\mathbf{y}) \approx \mathbf{h}_{ij}(\boldsymbol{\mu}_k) + \mathbf{J}_{ij}(\mathbf{y} - \boldsymbol{\mu}_k). \tag{3.5}$$

By substituting the Taylor approximation of Equation (3.5) into Equation (3.4) and gathering the associated terms, we obtain a linear approximation of the negative log-posterior.

This linear approximation can be rewritten into a quadratic form with information vector $\mathbf{b}$, information matrix $\mathbf{\Lambda_y}$ and a constant, or

$$l(\mathbf{y}) \approx \frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{\Lambda_y}\mathbf{y} - \mathbf{b}^{\mathrm{T}}\mathbf{y} + \text{const.} \tag{3.6}$$

This linear approximation in Equation (3.6) is used by the SLAM implementations to perform least-squares estimation to update the current estimate of the robot poses and map points. For a more in-depth discussion on least-squares estimators and SLAM implementations, we direct the interested reader to the tutorial by Grisetti *et al.* [23] and toolboxes developed for SLAM estimation (e.g. g^{2}o [44], GTSAM [45] and SLAM++ [46]). It is important to note that these approaches to SLAM do not calculate the covariance matrix, but the information matrix is readily available, as well as the derivatives and noise models. The approximate information matrix $\mathbf{\Lambda_y}$ can be calculated directly from all the quadratic terms from the linear approximation of the posterior in Equation (3.2) as

$$\mathbf{\Lambda_y} = \sum_{j=1}^{t}\sum_{i=1}^{n}\mathbf{J}_{ij}^{\mathrm{T}}\mathbf{\Omega}_{ij}\mathbf{J}_{ij}, \tag{3.7}$$

where $\mathbf{J}_{ij}$ is the Jacobian of the measurement function of map point i from pose j, and we assumed Gaussian measurement noise with an information matrix (or inverse covariance matrix) of $\mathbf{\Omega}_{ij}$. This dissertation uses this approach to directly calculate the information matrix from the non-linear posterior using the associated measurement Jacobians and noise precision matrices. This avoids substituting the Taylor approximation Equation (3.5) into the non-linear posterior distribution in Equation (3.4) and gathering all the terms.

We now consider the problem of evaluating the impact of map point selection on the back-end. Map point selection will impact the measurements and map points included in the estimation. As noted in Chapter 1, all map point measurements of map points not selected are discarded to limit the map data. Information-theoretic measures are a means to measure the impact of this selection and one such a measure is the differential entropy. The differential entropy is a measure of the uncertainty or the average amount of "suprise" associated with variables outcomes. When selecting map points for SLAM, it is desirable for the entropy of the estimated states to be low. For the random variable $\mathbf{y}$ with the general probability density function $P(\mathbf{y})$ defined over the domain Y, the differential entropy is defined as

$$H(\mathbf{y}) = -\int_{Y} P(\mathbf{y})\log(P(\mathbf{y}))d\mathbf{y}. \tag{3.8}$$

A common approximation is to use the linear Gaussian approximation of estimation problems to approximate relevant quantities of the non-linear estimation problems [20; 22]. For a general multivariate Gaussian distribution the entropy from Equation (3.8) simplifies to a function of the information matrix $\mathbf{\Lambda_y}$, which is the inverse of the covariance matrix,

and the dimension d of the matrix, or

$$H(\mathbf{y}) = \frac{d}{2} + \frac{d}{2}\log(2\pi) - \frac{1}{2}\log(|\mathbf{\Lambda_y}|). \tag{3.9}$$

The information gain for variables $\mathbf{y}$ is the reduction in entropy when knowledge of variables $\mathbf{z}$ is added. For a prior distribution $H(\mathbf{y})$ and a posterior distribution, or $H(\mathbf{y}|\mathbf{z})$, the information-gain is given by

$$IG(\mathbf{y}, \mathbf{z}) = H(\mathbf{y}) - H(\mathbf{y}|\mathbf{z}). \tag{3.10}$$

It is desirable for the information gain to be high as this implies a large reduction in entropy or uncertainty with respect to the prior distribution. For Gaussian distributions we substitute Equation (3.9) into Equation (3.10) to obtain the information-gain in terms of the difference between the posterior information matrix $\mathbf{\Lambda_{y|z}}$ and prior information matrix $\mathbf{\Lambda}_y$, or

$$IG(\mathbf{y}, \mathbf{z}) = \frac{1}{2}\log(|\mathbf{\Lambda_{y|z}}|) - \frac{1}{2}\log(|\mathbf{\Lambda_y}|). \tag{3.11}$$

For optimisation problems over the selected map points, it is beneficial to consider many potential map point selections. Using the posterior distribution that results from a selection, only the associated information matrix is needed to evaluate the information gain of the linear approximation as shown in Equation (3.11). This information matrix can be directly calculated from the sum of the information matrices of each measurement using the noise precision matrix and the measurement Jacobians as shown in Equation (3.7).

Map point selection will impact the back-end estimation by removing map points and the associated measurements from the SLAM estimation. This subsection provided an overview of the SLAM estimation problem as formulated in the back-end components of SLAM. For map point selection, it will be beneficial to efficiently evaluate the impact of map point selection on this estimation problem. This impact of map point selection can be measured using information gain and the associated information matrices.

3.1.3. Stereo Cameras for Visual SLAM Estimation

Subsection 3.1.2 provided an overview of the SLAM estimation problem as formulated by the back-end of a SLAM implementation and argued that information gain can be used to formulate the impact of map point selection on the back-end of a SLAM implementation. The aforementioned subsection also showed that the Jacobian of the measurement function is needed to evaluate the information gain. Consequently, this subsection provides an overview of the stereo camera measurement model used by visual SLAM, or $\mathbf{h}_{ij}(\mathbf{x}_j, \mathbf{m}_i)$.

Before we consider the measurement model of stereo cameras, it is beneficial to consider the monocular camera example. Figure 3.2 illustrates the pinhole camera model that

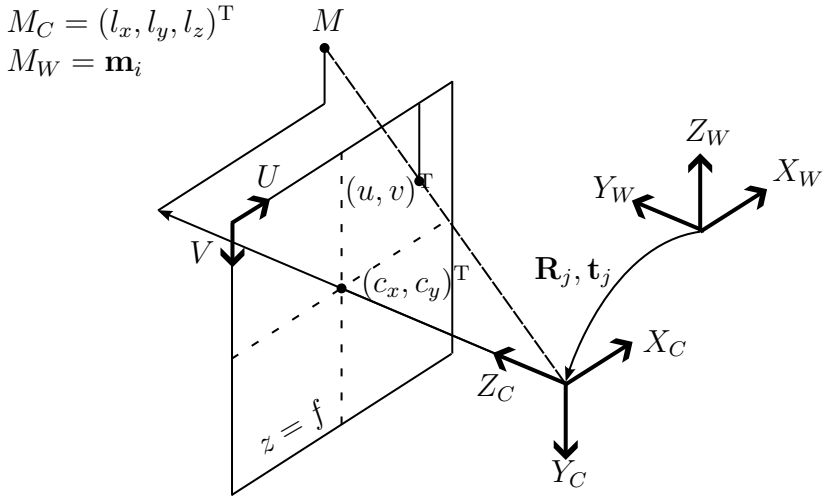

Figure 3.2: Visualisation of the pinhole camera model that describes the relationship between a point in 3-D space M and its camera projection of the image plane $(u, v)^{\mathrm{T}}$. The global coordinate frame is given by X_W, Y_W and Z_W, while the camera coordinate frame is given by X_C, Y_C and Z_C. The transformation $\mathbf{R}_j$, $\mathbf{t}_j$ describes the transformation of points from the global to camera coordinate frames (adapted from [47]).

describes the relationship between a point in 3-D space and its projection onto the image plane of a camera. The optical centre of the camera is located at the origin of the camera coordinate frame, with the principal axis facing the Z_C axis of the camera axis. The image plane is located at a specified focal length f along the principal axis. The intersection of the principal axis and the image plane is located at the image coordinates $(c_x, c_y)^{\mathrm{T}}$. The projected image location (u, v) of the point M located at $(l_x, l_y, l_z)^{\mathrm{T}}$ in the camera coordinate system is given by

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} l_x \\ l_y \\ l_z \end{pmatrix}, \tag{3.12}$$

where s is a scaling factor from the arbitrary scaling from the projection. The projected image location can equivalently be written as

$$p_{\mathrm{mono}}(l_x, l_y, l_z) = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_x \frac{l_x}{l_z} + c_x \\ f_y \frac{l_y}{l_z} + c_y \end{pmatrix}. \tag{3.13}$$

For the map point with index i, the position $\mathbf{m}_i$ is defined in the global coordinate frame. It is first necessary to calculate the position of the point in the camera coordinate frame, or M_C. The transformation of a point in the global coordinate frame to the camera coordinate frame can be described in terms of the 4×4 transformation matrix, or $\mathbf{T}_j$. This transformation matrix is the combination of a rotation matrix $\mathbf{R}_j$ and a translation

$\mathbf{t}_j$, or

$$\mathbf{T}_j = \begin{pmatrix} \mathbf{R}_j & \mathbf{t}_j \\ \mathbf{0} & 1 \end{pmatrix}. \tag{3.14}$$

The map point position in the camera coordinate frame in homogeneous coordinates is given in terms of the matrix product of the transformation matrix $\mathbf{T}_j$ and $\mathbf{m}_i$, or

$$\begin{pmatrix} l_x \\ l_y \\ l_z \\ 1 \end{pmatrix} = \mathbf{T}_j \begin{pmatrix} \mathbf{m}_i \\ 1 \end{pmatrix}. \tag{3.15}$$

The projected monocular map point measurement for a map point in the global coordinate frame using Equation (3.15) and the definition of the function p_{mono} from Equation (3.13) simplifies to

$$\begin{pmatrix} u \\ v \end{pmatrix} = p_{\text{mono}}(\mathbf{R}_j \mathbf{m}_i + \mathbf{t}_j). \tag{3.16}$$

Stereo vision SLAM uses two synchronised cameras placed at a known distance from one another. Stereo vision SLAM has a significant advantage over monocular approaches in that it allows the scale of the mapped environment to be observed and allows triangulating the position of new map points from a single pair of stereo pixel measurements.

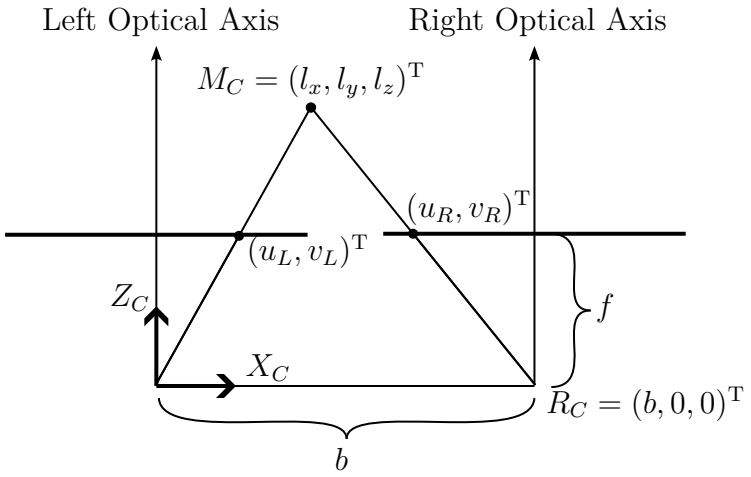

Figure 3.3: Rectified stereo camera pair viewed from above. Stereo cameras are placed with parallel optical axes. The left camera is chosen as the origin. The right camera is placed at known baseline distance b away from the left camera. The map point M_C is projected onto the left camera image plane at $(u_L, v_R)^{\mathrm{T}}$ and right camera image plane at $(u_R, v_R)^{\mathrm{T}}$. These image planes are located at a focal length f along the Z_C axis.

Figure 3.3 depicts a rectified stereo camera setup where the left camera is located at the origin of the left camera axis (shown by X_C and Z_C with Y_C facing into the page). An identical right camera is located at a known baseline distance, or b, along the X_C axis and

both cameras are facing parallel to the Z_C-axis. For rectified stereo cameras the projected height of the pixel positions are identical between the left and right cameras, $v_L = v_R$. The project stereo-camera pixel measurement is given by the left camera pixels $(u_L, v_L)^{\mathrm{T}}$ and the right camera pixel u_R, or

$$p_{\text{stereo}}(l_x, l_y, l_z) = \begin{pmatrix} u_L \\ v_L \\ u_R \end{pmatrix} = \begin{pmatrix} f_x \dfrac{l_x}{l_z} + c_x \\ f_y \dfrac{l_y}{l_z} + c_y \\ f_x \dfrac{l_x - b}{l_z} + c_x \end{pmatrix}. \tag{3.17}$$

For SLAM estimation we are interested in the pixel positions given a map point with index i and robot pose j, or $\mathbf{h}_{ij}(\mathbf{x}_j, \mathbf{m}_i)$. The predicted map point measurement $\mathbf{h}_{ij}(\mathbf{x}_j, \mathbf{m}_i)$ is given by substituting the transformation from global to camera coordinates from Equation (3.15) into Equation (3.17), or

$$\mathbf{h}_{ij}(\mathbf{x}_j, \mathbf{m}_i) = p_{\text{stereo}}(\mathbf{R}_j \mathbf{m}_i + \mathbf{t}_j) \tag{3.18}$$

Visual SLAM implementations commonly estimate the pose of the robot in terms of this rotation matrix $\mathbf{R}_j$ and translation $\mathbf{t}_j$ that describes the transformation from the global coordinate frame to the camera coordinate frame. This is the inverse of the transformation that describes the camera position and orientation with respect to the global coordinate frame. The position and orientation of the camera in the global coordinate frame can be recovered from the inverse of this transformation, or

$$\mathbf{T}^{-1} = \begin{pmatrix} \mathbf{R}_j^T & -\mathbf{R}_j \mathbf{t}_j \\ \mathbf{0} & 1 \end{pmatrix}. \tag{3.19}$$

While the rotation matrix $\mathbf{R}_j$ is a 3×3 matrix, a rotation matrix only has 3 degrees of freedom. The pose only has 6 degrees of freedom, and the map point position has 3 degrees of freedom. The exact parametrisation of the pose is implementation-specific, but it is common to use a minimal representation. The measurement Jacobian $\mathbf{J}_{ij}$ is the derivative of Equation (3.18) with respect to the estimated robot pose $\mathbf{x}_j$ and map point position $\mathbf{m}_i$ and is, therefore, a 3×9 matrix.

This section presents an overview of the visual SLAM stereo measurement model. Map point selection will impact the SLAM estimation problem of the back-end of a visual SLAM algorithm. Subsection 3.1.2 showed information-theoretic approaches to measure this impact using the associated information matrices. The information matrix from map point measurements is given by the Jacobian of the measurement model in this subsection and the associated measurement noise, as shown in Equation (3.7).

3.2. Submodular Functions and Greedy Algorithms

This dissertation proposes modelling the map point selection problem as an optimisation problem. Functions from literature for related problems mentioned in Chapter 2 were either information-theoretic, using objective functions such as information-gain mentioned in Subsection 3.1.2, or coverage-based, using objective functions based on the number of map points visible in a frame. Before we consider specific formulations, it is beneficial to consider optimisation algorithms applicable to these functions.

This section considers the general map point selection problem where for the set of available map points, V, and a selected subset, S, there is a function $f(S)$ that models the utility of S. Section 1.2 assumed relevant hardware constraints for map point selection (limited memory, storage space or communication budget) can be translated to a limited budget of map points k. This constraint on the total number of map points in the set, where $|S|$ is the cardinality of S, is referred to as a *cardinality constraint*. During map point selection, we wish to find the subset of map points that maximises the utility while satisfying the cardinality constraint, or

$$S^* = \underset{S \subseteq V, |S| \leq k}{\operatorname{argmax}} f(S). \tag{3.20}$$

This formulation of the map point selection problem is challenging in general, especially given the stringent requirements on computational requirements for use in online applications. Fortunately, both existing information-theoretic and coverage-based formulations used in existing work for related problems are submodular set functions. For set functions where the function $f(S)$ is monotone, submodular and normalised, greedy algorithms can be used to approximately maximise the problem in Equation (3.20) with strong guarantees using only a polynomial number of function evaluations. Greedy algorithms offer computationally inexpensive approaches for maximising such functions. The approach of this dissertation is instead to focus on the problem of designing suitable utility functions to use with these greedy algorithms to design computationally inexpensive map point selection approaches for SLAM.

Subsection 3.2.1 provides an overview of these mentioned properties of set functions; that is, submodular, monotone and normalised. Subsection 3.2.2 presents relevant greedy algorithms for maximising these functions subject to a cardinality constraint, such as maximising a utility function when selecting a limited number of map points. While not the focus of this dissertation, there are variations of the greedy algorithm that consider a wider range of constraints, as noted in Appendix A. This dissertation will use these greedy algorithms for cardinality-constrained problems to maximise the utility functions for map point selection developed in Chapter 4 to select map points for visual SLAM.

3.2.1. Properties of submodular functions

The utility functions in this dissertation that describe the utility of a set of map points belong to a class of set functions known as submodular functions. This subsection briefly describes submodular functions and relevant properties for using these functions with greedy algorithms [48].

We consider a discrete set function that assigns each subset $S \subseteq V$ a value $f(S)$, or

$$f : 2^V \rightarrow \mathbb{R}. \tag{3.21}$$

We define the marginal gain of element $m_i \in V$ as the increase in function value of adding the element m_i to the set S, or

$$f(m_i|S) = f(S \cup \{m_i\}) - f(S). \tag{3.22}$$

The function f is submodular if, for any sets A and B with $A \subseteq B \subseteq V$, the marginal gain has the property of diminishing returns, or

$$f(m_i|A) \geq f(m_i|B). \tag{3.23}$$

The set function $f(S)$ is monotone if, for any sets A and B with $A \subseteq B$, it has the property that

$$f(A) \leq f(B). \tag{3.24}$$

Lastly, the set function is normalised if, for the empty set $\emptyset = \{\}$,

$$f(\emptyset) = 0. \tag{3.25}$$

The utility functions developed in this dissertation for the map point selection problem are submodular, monotone and normalised (Equation (3.23), Equation (3.24) and Equation (3.25), respectively). These properties determine what algorithms are suitable for maximising these functions.

3.2.2. Algorithms for submodular maximisation subject to a cardinality constraint

This dissertation formulates the map point selection problem as a problem where the cardinality of the selected set of map points is constrained. Most of the utility functions for map point selection proposed in this dissertation are submodular functions. This subsection provides an overview of the relevant algorithms to maximise such functions under cardinality constraints and approximately solve the resulting map point selection

problem (see the survey article by Krause and Golovin [48] for additional information on submodular maximisation problems).

For the submodular maximisation problem subject to a cardinality constraint, we are given submodular function $f(S)$, we are interested in selecting some optimal subset S^* of at most k items that maximises a utility $f(S)$, or

$$S^* = \underset{S \subseteq V, |S| \leq k}{\operatorname{argmax}} f(S). \tag{3.26}$$

This problem is difficult in general and NP-hard for many cases of submodular functions [49]. This hardness classification implies that there are no known efficient algorithms to solve this problem optimally in polynomial time, and as a result, finding the optimal solution becomes infeasible as the problem size increases. Fortunately, greedy algorithms often provide strong guarantees for submodular functions. If $f(S)$ is a monotone, normalised and submodular function subjected to cardinality constraint k, the classic greedy algorithm returns a $1 - e^{-1}$ approximation [50]; that is, the utility obtained by the greedy solution S_g is at least within a constant factor of the function value obtained by the intractable optimal solution, or

$$f(S_g) \geq \left(1 - e^{-1}\right) f(S^*). \tag{3.27}$$

The classic greedy algorithm is described in Algorithm 1.

Algorithm 1: Classic greedy algorithm for monotone cardinality constrained submodular maximisation.

input : Utility function $f(S)$, set of items V and budget k.
output : S_g

$S_g \leftarrow \emptyset$
$V_g \leftarrow V$ `// Used to represent` $V \setminus S_g$
for $i \leftarrow 1$ **to** k **do**
  $m_i \leftarrow \underset{m_i \in V_g}{\operatorname{argmax}} f(m_i|S_g)$
  $S_g \leftarrow S_g \cup \{m_i\}$
  $V_g \leftarrow V_g \setminus \{m_i\}$

As shown in Algorithm 1, the classic greedy algorithm requires $n \times k$ function evaluations of the marginal gain, or $f(m_i|S)$, where n is the number of items in V and k is the budget or the number of selected items. If the budget k grows proportional to n, the greedy algorithm requires $O(n^2)$ function evaluations.

Minoux [51] proposes an improved lazy greedy algorithm that uses submodularity to reduce the number of function evaluations. An implementation of the lazy greedy algorithm is shown in Algorithm 2. Instead of computing the marginal gain for all items at every iteration, the lazy greedy algorithm maintains the upper bounds on the marginal gains in decreasing order. At every iteration, the lazy greedy algorithm evaluates the

Algorithm 2: The lazy greedy algorithm for monotone cardinality constrained submodular maximisation.

input : Utility function $f(S)$, set of items V and budget k.
output : S_g

$S_g \leftarrow \emptyset$
$n \leftarrow |V|$
Buffer $\leftarrow$ Allocate a buffer of n {item, value} pairs
for $i \leftarrow 1$ **to** n **do**
  buffer[i] $\leftarrow \{m_i, f(m_i|S_g)\}$ // $O(g(n))$
PQ $\leftarrow$ Priority queue that sorts {item, value} pairs by descending value.
Initialise PQ using buffer // $O(n)$
while $|S_g| < k$ **do**
  /* at most $O(kn)$ iterations */
  $m_{\max}, v_{\text{old}} \leftarrow$ PQ.pop() // heap: $O(\log(n))$
  $v_{\text{new}} \leftarrow f(m_{\max}|S_g)$ // $O(g(n))$
  $m_{\text{top}}, v_{\text{top}} \leftarrow$ View next best item in PQ // $O(1)$
  if $v_{\text{new}} >= v_{\text{top}}$ **then**
    /* at most k iterations */
    $S_g \leftarrow S_g \cup \{m_i\}$
    // Update $f(m|S_g)$ implementation
  else
    PQ.insert($\{m_{\max}, v_{\text{new}}\}$) // heap:$O(\log(n))$

marginal gain for the element with the highest upper bound, or the item $m_{\max}$. The lazy greedy algorithm then evaluates the marginal gain of $m_{\max}$, or $f(m_{\max}|S_g)$. If this is larger than the upper bound of the next best item, we have found the item with the highest marginal gain, and it is selected. Otherwise, the calculated marginal gain is the new marginal gain upper bound for the item stored with the value $m_{\max}$. The lazy greedy algorithm is typically implemented using a priority queue to iterate over items in the order of their marginal gain upper bounds [52]. The lazy greedy algorithm shares the same worst-case function evaluations of $O(kn)$ or $O(n^2)$ if $k = O(n)$ as the classic greedy algorithm. However, it is often significantly faster in practice [51].

For bounded integer functions, a bucket queue implementation can be used [53]. A bucket queue implementation allows for $O(1)$ queue insertion and removals the maximum element by tracking the largest bucket. For the more general case of floating point numbers, a typical implementation of a priority queue is a max-heap [54]. We include the computational complexity of heap operations for a max-heap as comments in Algorithm 2. Inserting and removing items into max-heap has a computational complexity of $O(\log(n))$. The asymptotic computational complexity of this lazy greedy algorithm implementation using heaps is, therefore, $O\left(kn \times (\log(n) + g(n)\right)$, where $g(n)$ is the cost of evaluating marginal gain and the $\log(n)$ term results from the heap operations. This logarithmic term is only significant if the asymptotic computational complexity of the marginal gain

$O(g(n))$ is less than $\log(n)$. However, in practical problems, we found that function evaluations almost always dominate run times, and the lazy greedy algorithm often performs significantly better than suggested by the worst-case asymptotic performance.

Algorithm 3: The stochastic greedy algorithm for monotone cardinality constrained submodular maximisation.

input : Utility function $f(S)$, set of items V, budget k, approximation paramater ϵ.
output : S_g

$n \leftarrow |V|$
$r \leftarrow \frac{n}{k}\log(\frac{1}{\epsilon})$
$S_g \leftarrow \emptyset$
$V_g \leftarrow V$ `// Used to represent` $V \setminus S_g$
for $i \leftarrow 1$ **to** k **do**
  $R \leftarrow$ Select random subset of r elements from V_g
  $m_i \leftarrow \underset{m_i \in R}{\operatorname{argmax}} f(m_i|S_g)$
  $S_g \leftarrow S_g \cup \{m_i\}$
  $V_g \leftarrow V_g \setminus \{m_i\}$
  `/* Update` $f(m|S_g)$ `implementation */`

An efficient stochastic variation of the lazy greedy algorithm was proposed by Mirzasoleiman *et al.* [39] and is shown in Algorithm 3. This algorithm adapts the classic greedy algorithm, where rather than selecting the item with the highest marginal gain from all of the remaining items, it only evaluates a random subset of the remaining items. The stochastic greedy algorithm requires only $O(n\log(\frac{1}{\epsilon}))$ function evaluations, while having a weaker approximation guarantee of $1 - e^{-1} - \epsilon$ and only on the expected function value of the utility, or

$$E[f(S_g)] \geq \left(1 - e^{-1} - \epsilon\right) f(S^*), \tag{3.28}$$

where $f(S^*)$ is the utility of the optimal intractable set S^*. Optimising a utility function with the stochastic greedy algorithm has an asymptotic complexity of $O(n\log(\frac{1}{\epsilon})g(n))$, where $g(n)$ is the computational complexity of evaluating the marginal gain. The stochastic greedy algorithm offers an approximate alternative for scenarios where the lazy greedy algorithm is too computationally expensive.

This section provides an overview of greedy maximisation algorithms used to maximise a wide range of submodular utility functions. The lazy greedy algorithm provides strong performance guarantees with respect to the utility of the selected set. However, it can be substituted for the stochastic greedy algorithm for applications where using the lazy greedy algorithm is prohibitively expensive. These greedy algorithms allow for the development of computationally efficient approaches to map point selection. Since the resulting computational complexity will depend on the computational complexity of evaluating the marginal gain, approaches that can evaluate this quantity efficiently will

result in more scalable approaches.

3.3. Framework for Map Point Selection

Section 3.1 provides an overview of the visual SLAM problem and how map point selection impacts the front- and back-end components. Section 3.2 presents greedy algorithms applicable to the problem of selecting map points for SLAM, while maximising the value or utility of the selected subset of map points. Before this dissertation develops different utility functions for this map point selection problem, this section presents the framework for how map point selection will be used alongside a visual SLAM algorithm.

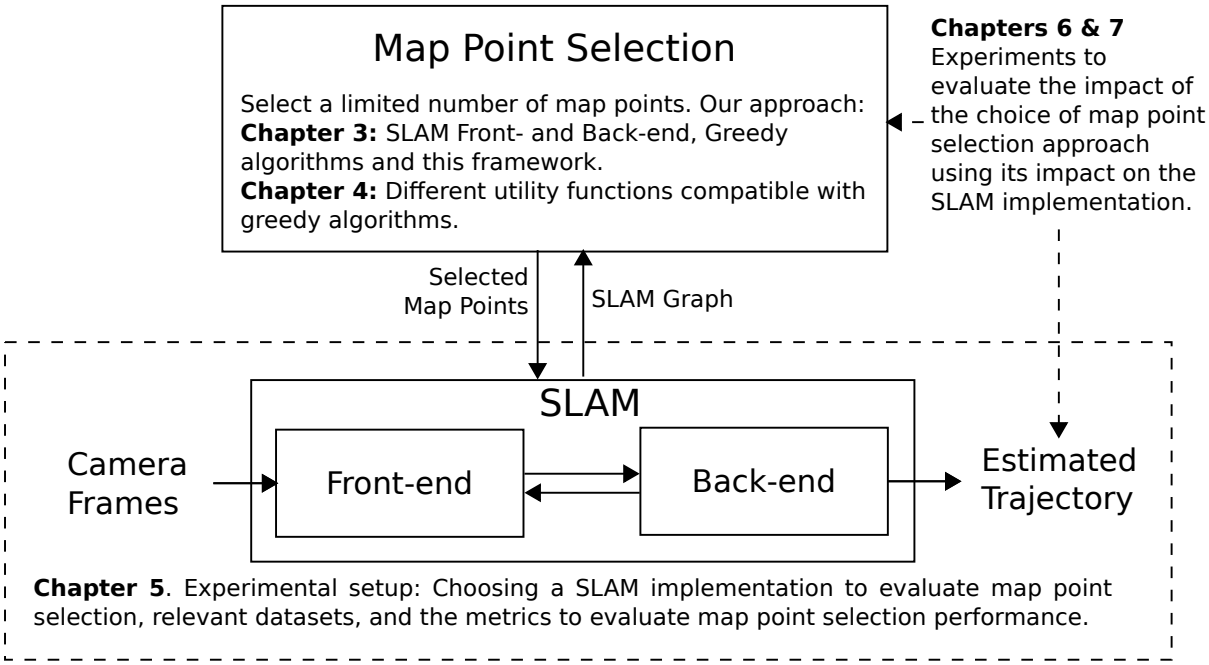

Figure 3.4: Overview of map point selection for visual SLAM and the layout of this dissertation. The different components of the SLAM implementation with map point selection are shown in solid outlines. Solid arrows indicate the flow of information between these components, while dashed boxes and arrows are used to indicate the relevance of sections to different components or emphasise the relationship of sections to items in the diagram.

Since the selection of map points will impact both front- and back-end components of visual SLAM, this dissertation proposes to add map point selection as an additional component, as shown in Figure 3.4. SLAM algorithms process camera frames in the front-end to produce map point measurements used by the back-end to solve the resulting estimation problem. Suppose the map points used by the SLAM algorithm exceed the provided map point budget. In that case, selecting a subset of map points is necessary, which discards all data related to map points not selected, including all map point measurements, estimates and associated feature descriptors. Map point selection can use

any information available to the SLAM algorithm to formulate a map point selection. This information includes the current SLAM graph with robot poses, map points, all the associated map point measurements, and the measurement models of those measurements and the associated noise to formulate a selection of map points.

An approach to map point selection should model the utility of a selected set of map points with sufficient accuracy to allow for the reduction of the map while minimising the impact on the front- and back-end components of a SLAM algorithm. However, it is essential to note that we do not expect this approach to allow reducing the maps to arbitrarily small sizes without impacting the accuracy of the SLAM algorithm. Instead, map point selection automates the process of limiting the number of map points at the cost of estimation accuracy. When the map point budget is limited with respect to the environment, we expect a utility function that accurately captures the underlying utility of map points will minimise the impact of map point selection by selecting a good subset of map points. By comparison, an approach that poorly captures the underlying utility of map points will result in selecting a comparatively poor SLAM accuracy for the same map point budget.

Another consideration is how often can map point selection be applied to reduce the map points within this framework. A SLAM implementation typically creates new map points with every keyframe (the subset of camera frames used for the map), commonly inserted multiple times a second. In the hypothetical scenario where we have map point selection approaches that would require negligible computation time to evaluate, it would be desirable to solve the map point selection problem at a similar frequency to which map points are created. However, given the scale and complexity of the problem, it will be necessary to consider the computation time for calculating map point selection and apply map point selection at a reduced frequency. Given that map point selection for SLAM is likely too expensive to be used at the frame rate of typical cameras, it is not used as a replacement for real-time heuristics which limit the growth of the map (i.e. by extracting a limited number of features or removing recently added map points with few feature map points). Map point selection is proposed as a complementary approach that allows further reduction of the map if necessary due to a limited map point budget.

We recommend implementing map point selection in parallel for online applications (Chapter 7 provides a more detailed discussion of an online implementation). Parallel implementation is a common strategy used by online visual SLAM implementations when an algorithm is too expensive for real-time use. This approach has been successful for PTAM [5] and derivatives such as ORB-SLAM [8]. While using a separate thread for map point selection can mitigate the problem of expensive map point selection approaches to a degree, the computational cost of map point selection remains an important consideration. Suppose a map point selection approach is expensive and requires a long time to calculate a selection. In a parallel implementation, new map points will be created while calculating

the map point selection problem. Even if the approach generates a good selection of map points, the selection may no longer be appropriate for the map that has since been updated. This motivates the development of computationally scalable approaches to map point selection for online problems, even if it is not necessarily required to develop real-time approaches.

This dissertation uses a SLAM software implementation to evaluate the quality of a selected set of map points using the measured impact of the accuracy of the SLAM algorithm. The accuracy of SLAM is typically evaluated using the estimated trajectory. Using a software implementation ensures that the evaluation measures the impact of the selected set of map points on both the front-end and back-end components. This evaluation requires further considerations, such as choosing a representative SLAM implementation and addressing potential challenges that arise from some map point selection approaches not being suitable for online applications and SLAM implementations not being designed for using map point selection approaches.

Figure 3.4 also provides an overview of how the different parts of this dissertation address the map point selection problem for visual SLAM and the evaluation of map point selection approaches. Section 3.2.2 provides an overview of the necessary greedy algorithms. The selected map impacts both the front-end and back-end of a SLAM implementation (described in Section 3.1). Chapter 4 proposes different utility functions optimised with these general greedy algorithms from Section 3.2.2 or function-specific algorithms. Chapter 5 considers the problem of evaluating map point selection approaches using a SLAM implementation. These map point selection approaches are evaluated offline in Chapter 6 and online in Chapter 7.

This section provides an overview of the framework of map point selection for this dissertation and shows how map point selection approaches fit into the larger visual SLAM problem. Furthermore, this section also shows how the discussion of these components is arranged in the different chapters of this dissertation.

Chapter 4

Map Point Selection Approaches

Section 3.2 presents different greedy algorithms applicable to maximisation problems where we are given some utility function $f(S)$ that describes the value of a selected set of map points. For the map point selection problem, we are interested in selecting the "best" subset of map points, i.e. the subset that maximises a given utility while not exceeding the specified budget of map points k, or

$$S^* = \underset{S \subseteq V, |S| \leq k}{\operatorname{argmax}} f(S). \tag{4.1}$$

This chapter proposes different utility functions $f(S)$ that can be approximately maximised with algorithms such as the lazy or stochastic greedy algorithm from Subsection 3.2.2 to approximate the resulting map point selection problem in Equation (4.1) and result in different map point selection approaches. For these greedy algorithms, it is necessary to evaluate the marginal gain, that is, the increase in utility for adding a map point m_i to the currently selected set S from Equation (3.22), or

$$f(m_i|S) = f(S \cup \{m_i\}) - f(S). \tag{4.2}$$

As shown in Subsection 3.2.2, the asymptotic computational complexity of maximising utility functions with greedy algorithms depends on the computational complexity of evaluating the marginal gain in Equation (4.2). Consequently, this chapter evaluates the computational cost of evaluating the marginal gain for the developed utility functions.

We anticipate that to accurately describe the utility of a set of map points for a SLAM implementation, it would be beneficial to consider the impact of map point selection on both of the front-end and back-end of a visual SLAM algorithm. Online visual SLAM algorithms such as ORB-SLAM 2 further separate the SLAM problem into sub-components that operate in different threads at separate frequencies (with the associated labels in parentheses). Localisation (Tracking) is done at framerate, and local-SLAM estimation (Local Mapping) is done at a reduced frequency. Large-scale loop correction is first performed with pose-graph estimation before the full SLAM problem (Bundle adjustment) is performed. Map point selection will impact not only the front- and back-end components, but also the interactions of these sub-components. However, we expect that the utility

function that exactly describes these interactions would likely be both challenging to obtain and efficiently optimise. The approach of this dissertation is to develop simpler, generally applicable utility functions based on approximate models of the SLAM front-end and back-end.

The four sections of this chapter each present different classes of utility functions for map point selection. Section 4.1 details a novel information-theoretic utility based on the SLAM estimation problem. This section uses the information-theoretic concept of information gain as a utility function for SLAM and is based on modelling the utility for the back-end estimation.

Section 4.2 presents two novel approximations of the SLAM utility from Section 4.1. These approaches are based on approximating the posterior distribution over robot poses from the SLAM utility with the posterior distribution corresponding to simpler estimation problems. The first approach approximates the posterior of the robot poses as a set of localisation problems. The second approximation uses a set of stereo-visual-odometry problems. These simplifying approximations are motivated to reduce the computational cost of the SLAM approach, while still considering a utility based on a model for SLAM estimation, albeit simplified.

Section 4.3 presents the development of coverage-based map point selection approaches. These approaches are motivated by the requirement of the front-end to have a sufficient number of map points for feature matching and do not necessarily use the lazy or stochastic greedy algorithms. That section presents three different coverage-based approaches. The first is an existing coverage-based approach by Dymczyk *et al.* [14] and was implemented using integer programming algorithms. This approach is proposed for visual odometry and localisation, but its performance has not yet been evaluated for SLAM. The second is an adaptation of this existing approach into a utility to be compatible with greedy algorithms to improve the asymptotic computational complexity. The last is an alternative approach based on maximising the minimum number of map points in a frame. This last utility is not submodular and requires a different existing greedy algorithm that is also presented in the section.

Lastly, Section 4.4 presents a combined utility function based on combining the approximate information theoretic utility from Subsection 4.2.2, with a coverage-based utility function similar to those in Section 4.3. This approach aims to improve the previous information-theoretic utility by also considering the front-end.

4.1. Utility based on SLAM information gain

This section uses the information-theoretic concept of information gain as a utility for SLAM-based selection problems. This approach can be seen as developing a utility function for the back-end estimation component of a visual SLAM algorithm. The development of

this utility function is detailed in Subsection 4.1.1, along with a naive implementation of this utility function. An alternative implementation of this utility function investigated in this dissertation is presented in Subsection 4.1.2. While this alternative implementation reduces the asymptotic computational complexity, this alternative was not used due to the necessity of calculating and storing the covariance matrix. Lastly, Subsection 4.1.3 presents the implementation of the SLAM utility used in this dissertation and the resulting asymptotic complexity.

4.1.1. Development of the SLAM-based Utility

The performance of SLAM algorithms is often evaluated by the accuracy of the estimated robot trajectories [55]. Therefore, an appropriate choice for a utility function would be to model a point's contribution to the estimation of the robot's trajectory. This section specifically considers the SLAM trajectory as estimated by the back-end. This section specifically considers the SLAM problem as estimated by bundle adjustment or the full SLAM problem and not potential sub-components to generate approximate real-time estimates.

As noted in Chapter 1, when map point selection selects a subset of map points, all other map points and the associated observations are removed. As a result, map point selection not only impacts the SLAM trajectory estimation at the current timestep by removing usable map point observations used by estimation at that timestep, but it also affects the SLAM estimation at future timesteps. This is in contrast to problems where selection is used only to choose what to include in the estimation, but all information is still retained for future timesteps. Therefore, to consider the impact of map point selection, it is necessary to consider potential future observations of those map points and the resulting potential beliefs. An approximation of this intractable full model is to use the SLAM model at the current timestep and select map points to minimise uncertainty over the past trajectory.

The formulation in this section has some similarities to that of Carlone and Karaman [22], which considers the problem of selecting feature measurements for visual inertial-odometry. However, in their formulation, the utility of features is based on the entropy of the current and future poses. Maximum likelihood estimates are used to predict feature observations, which are then treated as known when formulating the selection problem. Rather than just selecting measurements for the current frame, this selects "future" map points and all their associated measurements for visual-inertial odometry. They recognised that this problem is submodular and used the lazy greedy algorithm. This section also considers the marginal distribution over robot poses. However, rather than for a small number of future poses (15 frames 3 seconds into the future), our selection impacts the full SLAM trajectory, which typically involves thousands of frames and hundreds of thousands

of map points. While it is reasonable that the short-term trajectory of a few can be predicted accurately by the planner, the assumption that a planner knows the exact long-term trajectory, especially in an unknown environment, is inappropriate for the SLAM applications. Therefore, we do not predict map point measurements in our formulation.

Kopitkov and Indelman [20] proposes a general framework that, amongst other problems, considers selecting map point measurements based on maximising the information gain of the marginal distribution over the last pose. The utility definition here can be seen as selecting map points (and their associated measurements across all timesteps) for the whole trajectory. Their framework for efficiently evaluating information gain requires that a limited number of states is affected by the selection problem. Unfortunately, map point selection affects all the robot poses in the trajectory and is not compatible with their framework.

We define the baseline SLAM utility of a selected set of map points S as the information gained in the posterior distribution with respect to the prior distribution over the full trajectory of robot poses (i.e. past and current), or

$$f_{\text{SLAM}}(S) = H(\mathbf{x}_{1:t}) - H(\mathbf{x}_{1:t}|\mathbf{z}^S_{1:t}). \tag{4.3}$$

As shown in Subsection 3.1.2, for Gaussian distributions, the information gain simplifies to the difference between the log-determinant of the information matrices. The utility in Equation (4.3) can be rewritten in terms of the posterior information matrix, or $\mathbf{\Lambda}^S_{\mathbf{x}_{1:t}}$, and a constant term based on the prior distribution $c_\emptyset$, as

$$f_{\text{SLAM}}(S) = -\frac{1}{2}c_\emptyset + \frac{1}{2}\log(|\mathbf{\Lambda}^S_{\mathbf{x}_{1:t}}|). \tag{4.4}$$

The rest of this subsection shows the necessary steps to calculate the posterior information matrix over robot poses $\mathbf{\Lambda}^S_{\mathbf{x}_{1:t}}$ to evaluate this utility in Equation (4.4) and discuss the computational complexity of this utility when used by greedy algorithms.

For the selected set of map points, S, we define $\mathbf{m}_S$ as the map point positions corresponding to the selected set and $\mathbf{z}^S_{1:t}$ as the subset of observations corresponding to those selected map points. The marginal information matrix in Equation (4.4) over robot poses is calculated from the joint posterior distribution over robot poses and map points $P(\mathbf{x}_{1:t}, \mathbf{m}_{1:n}|\mathbf{z}^i_{1:t})$. The joint distribution over the trajectory of robot poses and selected map point with indices S and observations $\mathbf{z}^S_{1:t}$ is given by

$$P(\mathbf{x}_{1:t}, \mathbf{m}_{1:n}|\mathbf{z}^i_{1:t}) = \eta P(\mathbf{x}_{1:t}) \prod_{j=1}^{t} \prod_{i \in S} P(\mathbf{z}^i_j|\mathbf{x}_j, \mathbf{m}_i) \tag{4.5}$$

$$= \eta P(\mathbf{x}_{1:t}) \prod_{i \in S} P(\mathbf{z}^i_{1:t}|\mathbf{x}_{1:t}, \mathbf{m}_i), \tag{4.6}$$

The equation in Equation (4.5) is the joint posterior distribution from Equation (3.2) from

Subsection 3.1.2 with only the subset of selected observations and map points, or $\mathbf{z}^S_{1:t}$ and $\mathbf{m}_S$, instead of all of the observations and map points, or $\mathbf{z}_{1:t}$ and $\mathbf{m}_{1:n}$, respectively.

The marginal distribution over robot poses for the selected map points and map point measurements is given by the joint distribution with selected map points marginalised out $\mathbf{m}_S$, or

$$P(\mathbf{x}_{1:t}|\mathbf{z}^S_{1:t}) = \int P(\mathbf{x}_{1:t}, \mathbf{m}_S|\mathbf{z}^S_{1:t})\mathrm{d}\mathbf{m}_S. \tag{4.7}$$

We now substitute Equation (4.5) into Equation (4.7) and manipulate the posterior into the factors associated with each map point, while defining the associated information matrices at the respective steps (indicated using an underbrace), or

$$\underbrace{P(\mathbf{x}_{1:t}|\mathbf{z}^S_{1:t})}_{\mathbf{\Lambda}^S_{\mathbf{x}_{1:t}}} = \eta \int P(\mathbf{x}_{1:t}) \prod_{i\in S} \underbrace{P(\mathbf{z}^i_{1:t}|\mathbf{x}_{1:t}, \mathbf{m}_i)}_{\mathbf{\Lambda}_{\mathbf{x}_{1:t},\mathbf{m}_i}} \mathrm{d}\mathbf{m}_S \tag{4.8}$$

$$= \eta \underbrace{P(\mathbf{x}_{1:t})}_{\epsilon\mathbf{I}} \prod_{i\in S} \underbrace{\int P(\mathbf{z}^i_{1:t}|\mathbf{x}_{1:t}, \mathbf{m}_i)\mathrm{d}\mathbf{m}_i}_{\mathbf{\Lambda}^i_{\mathbf{x}_{1:t}}}. \tag{4.9}$$

We assume an independent and identically distributed Gaussian prior over the robot poses, $P(\mathbf{x}_{1:t})$, with precision ϵ, yielding $\epsilon\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Equivalent to Equation (4.9), the marginal posterior information matrix over the poses $\mathbf{\Lambda}^S_{\mathbf{x}_{1:t}}$ can be written in terms of a prior information matrix plus a term contributed by each selected map point $\mathbf{\Lambda}^i_{\mathbf{x}_{1:t}}$, or

$$\mathbf{\Lambda}^S_{\mathbf{x}_{1:t}} = \epsilon\mathbf{I} + \sum_{i\in S} \mathbf{\Lambda}^i_{\mathbf{x}_{1:t}}. \tag{4.10}$$

The information matrix for each selected map point i, or $\mathbf{\Lambda}^i_{\mathbf{x}_{1:t}}$, is given by the marginal of the joint map point information matrix, or $\mathbf{\Lambda}_{\mathbf{x}_{1:t},\mathbf{m}_i}$, from Equation (4.9). To perform this calculation, we first consider the joint map point information matrix, which is given by the sum of all the measurements of the map point $\mathbf{m}_i$, or

$$\mathbf{\Lambda}_{\mathbf{x}_{1:t},\mathbf{m}_i} = \sum_{j=1}^{t} \mathbf{J}_{ij}^{\mathrm{T}} \mathbf{\Omega}_{ij} \mathbf{J}_{ij}, \tag{4.11}$$

where $\mathbf{J}_{ij}$ and $\mathbf{\Omega}$ are, respectively, the SLAM measurement Jacobian and measurement noise as defined in Subsection 3.1.2. Note that a measurement Jacobian $\mathbf{J}_{ij}$ only involves the corresponding states of the robot pose $\mathbf{x}_j$ and $\mathbf{m}_i$. For the summation in Equation (4.11), the Jacobian with respect to uninvolved states is simply zero. For example, in Equation (4.11), all entries of the Jacobian would be zero, except for the columns that correspond with $\mathbf{x}_j$ and $\mathbf{m}_i$, or

$$\mathbf{J}_{ij} = \begin{pmatrix} 0 & \dots & 0 & \frac{\partial \mathbf{h}_{ij}(\mathbf{x}_j,\mathbf{m}_i)}{\partial \mathbf{x}_j} & 0 & \dots & 0 & \frac{\partial \mathbf{h}_{ij}(\mathbf{x}_j,\mathbf{m}_i)}{\partial \mathbf{m}_i} \end{pmatrix}. \tag{4.12}$$

Before we obtain the marginal information matrix, we first partition the joint informa-

tion matrix $\mathbf{\Lambda}_{\mathbf{x}_{1:t},\mathbf{m}_i}$ into sub-matrices, or

$$\mathbf{\Lambda}_{\mathbf{x}_{1:t},\mathbf{m}_i} = \begin{pmatrix} \mathbf{C}_i & \mathbf{B}_i \\ \mathbf{B}_i^{\mathrm{T}} & \mathbf{P}_i \end{pmatrix} \tag{4.13}$$

where $\mathbf{C}_i$ is the block associated with all the robot poses, $\mathbf{P}_i$ is the block associated with map point i, and $\mathbf{B}_i$ is the block of off-diagonal entries. With the sub-matrix definitions in Equation (4.13), the marginal information matrix over robot poses from all the map point measurements of a map point, or $\mathbf{\Lambda}^i_{\mathbf{x}_{1:t}}$, is given by using the Schur complement on $\mathbf{\Lambda}_{\mathbf{x}_{1:t}\mathbf{m}_i}$, or

$$\mathbf{\Lambda}^i_{\mathbf{x}_{1:t}} = \mathbf{C}_i - \mathbf{B}_i\mathbf{P}_i^{-1}\mathbf{B}_i^{\mathrm{T}}. \tag{4.14}$$

These equations conclude the calculation of the baseline SLAM utility. The information gain in Equation (4.4) is evaluated using the information matrix of the posterior distribution of poses using the selected map points, or $\mathbf{\Lambda}^S_{\mathbf{x}_{1:t}}$. This information matrix is calculated using the sum of information matrices from the selected map points and the prior information matrix, as shown in Equation (4.10). The information matrix for each selected map point is given by first forming the associated joint information matrix Equation (4.11) and applying the Schur's complement as shown in Equation (4.14).

For greedy algorithms such as the lazy greedy algorithm or stochastic greedy algorithm, it is necessary to calculate the marginal gain of a map point at each iteration (refer to Subsection 3.2.2 for more details regarding these algorithms). The marginal gain for the SLAM utility is given by the increase in utility by adding the map point with index m_i to the currently selected set S or $f_{\mathrm{SLAM}}(m_i|S) = f_{\mathrm{SLAM}}(S \cup \{m_i\}) - f_{\mathrm{SLAM}}(S)$. Substituting the SLAM utility from Equation (4.4) into the marginal gain yields

$$f_{\mathrm{SLAM}}(m_i|S) = \frac{1}{2}\log(|\mathbf{\Lambda}^S_{\mathbf{x}_{1:t}} + \mathbf{\Lambda}^i_{\mathbf{x}_{1:t}}|) - \frac{1}{2}\log(|\mathbf{\Lambda}^S_{\mathbf{x}_{1:t}}|). \tag{4.15}$$

The calculation of marginal gain in Equation (4.15) requires the computation of the matrix determinant of a $6t$ by $6t$ matrix, where t is the number of robot poses in the trajectory. An efficient and numerically stable way to evaluate the log-determinant of a positive-definite matrix is to calculate the Cholesky decomposition [56]. The matrix given by the Cholesky decomposition is a triangular matrix. The determinant of a triangular matrix is the product of the terms of the diagonal. The log of the determinant of the posterior information matrix can, therefore, be written in terms of the sum of the log of the terms on the diagonal of the Cholesky decomposition as

$$\log(|\mathbf{\Lambda}_{\mathbf{x}_{1:t}}|) = \log(|\mathbf{L}\mathbf{L}^{\mathrm{T}}|) = 2\log(|\mathbf{L}|) = 2\sum_i \log(\mathbf{L}_{ii}), \tag{4.16}$$

where $\mathbf{L}$ is the Cholesky decomposition of $\mathbf{\Lambda}_{\mathbf{x}_{1:t}}$. The cost of calculating the determinant

is dominated by the computational complexity of evaluating the Cholesky decomposition, which has a computational complexity of $O(t^3)$.

A hypothetical naive implementation of the SLAM utility would use this approach to calculate the utility at every iteration of the greedy algorithm. As shown in Subsection 3.2.2, the lazy greedy and stochastic greedy algorithms have a worst-case complexity of $O(n^2 \times g(n))$ and $O(n \times g(n))$, respectively, where $g(n)$ is the complexity to evaluate the marginal gain and n is the number of map points. Maximising this implementation with the lazy greedy algorithm has a worst-case asymptotic computational complexity of $O(n^2t^3)$, while maximising the utility with the stochastic greedy algorithm has an asymptotic computational complexity of $O(nt^3)$. These asymptotic computational complexities are very high, which implies that such approaches would scale poorly to larger map point selection problems. The following two subsections cover alternative implementations to improve the computational complexity of evaluating the marginal gain in Equation (3.22).

4.1.2. Alternative Implementation of SLAM utility using the Matrix Determinant Lemma

This subsection presents an alternative implementation of the SLAM utility from Subsection 4.1.1. This implementation uses the matrix-determinant lemma, which is a matrix identity which allows the calculation of the determinant after it has been modified in a small way (specifically, by adding a low-rank matrix), in terms of the determinant of the original matrix and the inverse of the original matrix [57]. The determinant lemma is used by Kopitkov and Indelman [20] for a group of general selection problems, but their framework requires that the selection problem only affects a limited subset of the entries in the posterior information matrix. For map point selection, the selection problem affects measurements for all the poses in the trajectory, and this existing application of the determinant-lemma is not applicable.

Before we can apply the matrix determinant lemma, it is necessary to decompose the information matrix that corresponds with the map point with index i, or $\mathbf{\Lambda}^i_{\mathbf{x}_{1:t}}$, into a low-rank update matrix as

$$\mathbf{\Lambda}^i_{\mathbf{x}_{1:t}} = \mathbf{A}_i \mathbf{A}_i^{\mathrm{T}}, \tag{4.17}$$

where $\mathbf{A}_i$ is a low-rank matrix with dimensions $6t \times r$, where r is the rank of the matrix and $r \ll t$. Note that since the information matrix $\mathbf{\Lambda}^i_{\mathbf{x}_{1:t}}$ is only non-zero at the entries that correspond with robot poses where the map point with index i was measured. Similarly, its low-rank decomposition $\mathbf{A}_i$ is only non-zero at all the rows that correspond to poses where the map point is measured.

The posterior information matrix for a selected set of map points S and map point m_i is given by the sum of the posterior information matrix for the set, or $\mathbf{\Lambda}^S_{\mathbf{x}_{1:t}}$, and the corresponding map point information matrix $\mathbf{\Lambda}^i$, or in terms of the low-rank decomposition

in Equation (4.17) as

$$\mathbf{\Lambda}^{S\cup\{m_i\}}_{\mathbf{x}_{1:t}} = \mathbf{\Lambda}^{S}_{\mathbf{x}_{1:t}} + \mathbf{A}_i\mathbf{A}_i^{\mathrm{T}}. \tag{4.18}$$

The matrix determinant lemma allows calculating the determinant of the matrix after it has been updated by a low-rank matrix in terms of the original matrix and its inverse, or

$$|\mathbf{\Lambda}^{S}_{\mathbf{x}_{1:t}} + \mathbf{A}_i\mathbf{A}_i^{\mathrm{T}}| = |\mathbf{\Lambda}^{S}_{\mathbf{x}_{1:t}}||\mathbf{I} + \mathbf{A}_i^{\mathrm{T}}(\mathbf{\Lambda}^{S}_{\mathbf{x}_{1:t}})^{-1}\mathbf{A}_i|. \tag{4.19}$$

where $\mathbf{I}$ is the identity matrix with r rows and columns. The marginal gain for the SLAM utility from Equation (4.15) is given by the equation

$$f(m_i|S) = \frac{1}{2}\log(|\mathbf{\Lambda}^{S}_{\mathbf{x}_{1:t}} + \mathbf{A}_i\mathbf{A}_i^{\mathrm{T}}|) - \log(|\mathbf{\Lambda}^{S}_{\mathbf{x}_{1:t}}|), \tag{4.20}$$

where the information matrix of the map point with index i is instead given in terms of the low-rank decomposition from Equation (4.18). Substituting the matrix determinant lemma into Equation (4.20) simplifies the marginal gain to

$$f(m_i|S) = \frac{1}{2}\log(|\mathbf{I} + \mathbf{A}_i^{\mathrm{T}}(\mathbf{\Lambda}^{S}_{\mathbf{x}_{1:t}})^{-1}\mathbf{A}_i|). \tag{4.21}$$

For the analysis of the computational complexity, we first assume the covariance matrix of the posterior, or $(\mathbf{\Lambda}^{S}_{\mathbf{x}_{1:t}})^{-1}$, is available. The matrix product in Equation (4.21) has a computational complexity of $O(rt^2)$, while the determinant has a computational complexity of $O(r^3)$. The resulting complexity is therefore $O(rt^2 + r^3)$ where t is the number of poses. However, the information matrix associated with a map point $\mathbf{\Lambda}^i$ is typically sparse and only non-zero at the entries that correspond to co-visible robot poses. Only the corresponding block of the covariance matrix is needed to evaluate the product in Equation (4.21). The rank of a matrix is also, at most, the number of non-zero rows (or columns) of the information matrix. The resulting complexity of evaluating the complexity in Equation (4.21) accounting for sparsity is $O(|X_i|^3)$, where X_i is the number of co-visible poses. If the number of co-visible poses is bounded, this results in a constant asymptotic computational complexity for evaluating the marginal gain. The asymptotic computational complexity of maximising this implementation is $O(|X_i|^3n^2)$ and $O(|X_i|^3n)$ for the lazy and stochastic greedy algorithms, respectively.

However, the matrix determinant lemma in Equation (4.21) requires calculating the covariance matrix over robot poses for the selected set of map points, or $(\mathbf{\Lambda}^S)^{-1}$. This matrix changes after every iteration of the greedy algorithm. A naive approach would be to invert the current information matrix after each map point selected by the greedy algorithm, which has an asymptotic complexity of $O(t^3)$ per selected map point. An alternative to using the Sherman-Morrison-Woodbury [58] formula allows updating the covariance matrix after low-rank updates, or $O(t^2)$. Updating the covariance matrix after

every selected map point requires $O(n)$ updates and has an asymptotic computational complexity of $O(nt^2)$.

This implementation, in contrast to other approaches in this dissertation, has the unique property that evaluating the marginal gain is no longer the dominating cost of the computational complexity for the approach. The computational complexity is instead determined by the cost of updating the covariance matrix (if we reasonably assume the number of map points grows with the length of the trajectory). From an asymptotic worst-case computational complexity perspective, this approach is an improvement over the previous approach, reducing the complexity for both the lazy greedy algorithm and the stochastic greedy algorithm of $O(nt^3)$ and $O(n^2t^3)$, respectively, to $O(nt^2)$. This is particularly significant when using the lazy-greedy algorithm with the SLAM utility. However, in practice, the lazy-greedy algorithm is far less expensive than predicted by its worst-case asymptotic computational complexity. This approach based on the determinant lemma also requires storing the dense covariance matrix, which can require a significant amount of memory for larger trajectories.

This subsection presented an alternative implementation of the SLAM utility using the matrix determinant lemma. While this approach offers very competitive asymptotic computational complexity, especially when using the lazy greedy algorithm, this approach requires storing and operating on the dense covariance matrix.

4.1.3. SLAM-based Utility Implementation using Cholesky Updates

Subsection 4.1.1 proposes a utility function based on the information gain of the SLAM posterior and a naive implementation that has a computational complexity of $O(t^3)$ for evaluating the marginal gain. Subsection 4.1.2 proposes an alternative implementation, which requires calculating the dense covariance matrix. This subsection instead presents an alternative approach that, rather than calculating the Cholesky decomposition from scratch, as shown in 4.1.1, updates the Cholesky decomposition. This approach reduces the asymptotic complexity of evaluating the marginal gain and uses sparse techniques.

The Cholesky update [59] is a common linear algebra technique that is an efficient way to calculate the Cholesky decomposition of a matrix after the original matrix has been modified by adding a low-rank matrix. For a given Cholesky decomposition $\mathbf{L}\mathbf{L}^{\mathrm{T}}$, the Cholesky update allows efficiently calculating the decomposition after the matrix has been subjected to the low-rank update $\mathbf{A}_i\mathbf{A}_i^{\mathrm{T}}$, or

$$\tilde{\mathbf{L}}\tilde{\mathbf{L}}^{\mathrm{T}} = \mathbf{L}\mathbf{L}^{\mathrm{T}} + \mathbf{A}_i\mathbf{A}_i^{\mathrm{T}}, \tag{4.22}$$

where $\mathbf{A}_i$ is a $6t \times r$ matrix, and $\tilde{\mathbf{L}}$ is the Cholesky decomposition of the sum of the matrices in Equation (4.22).

The posterior information matrix after adding a map point with index i to the currently

selected set of map points S is given by the sum of the information matrix for the map point $\mathbf{\Lambda}^i_{\mathbf{x}_{1:t}}$ and the information matrix of the selected set $\mathbf{\Lambda}^S_{\mathbf{x}_{1:t}}$, or $\mathbf{\Lambda}^{S\cup\{m_i\}}_{\mathbf{x}_{1:t}} = \mathbf{\Lambda}^S_{\mathbf{x}_{1:t}} + \mathbf{\Lambda}^i_{\mathbf{x}_{1:t}}$. An important observation is that map points are typically only visible from a limited number of poses. The information matrix associated with a single map point with index i, or $\mathbf{\Lambda}^i_{\mathbf{x}_{1:t}}$, is mostly sparse and is only non-zero at the entries that correspond to the poses from which a map point was observed. The rank of a matrix is, at most, the number of non-zero rows or columns in a matrix. A bounded number of map point observations imply that $\mathbf{\Lambda}^i_{\mathbf{x}_{1:t}}$ is low rank. Similar to using the matrix determinant lemma from Subsection 4.1.2, it is necessary to decompose the information matrix $\mathbf{\Lambda}^i_{\mathbf{x}_{1:t}}$ into a low-rank update matrix $\mathbf{A}_i\mathbf{A}_i^{\mathrm{T}}$, which is the decomposition of the information matrix that corresponds with the map point with index i, or

$$\mathbf{\Lambda}^{S\cup\{m_i\}}_{\mathbf{x}_{1:t}} = \mathbf{\Lambda}^S_{\mathbf{x}_{1:t}} + \mathbf{A}_i\mathbf{A}_i^{\mathrm{T}}. \tag{4.23}$$

Let $\tilde{\mathbf{L}}$ be the Cholesky decomposition of $\mathbf{\Lambda}^{\{m_i\}\cup S}_{\mathbf{x}_{1:t}}$ and $\mathbf{L}$ be the decomposition of $\mathbf{\Lambda}^S_{\mathbf{x}_{1:t}}$. Using the Cholesky update routine, $\tilde{\mathbf{L}}$ can be calculated using the update $\mathbf{A}_i$ and $\mathbf{L}$, or

$$\tilde{\mathbf{L}} = \mathrm{CholeskyUpdate}(\mathbf{L}, \mathbf{A}_i). \tag{4.24}$$

This operation has an asymptotic complexity of $O(rt^2)$ instead of $O(t^3)$ for calculating the Cholesky decomposition from scratch [60]. If we assume the number of poses from which a map point is visible is bounded, the rank of the update is bounded and this simplifies to $O(t^2)$. Updating the Cholesky decomposition reduces the worst-case asymptotic complexity of optimising the SLAM utility with the stochastic greedy algorithm and lazy greedy algorithms to $O(nt^2)$ and $O(n^2t^2)$, respectively.

We can compare this asymptotic complexity of the greedy selection to the complexity of performing SLAM estimation. In practice, a limited number of new map points are observed at each timestep; therefore, the total number of map points is at most proportional to the length of the trajectory, or $n = O(t)$. The resulting worst-case complexity is $O(t^3)$. This is noteworthy since it is equivalent to the worst-case complexity of SLAM estimation using bundle adjustment, which implies similar asymptotic scaling. However, it is important to note that SLAM estimation is already challenging to perform online for large-scale problems, and there is typically an order of magnitude more map points than poses in the trajectory. Therefore, we expect map point selection using the SLAM utility to be very computationally expensive, even when updating the Cholesky decomposition and likely not suitable for selecting map points online in large-scale problems.

When implementing the SLAM utility and using Cholesky updates to speed up the calculation of the marginal gain, we store a copy of the Cholesky decomposition of the currently selected set of map points $\mathbf{L}$. Whenever a greedy algorithm selects a map point,

the currently stored Cholesky decomposition is also updated, or

$$\mathbf{L} \leftarrow \text{CholeskyUpdate}(\mathbf{L}, \mathbf{A}_i). \tag{4.25}$$

To calculate the marginal gain of a map point for a greedy algorithm, we calculate the updated Cholesky decomposition of the posterior matrix $\tilde{\mathbf{L}}$ for map point m_i using the Cholesky update Equation (4.24) and evaluate

$$f(m_i|S) = \log(|\tilde{\mathbf{L}}|) - \log(|\mathbf{L}|). \tag{4.26}$$

These operations require, in the worst case, memory quadratic with the length of the trajectory $O(t^2)$.

The marginal information matrix over poses $\mathbf{\Lambda}_{\mathbf{x}_{1:t}}$ is sparse for many SLAM trajectories. If an efficient elimination ordering can be found, its Cholesky decomposition, or $\mathbf{L}$, is also sparse. As a result, we recommend using implementations of the Cholesky update designed to work on sparse matrices, as these approaches can exploit the problem structure and are often significantly less expensive in practice than suggested by the worst-case asymptotic complexity. This dissertation uses a general-purpose implementation of the Cholesky updates (CHOLMOD [61]). This approach of updating the Cholesky decomposition is similar to the approach used for incremental non-linear least squares solvers developed for the SLAM estimation problem, where the decomposition of the posterior information matrix is updated between time steps (see ISAM 2 [62] or SLAM++ [46]).

This subsection presents our approach of updating the Cholesky decomposition to improve the asymptotic complexity of evaluating the marginal gain from the previous naive approach from Subsection 4.1.1 and reduces the computational complexity from $O(t^3)$ to $O(t^2)$. This approach also avoids storing the dense covariance matrix required for the approach from Subsection 4.1.2, while having a similar worst-case computational complexity when this utility implementation is maximised using the stochastic greedy algorithm of $O(nt^2)$.

4.2. Utilities based on Information-Theoretic Approximations

Section 4.1 proposed a utility based on the SLAM posterior, but this baseline utility is computationally expensive to evaluate due to the high asymptotic complexity. This section applies a simplifying approximation to the SLAM posterior to obtain approaches with reduced asymptotic computational complexity.

This section investigates two approximations: The first is an approximation based on approximating the posterior with a series of localisation problems. The second is

an approximation based on a series of odometry problems. These approximations are presented in Subsections 4.2.1 and 4.2.2, respectively.

4.2.1. Utility based on a Localisation Approximation

Section 4.1 proposed a utility based on the SLAM posterior, but this baseline utility is expensive to evaluate. This subsection proposes a novel utility that improves the scalability of the baseline SLAM approach by approximating the SLAM posterior with a set of independent localisation problems. This approximation results from the assumption that perfect knowledge of the map points has a negligible effect on the SLAM posterior distribution, or

$$P(\mathbf{x}_{1:t}|\mathbf{z}^S_{1:t}) \approx P(\mathbf{x}_{1:t}|\mathbf{z}^S_{1:t}, \mathbf{m}_S). \tag{4.27}$$

This approximation is reasonable if there is very little uncertainty in the map point positions. We expect this simplifying assumption will overvalue the utility of map points with a limited number of observations or poor viewpoint coverage. The impact of this approximation when using this utility with greedy algorithms will depend on the utility of the resulting greedy selection of map points. If using the approximate utility function still accurately ranks the relative utility of map points, the greedy algorithm will continue to select sets with high utility despite using an approximate utility function.

For the development of the localisation utility, we use the assumption in Equation (4.27) and the assumption from the development of the SLAM utility that the prior distribution over the robot trajectory is independent and identically distributed, as shown in Equation (4.9), to simplify the calculation of the information gain. These two assumptions allow factorising the posterior distribution into the product of independent factors associated with each pose, or

$$P(\mathbf{x}_{1:t}|\mathbf{z}^S_{1:t}, \mathbf{m}_S) = \eta \prod_{j=1}^{t} P(\mathbf{x}_j) P(\mathbf{z}^S_j|\mathbf{x}_j, \mathbf{m}_S), \tag{4.28}$$

where η is a normalising constant.

The approximate information matrices can be calculated directly from the factors of the posterior distribution, as shown in Subsection 3.1.2 and used in Subsection 4.1 for the calculation of the SLAM utility. Each factor in the product in Equation (4.28) only involves one pose and is independent of other poses due to the aforementioned assumptions. This simplifies the calculation of the SLAM utility in Equation (4.4) to the sum of terms originating from independent localisation problems, or

$$f_{\text{local}}(S) = \sum_{j=1}^{t} \log(|\mathbf{\Lambda}^S_{\mathbf{x}_j|\mathbf{m}}|) - c_\emptyset, \tag{4.29}$$

where $\mathbf{\Lambda}^S_{\mathbf{x}_j|\mathbf{m}}$ is the information matrix for $\mathbf{x}_j$ due to the localisation approximation. We define $\mathbf{J}^X_{ij}$ as the Jacobian of the measurement function with respect to the robot pose.

The information matrix $\mathbf{\Lambda}^S_{\mathbf{x}_j|\mathbf{m}}$ can be calculated in terms of the selected map point measurements with measurement Jacobian $\mathbf{J}^X_{ij}$ and measurement noise $\mathbf{\Omega}_{ij}$, or

$$\mathbf{\Lambda}^S_{\mathbf{x}_j|\mathbf{m}} = \epsilon\mathbf{I} + \sum_{i\in S}(\mathbf{J}^X_{ij})^{\mathrm{T}}\mathbf{\Omega}(\mathbf{J}^X_{ij}). \tag{4.30}$$

We now consider evaluating the marginal gain for the localisation approximation. We define X_i as the set of poses from which map point $\mathbf{m}_i$ is visible. By substituting the localisation utility in Equation (4.29) into the marginal gain definition in Equation (3.22), the marginal gain simplifies to only the terms related to poses from which $\mathbf{m}_i$ was visible, or

$$f_{\text{local}}(m_i|S) = \sum_{j\in X_i} \log(|\mathbf{\Lambda}^{S\cup\{m_i\}}_{\mathbf{x}_j|\mathbf{m}}|) - \log(|\mathbf{\Lambda}^S_{\mathbf{x}_j|\mathbf{m}}|). \tag{4.31}$$

Note that the information matrices of each localisation sub-problem are of constant size and are 6×6 matrices. While evaluating the localisation utility in Equation (4.29) takes $O(t)$ time, evaluating the marginal gain in Equation (4.31) takes only $O(|X_i|)$ time, as only the co-visible localisation problems contribute to the marginal gain. This reduces to $O(1)$ if we assume the number of poses from which map points are visible, or $|X_i|$, is bounded. This is significantly less expensive than the $O(t^2)$ complexity of evaluating the marginal gain for the full SLAM utility. The computational complexity of the function evaluations for the lazy greedy and stochastic greedy algorithms are $O(n^2|X_i|)$ and $O(n|X_i|)$, respectively.

This utility function is implemented as follows: The current information matrix of each frame, or $\mathbf{\Lambda}^S_{\mathbf{x}_j|m}$ is stored and initialised with the prior information matrix, or

$$\mathbf{\Lambda}^{\emptyset}_{\mathbf{x}_j} = \epsilon\mathbf{I}. \tag{4.32}$$

Whenever a greedy algorithm selects a map point, the stored matrices for each frame are also updated, or

$$\mathbf{\Lambda}^{S\cup\{m_i\}}_{\mathbf{x}_j} = \mathbf{\Lambda}^S_{\mathbf{x}_j} + (\mathbf{J}^X_{ij})^{\mathrm{T}}\mathbf{\Omega}(\mathbf{J}^X_{ij}). \tag{4.33}$$

Storing the 6×6 information matrix for each frame takes $O(t)$ memory. Each matrix of 36 elements requires 288 bytes if stored as doubles of 8 bytes each. For a trajectory of 1500 frames, this would require 432 kB. If we exploit that information matrices are symmetric and only store the lower triangular of the matrices, this reduces to 252 kB. This is negligible compared to storing the map or the memory required by SLAM algorithms. This utility function is not only computationally inexpensive, but also requires limited memory. It is possible to speed up the evaluation of the marginal gain in Equation (4.31) by storing the intermediate information matrices $(\mathbf{J}^X_{ij})^{\mathrm{T}}\mathbf{\Omega}(\mathbf{J}^X_{ij})$ associated with each map point observation. However, storing these intermediate matrices requires $O(n|X_i|)$ memory and is not recommended for memory-constrained applications.

This subsection presents a utility function that approximates the SLAM posterior based

on a series of localisation problems to obtain a computationally inexpensive alternative. The calculation of this utility is also applicable for memory-constrained applications.

4.2.2. Utility based on a Stereo Odometry Approximation

Subsection 4.2.1 proposed a utility based on approximating the SLAM posterior over robot poses given the selected map points with a set of localisation problems. This subsection instead proposes a novel approximation of the posterior distribution over robot poses with a set of independent stereo odometry problems. For these stereo odometry problems, map point measurements are converted into a dead reckoning measurement of the relative motion between a pair of frames.

The entropy of the posterior distribution over poses can be expanded using the chain rule. This general property allows expanding the joint posterior distribution into a sum of entropy terms related to each pose conditioned on all observations and previous robot poses in the chain, or

$$H(\mathbf{x}_{1:t}|\mathbf{z}^S_{1:t}) = H(\mathbf{x}_1|\mathbf{z}^S_{1:t}) + H(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}^S_{1:t}) + \cdots + H(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}^S_{1:t}). \tag{4.34}$$

The odometry utility uses the general property of entropy that conditioning on fewer variables always increases the entropy. For the odometry utility, each term within Equation (4.34) is approximated by conditioning only on the variables involved in a pair of timesteps. Note that we cannot necessarily use consecutive poses since keyframes are only inserted when exploring novel regions and consecutive poses might not share any map points. Instead of using consecutive poses, this work uses the simple heuristic of pairing the pose $\mathbf{x}_j$ with the frame with the highest number of co-visible map points $\mathbf{x}_{p_j}$ with a lower index, or $p_j < j$. Despite the simplicity of this approach, we found it to work well in practice. The entropy of each pose term in the chain is approximated by conditioning on the pair of poses and the associated observations from those poses,

$$H(\mathbf{x}_j|\mathbf{x}_{1:j-1}, \mathbf{z}^S_{1:t-1}) \approx H(\mathbf{x}_j|\mathbf{x}_{p_j}, \mathbf{z}^S_{p_j}, \mathbf{z}^S_j). \tag{4.35}$$

Substituting Equation (4.35) into the sum in Equation (4.34) yields a bound on the posterior entropy using the sum of pose pairs, or

$$H(\mathbf{x}_{1:t}|\mathbf{z}^S_{1:t}) \approx H(\mathbf{x}_1) + \sum_{j=2}^{t} H(\mathbf{x}_j|\mathbf{x}_{p_j}, \mathbf{z}^S_{p_j}, \mathbf{z}^S_j). \tag{4.36}$$

The information gain using the SLAM posterior is the difference between $H(\mathbf{x}_{1:t})$ and $H(\mathbf{x}_{1:t}|\mathbf{z}^S_{1:t})$. Substituting Equation (4.36) into this equation yields the map point utility

based on visual odometry, or

$$f_{\text{odometry}}(S) = c_{\emptyset} + \sum_{j=2}^{t} \log(|\mathbf{\Lambda}^{S}_{\mathbf{x}_j|\mathbf{x}_{p_j}}|), \tag{4.37}$$

where $\mathbf{\Lambda}^{S}_{\mathbf{x}_j|\mathbf{x}_{p_j}}$ is the information matrix of the conditional distribution over $\mathbf{x}_j$, conditioned on the paired pose $\mathbf{x}_{p_j}$ and the measurements between those poses. This odometry utility formulation is a lower bound to the information gain from map point measurements using the SLAM posterior. It is also worth noting that this approach implicitly only evaluates stereo odometry information by only considering a pair of poses. As with the localisation approximation from Subsection 4.2.1, the impact of this approximation depends on how it affects the selection of the greedy algorithm. By only considering the stereo odometry information, this utility undervalues map points observed from multiple frames.

The information matrix of each pose j is given by the selected stereo odometry measurements and the prior, or

$$\mathbf{\Lambda}^{S}_{\mathbf{x}_j|\mathbf{x}_{p_j}} = \epsilon\mathbf{I} + \sum_{i \in S} \mathbf{\Lambda}^{i}_{\mathbf{x}_j|\mathbf{x}_{p_j}}. \tag{4.38}$$

We now show how to calculate the stereo odometry matrices $\mathbf{\Lambda}^{i}_{\mathbf{x}_j|\mathbf{x}_{p_j}}$. For a map point i that is visible at timestep j, we search for the corresponding measurement of i in the parent pose p_j. If the map point is not visible from both pose j and pose p_j, the information matrix is $\mathbf{\Lambda}^{i}_{\mathbf{x}_j|\mathbf{x}_{p_j}} = \mathbf{0}$. Only the odometry information; that is, measurements between the parent pose p_j and j, are used in this approximation. If a matching observation of map point i is found in p_j the information matrix is calculated. To calculate the odometry information matrix, we first calculate the joint information matrix of the pose $\mathbf{x}_j$, its parent $\mathbf{x}_{p_j}$ and the map point $\mathbf{m}_i$ for these two map point measurements, or

$$\mathbf{\Lambda}_{\mathbf{x}_j,\mathbf{x}_{p_j},\mathbf{m}_i} = \mathbf{J}^{\mathrm{T}}_{ij}\mathbf{\Omega}\mathbf{J}_{ij} + \mathbf{J}^{\mathrm{T}}_{ip_j}\mathbf{\Omega}\mathbf{J}_{ip_j}. \tag{4.39}$$

For notational simplicity, we define the block matrices of the Jacobians as follows: we define $\mathbf{A}$ as the block of the measurement that corresponds with the robot pose j and $\mathbf{B}$ at the block that corresponds with the map point position, or

$$\mathbf{J}_{ij} = \begin{pmatrix} \mathbf{A} & \mathbf{0} & \mathbf{B} \end{pmatrix}. \tag{4.40}$$

Similarly, we define the blocks of the Jacobian of the measurement of the map point with index i at the parent pose as $\mathbf{C}$ for the block that corresponds with the robot pose $\mathbf{x}_{p_j}$ and $\mathbf{D}$ as the block that corresponds to the map point $\mathbf{m}_i$, or

$$\mathbf{J}_{ip_j} = \begin{pmatrix} \mathbf{0} & \mathbf{C} & \mathbf{D} \end{pmatrix}. \tag{4.41}$$

The joint information matrix of these two measurements is given by the sum of the two information matrices, or

$$\begin{aligned}\boldsymbol{\Lambda}_{\mathbf{x}_j,\mathbf{x}_{p_j},\mathbf{m}_i} &= \mathbf{J}_{ij}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{J}_{ij} + \mathbf{J}_{ip_j}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{J}_{ip_j} \\ &= \begin{pmatrix} \mathbf{A}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{A} & \mathbf{0} & \mathbf{A}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{B} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{B}\boldsymbol{\Omega}\mathbf{A}^{\mathrm{T}} & \mathbf{0} & \mathbf{B}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{B} \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{C} & \mathbf{C}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D} \\ \mathbf{0} & \mathbf{D}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{C} & \mathbf{D}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{A}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{A} & \mathbf{0} & \mathbf{A}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{B} \\ \mathbf{0} & \mathbf{C}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{C} & \mathbf{C}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D} \\ \mathbf{B}\boldsymbol{\Omega}\mathbf{A}^{\mathrm{T}} & \mathbf{D}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{C} & \mathbf{B}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{B} + \mathbf{D}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D} \end{pmatrix}. \end{aligned} \tag{4.42}$$

The marginal information matrix over only the two poses, $\mathbf{x}_j$ and $\mathbf{x}_{p_j}$, is given by the Schur's complement of the map point states, or

$$\begin{aligned}\boldsymbol{\Lambda}^i_{\mathbf{x}_j,\mathbf{x}_{p_j}} &= \begin{pmatrix} \mathbf{A}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{C} \end{pmatrix} \\ &\quad - \begin{pmatrix} \mathbf{A}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{B} \\ \mathbf{C}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D} \end{pmatrix} (\mathbf{B}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{B} + \mathbf{D}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D})^{-1} \begin{pmatrix} \mathbf{B}\boldsymbol{\Omega}\mathbf{A}^{\mathrm{T}} & \mathbf{D}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{C} \end{pmatrix}. \end{aligned} \tag{4.43}$$

The information matrix associated with the posterior distribution over the pose $\mathbf{x}_j$ conditioned on $\mathbf{x}_{p_j}$ requires only the associated sub-block of the information matrix. The conditional information matrix of the map point measurement i using the odometry approximation, or $\boldsymbol{\Lambda}^i_{\mathbf{x}_j|\mathbf{x}_{p_j}}$ calculated from $\boldsymbol{\Lambda}^i_{\mathbf{x}_j,\mathbf{x}_{p_j}}$ as

$$\boldsymbol{\Lambda}^i_{\mathbf{x}_j|\mathbf{x}_{p_j}} = \mathbf{A}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{A} - (\mathbf{A}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{B})(\mathbf{B}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{B} + \mathbf{D}^{\mathrm{T}}\boldsymbol{\Omega}\mathbf{D})^{-1}(\mathbf{B}\boldsymbol{\Omega}\mathbf{A}^{\mathrm{T}}), \tag{4.44}$$

where $\mathbf{A}, \mathbf{B}$ and $\mathbf{D}$ correspond to the blocks of the measurement Jacobians as defined in Equation (4.40) and Equation (4.41). The block $\mathbf{C}$ is not present in this equation. Rather than explicitly forming the joint information matrix, the odometry approximation can be directly calculated from Equation (4.44) using the corresponding measurement Jacobians and noise matrices. This concludes the development of the odometry utility. The posterior information matrix of a pose in the trajectory in Equation (4.38) is given by the sum of the selected map points and their associated stereo odometry information matrices as given by Equation (4.44). The utility for the full trajectory is given by the log-determinant of each pose information matrix, as shown in Equation (4.38).

The greedy algorithms from Subsection 3.3 require calculating the marginal gain at each iteration. The marginal gain of odometry utility for a map point m_i simplifies to only

the limited set of poses X_i where the map point m_i was visible, or

$$f_{\text{odometry}}(m_i|S) = \sum_{j \in X_i} \log(|\mathbf{\Lambda}^{S\cup\{m_i\}}_{\mathbf{x}_j|\mathbf{x}_{p_j}}|) - \log(|\mathbf{\Lambda}^{S}_{\mathbf{x}_j|\mathbf{x}_{p_j}}|). \tag{4.45}$$

Evaluating the marginal gain for the odometry utility has a computational complexity of $O(|X_i|)$. The number of poses from which map points are visible is typically bounded in SLAM datasets and we expect it to be constant in practice. Maximising the odometry utility with the lazy and stochastic greedy algorithms has a complexity of $O(n^2|X_i|)$ and $O(n|X_i|)$, respectively.

The odometry utility is similar to the localisation utility from Subsection 4.2.1 in that a sum is defined over the information matrices of each pose. These two utilities are identical from a computational complexity and memory usage perspective. The only difference to the localisation utility is in the calculation of the measurement information matrices. The odometry utility is another approximate alternative to the SLAM utility from Subsection 4.1.1.

4.3. Utility based on Coverage and Associated Approaches

Previously, Sections 4.1 and 4.2 proposed information-theoretic utility functions aimed at the utility of map points for the back-end. Section 2.2 reviewed coverage-based approaches that have been used for selecting map points for localisation and visual odometry [16; 30; 31; 33; 34]. These coverage-based approaches instead select map points based on the number of map points visible in the respective camera frames. These approaches are motivated by the necessity of localisation algorithms to have sufficient reliable features for feature matching. Therefore, these coverage-based approaches are aimed at capturing the utility of map points for front-end algorithms. This dissertation investigates the use of coverage-based approaches for the map point selection problem for SLAM.

This section presents an existing coverage-based approach by Dymczyk *et al.* [34]. This dissertation evaluates its use for the map point selection problem for SLAM. This existing coverage-based approach is detailed in Subsection 4.3.1, along with a brief overview of the approaches it is based on. A discussion of these alternative approaches provides insight into the relevant parameters. While the formulation from Subsection 4.3.1 is a minimisation problem, Subsection 4.3.2 shows that this problem can be adapted into an equivalent submodular maximisation problem where the utility is based on the map points visible in each frame. This adaptation allows the use of greedy algorithms to maximise the utility at reduced asymptotic computational complexity compared to the previously used integer programming techniques. Lastly, Subsection 4.3.3 presents an alternative novel coverage-based approach that maximises the minimum number of map points visible in a

camera frame.

4.3.1. Existing Weighted Coverage Integer Program for Map Point Selection

This subsection presents an existing coverage-based approach for selecting map points for visual localisation and odometry in robotics applications [34]. This dissertation evaluates this approach for selecting map points for visual SLAM. Before this subsection presents the formulation from Dymczyk *et al.* [34], the subsection first provides a brief overview of the formulations this coverage model is based on, as it is beneficial for explaining the relevant parameters.

Several existing approaches formulate map point selection for localisation as minimisation problems. These approaches minimise the number of selected map points or the weighted cost of map points while enforcing the constraint that each camera frame contains at least a desired selected number of map points, or b_{cover}. Using linear integer programming optimisation techniques to solve these problems is common, which formulates the resulting optimisation problem using integer variables and linear objectives and constraints. We define a minimisation problem over the column vector $\mathbf{s}$, which encodes the selected set of map points. The algorithm adjusts these selected map points encoded by this vector to minimise the objective value. Each element in this column vector $\mathbf{s}$, or s_i, is either 1 to indicate that the map point is selected or 0 to indicate that it was not selected. We also define a $t \times n$ matrix $\mathbf{A}$ where the entry at row j and column i, or a_{ji}, encodes whether the map point with index i is visible in the frame with index j. The selected map points visible in a frame are, therefore, given by the product $\mathbf{As}$. The resulting set multi-cover problem is the minimisation problem where the minimum number of map points are selected so that each camera frame contains at least a given coverage value, or b_{cover}, which is formulated by the equations

$$\begin{aligned} \arg\min_{\mathbf{s}} & \sum_{i=1}^{n} s_i \\ & \mathbf{As} \geq b_{\text{cover}}\mathbf{1} \\ & \mathbf{s} \in \{0,1\}^n, \end{aligned} \tag{4.46}$$

where $\mathbf{1}$ is a unit column vector.

Park *et al.* [33] proposes to bias map point selection towards more stable higher quality map points by assigning different costs to the selected map points. This variation introduces a weight vector $\mathbf{q}$ where each entry q_i is the cost of the associated map point. Rather than minimising the number of map points, this formulation minimises the weighted cost of the

selected map points while satisfying the coverage constraint, or

$$\begin{aligned} \arg\min_{\mathbf{s}} \mathbf{q}^{\mathrm{T}}\mathbf{s} \\ \mathbf{A}\mathbf{s} \geq b_{\text{cover}}\mathbf{1} \\ \mathbf{s} \in \{0,1\}^n. \end{aligned} \tag{4.47}$$

This weighted variation allows assigning lower costs to higher quality map points to bias selection towards higher quality map points. The formulations in Equation (4.46) and Equation (4.47) are generalisations of the set cover problem and the weighted set cover problem, respectively, where $b_{\text{cover}} = 1$ [63].

A notable disadvantage of these set multi-cover formulations is that these approaches do not allow the specification of the total number of selected map points. Dymczyk *et al.* [34] suggests modifying the formulation by considering the coverage constraint $\mathbf{A}\mathbf{x} \geq b_{\text{cover}}\mathbf{1}$ as a soft constraint. Unlike hard constraints, where the solution is not valid if the constraint is not satisfied, soft constraints only penalise the objective function. Modelling the problem with a soft constraint introduces the slack variable $\boldsymbol{\zeta}$ that penalises select sets of map points that do not satisfy the coverage constraint. This slack variable can be interpreted as the number of map points missing from the map for each frame to contain b_{cover} selected map points. This modification allows the introduction of a new constraint on the selected number of map points k, or

$$\begin{aligned} \arg\min_{\mathbf{s},\boldsymbol{\zeta}} \mathbf{q}^{\mathrm{T}}\mathbf{s} + \lambda\mathbf{1}^{\mathrm{T}}\boldsymbol{\zeta} \\ \mathbf{A}\mathbf{s} + \boldsymbol{\zeta} \geq b_{\text{cover}}\mathbf{1} \\ \sum_{i=1}^{n} s_i = k \\ \mathbf{s} \in \{0,1\}^n \\ \boldsymbol{\zeta} \in \{\mathbb{N}\}^t, \end{aligned} \tag{4.48}$$

where λ is a design variable that controls the trade-off between the degree to which the coverage constraint is satisfied and the weighted cost $\mathbf{q}^{\mathrm{T}}\mathbf{s}$. Each map point missing from a camera frame penalises the solution by λ. Dymczyk *et al.* [34] suggests assigning each term in $\mathbf{q}$ or q_i to be the difference between the highest number of frames any map point is visible from and the frames the map point with index i is visible from, or

$$q_i = \left(\max_{k \in V} |X_k|\right) - |X_i|, \tag{4.49}$$

where X_i is the set of frames from which map point i is visible. This formulation assigns a lower cost to map points visible from a larger number of frames.

This optimisation problem in Equation (4.48) can be solved exactly by using integer

programming techniques. Similar to Dymczyk *et al.* [34], we use the GUROBI [64] toolbox, which uses a branch-and-cut algorithm along with various heuristics. While this approach has a poor theoretical worst-case asymptotic complexity for small to moderately-sized problems, integer programming techniques can be very efficient and have low execution times. Unlike the greedy counterparts used in this dissertation, this integer programming technique finds the optimal solutions.

4.3.2. Utility based on Greedy Weighted Coverage

This subsection adapts the formulation by Dymczyk *et al.* [34] in Equation (4.47) into an equivalent submodular maximisation problem subject to a cardinality constraint. This adaptation allows this dissertation to solve an equivalent problem using greedy algorithms[1]. Using greedy algorithms improves the asymptotic computational complexity compared to using integer programming techniques.

Unlike when using linear integer programming techniques where the objective function to be minimised is required to be linear, the submodular maximisation algorithms can consider non-linear utility functions to maximise. Before this subsection adapts the optimisation problem from Equation (4.48) into a compatible maximisation problem, it is first necessary to rewrite the equations only in terms of $\mathbf{s}$ and avoid the slack variable $\boldsymbol{\zeta}$ associated with the soft constraint. Rewriting the equation from Equation (4.48) such that $\boldsymbol{\zeta}$ is the subject of the equation yields

$$\boldsymbol{\zeta} \geq b_{\text{cover}}\mathbf{1} - \mathbf{A}\mathbf{s}. \tag{4.50}$$

Additionally, since the elements of $\boldsymbol{\zeta}$ are limited to natural numbers, it implies the constraint

$$\boldsymbol{\zeta} \geq 0. \tag{4.51}$$

The objective function in Equation (4.48) is minimised when $\boldsymbol{\zeta}$ is minimised. Suppose a camera frame j has less than b_{cover} map points. In that case, the corresponding slack variable element ζ_j equals the difference between the b_{cover} and the selected number of map points in the frame. Alternatively, suppose a camera with index j contains more than b_{cover} map points. In that case, the corresponding element of the slack variable will be zero. These two observations imply that the minimum value of $\boldsymbol{\zeta}$ is equivalently given by the equation

$$\boldsymbol{\zeta} = b_{\text{cover}}\mathbf{1} - \min(\mathbf{A}\mathbf{s}, b_{\text{cover}}\mathbf{1}), \tag{4.52}$$

where min calculates the element-wise minimum between the two column vector inputs. As noted in Section 4.3.1, this slack variable is the number of missing map points. By

[1] Our greedy optimisation of this objective function is separate from the greedy heuristic from Lynen *et al.* [17] compared in Dymczyk *et al.* [34] that uses a different utility function.

substituting Equation (4.52) into the optimisation problem, the resulting optimisation problem can be formulated only in terms of $\mathbf{s}$.

Minimising an objective $-f(\mathbf{s})$ is equivalent to maximising $f(\mathbf{s})$. As a result, Equation (4.48) can be rewritten as a maximisation problem as

$$\begin{aligned} &\arg\max_{\mathbf{s}} f(\mathbf{s}) \\ &\sum_{i=1}^{n} s_i = k. \end{aligned} \tag{4.53}$$

$f(\mathbf{s})$ is the negative of the objective function from Equation (4.48) with Equation (4.52) substituted to replace $\boldsymbol{\zeta}$, or

$$f(\mathbf{s}) = -\mathbf{q}^{\mathrm{T}}\mathbf{s} - \lambda \mathbf{1}^{\mathrm{T}}(b_{\text{cover}}\mathbf{1} - \min(\mathbf{A}\mathbf{s}, b_{\text{cover}}\mathbf{1})). \tag{4.54}$$

For the derivation in this subsection, it is beneficial to rewrite the weight vector of map points, or $\mathbf{q}$ from Equation (4.49), in terms of matrix equations. As defined earlier in Subsection 4.3.1, each element in $\mathbf{q}$ is chosen to be equal to the difference between the maximum number of map point observations of any map point and observations of the corresponding map point at that index, or $q_i = \left(\max_{k \in V} |X_k|\right) - |X_i|$. For notational simplicity, we define $x_{\max}$ as the maximum number of map point observations of any map point; that is, $x_{\max} = \max_{k \in V} |X_k|$. Additionally, the number of camera poses that include a map point can equivalently be written using the matrix $\mathbf{A}$, where each entry at a_{ji} encodes whether a map point at column i is visible in the frame corresponding to row j. These substitutions into Equation (4.49) yields

$$\mathbf{q} = x_{\max}\mathbf{1} - \mathbf{A}^{\mathrm{T}}\mathbf{1}. \tag{4.55}$$

If we substitute Equation (4.55) into the objective in Equation (4.54), it yields the equation

$$f(\mathbf{s}) = -(x_{\max}\mathbf{1} - \mathbf{A}^{\mathrm{T}}\mathbf{1})^{\mathrm{T}}\mathbf{s} - \lambda \mathbf{1}^{\mathrm{T}}(b_{\text{cover}}\mathbf{1} - \min(\mathbf{A}\mathbf{s}, b_{\text{cover}}\mathbf{1})). \tag{4.56}$$

To further simplify Equation (4.58), we note that the sum of the selected map points is constrained to the map point budget, or

$$\mathbf{1}^{\mathrm{T}}\mathbf{s} = k. \tag{4.57}$$

Substituting Equation (4.57) into Equation (4.57) and gathering all the terms yields the equation

$$f(\mathbf{s}) = \mathbf{1}^{\mathrm{T}}\mathbf{A}\mathbf{s} + \lambda \mathbf{1}^{\mathrm{T}} \min(\mathbf{A}\mathbf{s}, b_{\text{cover}}\mathbf{1}) + (-x_{\max}k + b_{\text{cover}}\lambda t). \tag{4.58}$$

All the terms in the second bracket are constants, and therefore, Equation (4.58) can

equivalently be written as

$$f(\mathbf{s}) = \mathbf{1}^{\mathrm{T}}\mathbf{A}\mathbf{s} + \lambda\mathbf{1}^{\mathrm{T}}\min(\mathbf{A}\mathbf{s}, b_{\mathrm{cover}}\mathbf{1}) + \text{const.} \tag{4.59}$$

For greedy algorithms, the objective is normalised, that is $f(\emptyset) = 0$ from Equation (3.25), and we drop the constant term from Equation (4.60), or

$$f_{\mathrm{wcover}}(\mathbf{s}) = \mathbf{1}^{\mathrm{T}}\mathbf{A}\mathbf{s} + \lambda\mathbf{1}^{\mathrm{T}}\min(\mathbf{A}\mathbf{s}, b_{\mathrm{cover}}\mathbf{1}). \tag{4.60}$$

The first term in the utility function in Equation (4.60) is the sum of the selected map points visible in each camera frame. The second term is the bounded sum of the map points visible in each frame where the map point observations are counted up to a maximum of b_{cover}, after which further map points do not contribute to the second term. This second term represents the coverage of the selected map points. The parameter λ controls the trade-off between the first and second terms.

This dissertation mainly uses set notation when formulating greedy utility functions. To comply with this convention, we define V_j as the map points visible in the frame with index j and use S to represent the selected map points. The weighted coverage in Equation (4.60) can equivalently be written in terms of S as

$$f_{\mathrm{wcover}}(S) = \sum_{j=1}^{t} |S \cap V_j| + \lambda \min(|S \cap V_j|, b_{\mathrm{cover}}). \tag{4.61}$$

This is a new weighted coverage utility function usable by greedy algorithms for the selection of map points.

The lazy greedy and stochastic greedy algorithms use the marginal gain when selecting map points, which is given by the utility for selecting a map point, or $f(m_i|S) = f(S \cup \{m_i\}) - f(S)$ (as noted in Equation (3.22) from Subsection 3.2.2). Note that while the utility in Equation (4.61) sums over all the frames, only the subset of frames where the map point was visible, or X_i, contribute to the calculation of the marginal gain. The number of selected map points visible in each frame only changes at these frames where the map point m_i is visible. The summation terms over other frames in the trajectory where the map point is not visible are unaffected if the map point is selected and cancel out when calculating the marginal gain. Therefore, the marginal gain of Equation (4.61) is given by

$$\begin{aligned} f_{\mathrm{wcover}}(m_i|S) &= f(S \cup \{m_i\}) - f(S) \\ &= |X_i| + \lambda \sum_{j \in X_i} \min(|(S \cup \{m_i\}) \cap V_j|, b_{\mathrm{cover}}) - \min(|S \cap V_j|, b_{\mathrm{cover}}), \end{aligned} \tag{4.62}$$

where the marginal gain of the first term in Equation (4.61) is the number of camera

frames from which a map point is visible.

Evaluating the marginal gain has a complexity of $O(|X_i|)$ with this implementation. As noted in Subsection 3.2.2, the lazy and stochastic greedy algorithm requires evaluating the marginal gain at worst $O(n^2)$ and $O(n)$ times when selecting $O(n)$ map points from a set of $O(n)$. The asymptotic computational complexity of the function evaluations is $O(|X_i|n^2)$ for the lazy greedy algorithm and $O(|X_i|n)$ for the stochastic greedy algorithm.

This subsection adapts the existing coverage-based approach, which was previously used in integer programming techniques, to an equivalent cardinality-constrained submodular maximisation problem. The adaptation in this subsection allows this dissertation to use greedy algorithms to approximate the equivalent maximisation problem, which offers a computationally inexpensive polynomial time complexity alternative compared to the previous approach based on integer programming techniques. Using greedy algorithms could translate to significantly reduced execution times for larger maps.

4.3.3. Map Point Selection as Maximising the Minimum Coverage

The coverage-based approach from Subsection 4.3.1 and its adaptation in Subsection 4.3.2 formulates the map point selection problem as the trade-off between selecting map points with the highest number of observations and the coverage. These approaches require choosing the desired coverage value b_{cover} and the λ trade-off parameter. This subsection proposes an alternative coverage-based approach based on maximising the minimum number of observations visible in a camera frame. The approach in this subsection is parameter-free, but requires a different greedy algorithm than those presented in Subsection 3.2.2 to find the set of map points that approximately maximises the utility function.

This subsection proposes a simple alternative formulation where we maximise the minimum number of selected map points in each camera frame. We define the utility of a frame with index j in the trajectory as

$$f_{\text{cover},j}(S) = |S \cap V_j|. \tag{4.63}$$

For a selected set of map points, this subsection defines the utility as the number of selected map points visible in the frame with the minimum number of selected map points, or

$$f_{\text{min-cover}}(S) = \min_j f_{\text{cover},j}(S), \tag{4.64}$$

where $f_{\text{cover},1}, ..., f_{\text{cover},t}$ are the functions for each frame as defined in Equation (4.63).

The utility in Equation (4.64) is not submodular and, therefore, incompatible with the greedy algorithms from Subsection 3.2.2. The rest of this subsection presents an overview of the SATURATE algorithm [65] that allows approximating maximisation problems of this form, also called the robust submodular maximisation problem. For this problem, we

are given a cardinality constraint k and are interested in maximising the minimum of a range of submodular functions, or

$$\underset{S\subseteq V,|S|<k}{\arg\max}\left(\min_{j} f_j(S)\right), \tag{4.65}$$

where $f_1, ..., f_t$ are monotone submodular functions.

Algorithm 4: SATURATE algorithm for the robust submodular maximisation problem

input : Utility functions $f_1(S), ..., f_t(S)$, set of items V and budget k, increased budget factor α_{sat}

output : S_g

while $|b_{min} - b_{max}| > 1$ **do**
- $b_{\text{max}} = \max_j f_j(V)$
- $b_{\text{min}} = 0$
- $b_{\text{test}} \leftarrow \frac{b_{\text{min}}+b_{\text{max}}}{2}$
- $S_{\text{test}} \leftarrow \emptyset$
- $V_{\text{test}} \leftarrow V$
- $f_{\text{sat}}(S, b_{\text{test}}) \leftarrow \sum_{j=1}^{t} \min(f_j(S), b_{\text{test}})$
- **while** $f_{sat}(S_{test}, b_{test}) \neq f_{sat}(V, b_{test})$ **do**
  - $m_i \leftarrow \underset{m_i \in V_{\text{test}}}{\arg\max} f_{\text{sat}}(m_i|V_{\text{test}}, b_{\text{test}})$
  - $S_{\text{test}} \leftarrow S_{\text{test}} \cup \{m_i\}$
  - $V_{\text{test}} \leftarrow V_{\text{test}} \setminus \{m_i\}$
- **if** $|S_{test}| > \alpha_{sat} k$ **then**
  - $b_{\text{max}} \leftarrow b_{\text{test}}$
- **else**
  - $b_{\text{min}} \leftarrow b_{\text{test}}$
  - $S_g \leftarrow S_{\text{test}}$

Algorithm 4 describes this algorithm for the robust submodular maximisation problem used by this subsection. This algorithm formulates the saturated objective, which is the sum of the utility functions $f_j(S)$, each limited to a utility value b_{sat}, or

$$f_{\text{sat}}(S, b_{\text{sat}}) = \sum_{j=1}^{t} \min\left(f_j(S), b_{\text{sat}}\right). \tag{4.66}$$

The key observation of this algorithm is that for a given value b_{sat}, it is possible to formulate a new saturated optimisation problem to evaluate the minimum number of map points

needed such that $f_j(S) > b_{\text{sat}}$ by formulating a new optimisation problem, or

$$S^* = \underset{S \subseteq V, f_{\text{cover}}(S, b_{\text{sat}}) = f_{\text{cover}}(V, b_{\text{sat}})}{\arg\min} |S|. \tag{4.67}$$

This problem in Equation (4.67) is the submodular cover problem [66; 67]. Similar to the case for submodular maximisation, this can be approximated by greedily selecting map points until the constraint is satisfied, as shown in lines 8 to 11. Since the maximum value for b_{sat} for the given map point budget is not known in advance, a binary search procedure is used. The upper bound of this search range, or $b_{\max}$, is given by the maximum utility for any function in the range using the full set of potentially selected elements V, since these functions are monotone. The functions are assumed to be normalised, and therefore, the minimum value is $b_{\min} = 0$.

The greedy algorithm does not optimally solve the submodular cover problem in Equation (4.67). Therefore, the SATURATE algorithm inflates the allowable budget of map points by a factor of α_{greed}. Unless $P = NP$, in general, no efficient algorithm can provide constant factor guarantees with respect to the utility of the selected set for the robust submodular maximisation problem [65]. The algorithm instead provides a weaker guarantee where it is possible to find a solution with a utility competitive with the intractable optimal solution that uses k selected items, provided the greedy algorithm is given an increased constraint of $\alpha_{\text{greed}}k$ items, where α_{greed} is given by

$$\alpha_{\text{greed}} = 1 + \log\left(\max_{m_i \in V} \sum_{j=1}^{t} f_j(\{m_i\})\right). \tag{4.68}$$

This dissertation uses the parameter α_{sat} for the chosen value with which Algorithm 4 inflates the map point budget. A choice is to use $\alpha_{\text{sat}} = \alpha_{\text{greed}}$ to obtain guarantees with respect to the optimal solution obtained with a map point budget of k, whereas the algorithm uses a budget of $k\alpha_{\text{greed}}$ map points. This dissertation uses an alternative choice to use the algorithm without inflating the map point budget, i.e. $\alpha_{\text{sat}} = 1$ instead of α_{greed}. This parameter choice results in the SATURATE algorithm returning a set of map points that do not contain more than k map points. However, the utility of the returned solution is only competitive in the worst case with the utility of the optimal set for the map point budget of $\frac{k}{\alpha_{\text{greed}}}$.

Maximising the minimum coverage from Equation (4.67) with Algorithm 4 has an asymptotic complexity of $O(\log(b_{\max})n^2|X_i|)$. This complexity is worse than using the lazy greedy algorithm with the utility from Subsection 4.3.2, which has an asymptotic complexity of $O(n^2|X_i|)$.

The SATURATE algorithm is not limited to coverage-based functions and applies to any range of submodular functions. This dissertation explored using the SATURATE algorithm with the approximate information-theoretic functions from Section 4.2. Rather

than summing over the respective localisation or odometry utilities for each frame, this algorithm can be used to maximise the minimum odometry or localisation utility. However, optimising for the minimum utility did not improve map point selection performance compared to using lazy-greedy with the sum of the frame utilities.

Using the min-cover utility from Equation (4.64) with the SATURATE algorithm has two potential advantages and disadvantages over maximising the utility from Subsection 4.3.2 with the lazy greedy algorithm. The first advantage is that this approach does not require setting any parameters. This aspect could be particularly advantageous since choosing these parameters may be challenging if little is known about the scale of the environment with respect to the map point budget. The second advantage is that optimising for the minimum number of map points visible in a frame could have improved robustness and ensured a more uniform distribution of map points. A disadvantage of this approach is that it is computationally more expensive than the lazy greedy algorithm. A further disadvantage is that optimising for the worst case typically results in worse average-case performance. In some cases, performing well on average may be more critical than the worst-case performance. For example, given a minimal map point budget, it may be more beneficial to have some camera frames with a sufficient number of map points than an even distribution where no camera frames have sufficient map points.

This subsection proposed an alternative heuristic to the weighted coverage utility used in Subsections 4.3.1 and Subsection 4.3.2. Unlike the other utility functions in this chapter, the subsection specifically proposes to use the SATURATE algorithm with the utility from Equation (4.64) to maximise the minimum number of map points visible in a frame. The approach of this subsection is parameter-free, compared to the coverage-based approaches from the previous subsections. However, the algorithm of this approach involves an iterative search, which incurs additional computational overhead compared to the greedy algorithm from Subsection 4.3.2.

4.4. Combined Utility Formulation for Improved Loop-closure Detection

Section 4.2 proposed approximate information-theoretic models for map point utility based on the estimation performed in the SLAM back-end. We expect that a definition of map point utility that considers both the SLAM front-end and back-end should outperform approaches that only consider the back-end estimation problem. Consequently, this section proposes the augmentation of an information-theoretic utility with an additional term to improve its ability to capture aspects of the front-end, specifically loop closure detection. Section 6.2 shows that coverage-based functions can outperform alternatives in terms of loop-closure recall. This section, therefore, defines a combined utility function that can

be formulated as the weighted sum of the information-theoretic utility function and a coverage-based loop-closure utility function or

$$f_{\text{info+loop}}(S) = w_{\text{info}} f_{\text{info}}(S) + w_{\text{loop}} f_{\text{loop}}(S). \tag{4.69}$$

Submodularity is closed under non-negative linear combinations. For two submodular functions, $f_{\text{info}}(S)$ and $f_{\text{loop}}(S)$, and non-negative weights w_1 and w_2, the weighted sum $f_{\text{info+loop}}(S)$ is also monotone submodular.

We make the simplifying assumption that we can obtain the subset of keyframes or poses where it is necessary to be able to perform loop closures. In scenarios where a planner is available, this information could be extracted by evaluating the future trajectory. Alternatively, such information could be obtained from application-specific information. For some applications, it would be feasible to extract the likelihood or probability that a loop closure occurs in a specific scene (for example, by training a neural network-based classifier). In cases where such information is not available, this version of the algorithm serves as a comparison point against which to evaluate other algorithms.

Similar to the greedy coverage-based utility functions from Subsection 4.3.2, we use the bounded sum of the number of selected map points visible in a frame, or the frame coverage. Using the previous assumption from this section that the loop-closure frames L are available, the coverage-based utility function is defined using only those frames. For each frame j, we define the map points visible in that frame as V_j. The loop closure coverage is then given by

$$f_{\text{cover}}(S) = \frac{1}{|L|} \sum_{j \in L} \min\left(|S \cap V_j|, b_{\text{cover}}\right), \tag{4.70}$$

where b_{cover} is the chosen coverage parameter and should be adjusted based on the number of map points needed to perform loop closures reliably. This utility definition will be used as the utility $f_{\text{loop}}(S)$ in the combined formulation from Equation (4.69).

When using a single utility function for selecting map points, the single utility function should capture the utility of map points for both the front- and back-end. In contrast, when using a combined utility function, we can start with a utility function that already captures the utility for the back end and design a coverage-based function to capture the utility for loop closures specifically. This approach allows assigning a higher utility to these specific loop-closure map points undervalued by the original information-theoretic function. Using a more generic coverage-based formulation that does not focus on loop closures overestimates the utility of map points not relevant to loop closures and does not necessarily lead to improved map point selection performance.

This dissertation investigated alternative models of the loop-closure utility using the probabilistic models of the expected number of loop-closure feature matches. Initial

experimentation of this approach found that probabilistic approaches did not necessarily outperform the utility presented here. In a combined formulation, the information-theoretic term already causes greedy algorithms to assign higher utilities to map points with more observations. A notable challenge in obtaining reliable estimates of the probabilities during long-term and large-scale loop closures is that the feature-matching algorithms differ from the mechanism used in short-term feature matching (refer to Subsection 3.1.1 for a description of ORB-SLAM 2's front-end).

This section chooses to use the odometry utility from Subsection 4.2.2 as the information-theoretic utility, or $f_{\text{info}}(S)$, in the combined formulation shown in Equation (4.69). This choice for $f_{\text{info}}(S)$ could have been the localisation approach from Subsection 4.2.1. However, preliminary experimentation showed that the odometry utility often obtained improved trajectory accuracy over using the localisation utility. For the relative weighting of these two utility functions, or w_{info} and w_{loop} in Equation (4.69), it is important to consider that the utility functions are scaled differently. Since these functions are monotone, the maximum utility for each utility function is given by selecting all the available map points, or V. This dissertation found that using equal weighting after normalising the respective utilities with the utility obtained for selecting all map points, or $f(V)$, worked well despite the heuristic nature of this approach. The normalised sum of our utility functions $f_{\text{odom}}(S)$ and $f_{\text{cover}}(S)$ yields the combined utility function:

$$f_{\text{odom+cover}}(S) = \left(\frac{1}{f_{\text{odom}}(V)}\right) f_{\text{odom}}(S) + \left(\frac{1}{f_{\text{cover}}(V)}\right) f_{\text{cover}}(S). \tag{4.71}$$

Greedy algorithms require evaluating the marginal gain, which is given by the sum of the marginal gains of the two functions, or

$$f_{\text{odom+cover}}(\{m_i\}|S) = \frac{f_{\text{odom}}(\{m_i\}|S)}{f_{\text{odom}}(V)} + \frac{f_{\text{cover}}(\{m_i\}|S)}{f_{\text{cover}}(V)}. \tag{4.72}$$

For both the odometry utility in Equation (4.37) and the coverage-based utility for loop-closure frames in Equation (4.70), evaluating the marginal gain of a map point requires only considering the co-visible poses X_i. The asymptotic computational complexity of this combined function is, therefore, also $O(|X_i|)$, which we expect will be bounded for many SLAM datasets. This complexity for the marginal gain leads to an asymptotic computational complexity of $(|X_i|n)$ and $O(|X_i|n^2)$ for the function evaluations for the lazy greedy and stochastic greedy algorithms.

This chapter covers different information-theoretic and coverage-based utility functions for map point selection. Most of these utility functions can be maximised using the lazy and stochastic greedy algorithms from Subsection 3.2.2. Section 4.1 proposes a utility using the full SLAM posterior, but this approach proved computationally expensive, and Section 4.2 proposed approximations based on either the localisation or odometry problems

to offer inexpensive alternatives. Section 4.3 adapted different coverage-based approaches for use in SLAM, including an existing approach using integer-programming techniques, an adaptation of this coverage-based approach to be compatible with greedy algorithms for reduced computational complexity and an approach with a utility function based on the minimum number of selected map points visible in a frame that requires the SATURATE algorithm to maximise. Lastly, this section proposes a formulation that assumes additional information is available to combine the approximate odometry information-theoretic utility with a coverage-based utility for improved loop-closure detection.

Chapter 5

Evaluation Setup for Map Point Selection Approaches

It is necessary to use a representative stereo SLAM implementation to evaluate the impact of map point selection. Using a representative SLAM software algorithm allows testing on practical data and evaluates how map point selection impacts the overall SLAM implementation. This dissertation chooses ORB-SLAM 2 as a representative algorithm to evaluate map point selection approaches. However, this chapter introduces modifications to the algorithm to facilitate the evaluation of different map point selection approaches. This chapter presents the modified version of ORB-SLAM 2, evaluation metrics and test procedure used for the experimental results in Chapter 6.

There are two stereo vision datasets compatible with the ORB-SLAM 2 algorithm. For the scope of this work, this dissertation chooses to use the stereo vision datasets with ground truth information already compatible with the ORB-SLAM 2 algorithm: The KITTI odometry and EuRoC MAV datasets. ORB-SLAM 2 also supports the TUM-RGBD dataset by converting RGBD measurements into pseudo-stereo measurements. However, we did not use this dataset or functionality, as it would require adapting the proposed information-theoretic approaches for RGBD sensors.

The rest of the chapter is organised as follows: First, Section 5.1 provides an overview of evaluation metrics to measure the performance of a SLAM algorithm. These metrics will be used to measure the impact of different map point selection approaches on the accuracy of a SLAM algorithm. Second, Section 5.2 presents an overview of the ORB-SLAM 2, how its design impacts the evaluation of map point selection and motivates a series of modifications to the algorithm that facilitate the evaluation of map point selection approaches. Section 5.3 proposes an offline test procedure using the modified implementation of ORB-SLAM 2. Baseline map point selection approaches are detailed in Section 5.4. These baseline approaches serve both as points of comparison for different map point selection approaches and as a means to evaluate the measurable impact of map point selection on potential datasets. Sections 5.5 and 5.6 provide details for using the KITTI and EuRoC datasets with the proposed test procedure in Section 5.3.

5.1. Evaluation Metrics for Map Point Selection

The accuracy of SLAM algorithms is typically evaluated using the estimated robot trajectory and comparing it against the ground truth trajectory [68; 12]. However, map point selection also impacts the performance of the loop-closure detection. This dissertation uses both the estimated trajectory of the ORB-SLAM 2 algorithm and loop-closure detection performance to evaluate different map point selection approaches.

The estimated SLAM trajectory for SLAM is typically evaluated using both the absolute pose error (APE) and relative pose error (RPE). Kümmerle *et al.* [69] proposed relative pose errors to evaluate SLAM algorithms. Using relative pose error overcomes the limitations of absolute position metrics that penalise errors earlier in the trajectory to a greater degree than later in the trajectory. However, the relative pose error can be measured at different scales. Due to varying the accuracy of ground truth trajectories and scales of datasets, different relative pose definitions are used for different datasets (e.g. [12; 68]). Mur-Artal and Tardós [8] used these different definitions when evaluating the ORB-SLAM 2 algorithm on the KITTI and EuRoC datasets. Subsequently, this dissertation also uses these different definitions of relative pose error for the different scales of these datasets.

Before discussing trajectory error metrics, it is necessary to introduce some additional notation. For these metrics, we assume that poses are represented by 4×4 transformation matrices; that is, the combination of the position vector and rotation matrix. For each pose with index i along the trajectory, we define the ground truth pose $\mathbf{X}_i$ and estimated pose $\hat{\mathbf{X}}_i$. We also define $||.||_{\text{trans}}$ as the L2 norm of the translational components.

The absolute poses error (APE) calculates the difference in position between each pose of the estimated trajectory and the ground truth trajectory. The estimated trajectory is first aligned to the ground truth trajectory using the method proposed by Umeyama [70]. For the transformation matrix $\mathbf{T}$ that minimises the distance between the two trajectories, the absolute pose error for each pose with index i, or e_i, is the distance between the aligned pose $\mathbf{T}\hat{\mathbf{X}}_i$ and the ground truth pose $\mathbf{X}_i$, or

$$e_i = ||(\mathbf{T}\hat{\mathbf{X}}_i)^{-1}\mathbf{X}_i||_{\text{trans}}. \tag{5.1}$$

The absolute pose error is the root mean squared error (RMSE) of the absolute pose errors in Equation (5.1), or

$$\text{APE} = \sqrt{\frac{1}{t}\sum_{i=1}^{t}(e_i)^2}. \tag{5.2}$$

The relative pose error (RPE) uses the relative distance between pairs of poses to evaluate the error. For each pair of poses with indices i and j, the relative pose error (RPE) between a pair of poses, or $e_{i,j}$, is equal to the difference between the relative pose

in the estimated trajectory versus that of the ground truth trajectory, or

$$e_{i,j} = \left|\left|(\mathbf{X}_i^{-1}\mathbf{X}_j)^{-1}(\hat{\mathbf{X}}_i^{-1}\hat{\mathbf{X}}_j)\right|\right|_{\text{trans}}. \tag{5.3}$$

For the EuRoC dataset, the relative pose error is calculated using the definition of Sturm *et al.* [68] using all pairs of frames 15 frames apart. As with APE, the root-mean-square error is used, or

$$\text{RPE} = \sqrt{\frac{1}{t-15}\sum_{i=1}^{t-15}(e_{i,i+15})^2}. \tag{5.4}$$

The KITTI dataset uses a distanced normalised variation of the relative pose error that combines the relative pose errors measured over varying distances into a single metric [12]. The KITTI dataset evaluates errors using varying distances along the ground truth trajectory using the cumulative Euclidean distance, where this distance between poses with indices i and j is defined as

$$d_{ij} = \sum_{y=i}^{j-1} ||\mathbf{X}_y^{-1}\mathbf{X}_{y+1}||_{\text{trans}}. \tag{5.5}$$

This definition of the RPE is in contrast to Equation (5.4), where the RPE was evaluated using the distance in pose indices. To evaluate the relative pose error at a given pose index i and cumulative Euclidean distance Δ_k, a matching pose j is found first. This matching pose with index j is the smallest pose index where the corresponding pose is at least Δ_k away from the pose with index i using the cumulative Euclidean distance, or $d_{ij} \geq \Delta_k$. The normalised relative pose error for a given pose index i, pose index j and a distance Δ_k is given by

$$e_{i,\Delta_k,j} = \frac{1}{\Delta_k}||(\mathbf{X}_i^{-1}\mathbf{X}_j)^{-1}(\hat{\mathbf{X}}_i^{-1}\hat{\mathbf{X}}_j)||_{\text{trans}}. \tag{5.6}$$

The KITTI dataset only uses every tenth pose in the trajectory as a potential starting pose i to evaluate the relative pose error instead of all the poses in the trajectory. The cumulative distance Δ_k at which errors are evaluated is varied within the range $\{100, 200, \ldots, 800\}$ metres and used to find a pose j that is Δ_k further along the ground truth trajectory. A matching pose j only exists for some pairings of i and Δ_k, as the remaining portion of the trajectory might not be long enough. Therefore, before calculating the error metric, the set of evaluation triples, E, is calculated first. This set contains all valid groupings of index i, distance Δ_k and matching pose with index j. The root mean squared error of the normalised relative pose error for the set of evaluation triples is then given by

$$\text{RPE}_{\%} = \sqrt{\frac{1}{|E|}\sum_{(i,\Delta_k,j)\in E}(e_{i,\Delta_k,j})^2}. \tag{5.7}$$

Map point selection will affect not only the SLAM trajectory accuracy but also loop-

closure detection. Suppose map point selection impacts the ability to detect loop closures. In that case, this causes a loss in trajectory accuracy due to the inability to correct the large-scale errors in the estimated trajectory. Therefore, this dissertation also evaluates the performance of the loop-closure detection module as this provides insight into interpreting the cause of a loss in trajectory accuracy.

Loop-closure detection is typically modelled as a classification problem, where the problem is to label a loop-closure candidate as either a valid loop closure or an outlier. The performance of classifiers is typically evaluated using both recall and precision. Recall, in this instance, is the ratio between the correctly labelled loop closures and the ground-truth loop closures, or

$$\text{Recall} = \frac{\text{Correctly labelled loop closures}}{\text{Ground truth loop closures}}. \tag{5.8}$$

Precision is the ratio between the correctly labelled loop closures and all the frames labelled as loop closures, including frames incorrectly labelled as loop closures, or

$$\text{Precision} = \frac{\text{Correctly labelled loop closures}}{\text{Frames labelled as loop closures}}. \tag{5.9}$$

For classification problems, precision and recall are competing objectives. For example, it is trivial to obtain 100% recall by simply labelling all frames as loop closures at the cost of precision. Conversely, 100% precision can be obtained by not labelling any frames as loop closures if no frames are incorrectly labelled as loop closures. Loop-closure detection algorithms used for visual SLAM algorithms are designed to have very high precision at the expense of recall. While not shown here, it was found that recall performance can drop significantly for the map point selection approaches, and precision is mostly unaffected. These algorithms are designed to be very conservative with respect to rejecting false loop closures and require a high threshold of feature matches. Map point selection reduces the number of available map points and associated matches, typically resulting in these loop-closure detection algorithms rejecting even more loop closures and causing a drop in recall. As a consequence, Chapter 6 only reports recall for loop-closure performance for map point selection, as precision remained at 100%, and this was sufficient to interpret when a loss in loop-closure detection performance impacts the estimated trajectory accuracy.

This section reviewed APE and RPE metrics for SLAM algorithms that measure the estimated trajectory accuracy. Map point selection will also impact loop-closure detection, and therefore, it is beneficial to measure the performance of this sub-component, in particular, using recall.

5.2. Modified ORB-SLAM 2 for evaluating map point selection

This dissertation chooses ORB-SLAM 2 as a representative sparse visual SLAM algorithm but introduces modifications to facilitate the evaluation of different map point selection approaches. This section provides an overview of the ORB-SLAM 2 algorithm, its properties and motivates the modifications made.

Subsection 5.2.1 discusses the ORB-SLAM 2 algorithm and its properties when used to evaluate different map point selection approaches. This dissertation uses this modified version of ORB-SLAM 2 presented in Subsection 5.2.2. These modifications aim to mitigate the impact of the properties mentioned in Subsection 5.2.1 when evaluating different map point selection approaches.

5.2.1. ORB-SLAM 2 for Map Point Selection

This subsection motivates our choice using ORB-SLAM 2 as a representative algorithm and provides an overview of the algorithm. This subsection then further highlights different properties from ORB-SLAM 2 that are undesirable when using the algorithm to evaluate different map point selection approaches.

ORB-SLAM 2 has state-of-the-art accuracy, has validated performance on a range of datasets, and is open source [8; 28]. We found that ORB-SLAM 3 performs worse than the original ORB-SLAM 2 algorithm in the KITTI dataset when using only stereo cameras (as shown in Section 5.5). ORB-SLAM 3 includes improved support for different sensors, such as inertial measurement units, which we do not use here. For these reasons, this dissertation chose to focus on ORB-SLAM 2 despite the availability of a new version.

Before we discuss the properties of ORB-SLAM 2, this subsection briefly reviews the design of this SLAM algorithm. To allow real-time operation, ORB-SLAM 2 separates the SLAM problem into four sub-components running in different threads: *tracking*, *mapping*, *loop closing*, and *bundle adjustment*. The *tracking* sub-component runs at the camera frame rate and localises the camera using the current map. This localisation is done at the camera frame rate while the other sub-components of the algorithm run at a reduced frequency. Only a subset of frames (or keyframes) are used in the mapping sub-component, which performs SLAM in a local region. This sub-component dynamically inserts new keyframes based on various heuristics, such as whether the mapping sub-component has completed the local SLAM estimation. Concurrently, the loop closing sub-component uses appearance-based techniques to search for large-scale loop closures. If a loop closure is found, an initial approximate pose-graph heuristic is used to update the map. Pose-graph optimisation is a cheap approximation of the SLAM posterior over camera poses using the visual overlap (or the number of co-visible map points) between camera poses. Importantly,

this update does not model map point measurements. After this heuristic update, the bundle adjustment sub-component jointly estimates all past poses and map points in another parallel thread using the updated pose-graph estimate as initialisation. Once the estimation for the bundle adjustment is completed, the joint map used in all the sub-components is updated.

ORB-SLAM 2's performance varies significantly between multiple runs of the algorithm on the same input data. The specific multi-threaded implementation of the algorithm contributes to this variation. The different threads of the ORB-SLAM 2 algorithm each operates on the same shared map and estimated pose data without an enforced order of execution. The timing of threads in a multi-threaded application is non-deterministic and depends on various factors, including the other threads processed on the same CPU. ORB-SLAM 2's thread-timing dependent nature is further worsened by the decision of heuristics to govern decisions such as the insertion of keyframes based on whether threads are currently busy. While these choices are motivated to allow the algorithm to scale to available computational resources, these software design choices result in inconsistent and hardware-dependent estimation accuracy.

This dependence on timing results in two important challenges when used for evaluating map point selection approaches: the first challenge is that the algorithm is stochastic and can exhibit significant variation between runs of the algorithm. This variation can prevent consistent results in different map points between runs, which makes it challenging to compare the relative performance of map point selection approaches on the same data. The second challenge is that some approaches (such as the SLAM utility function from Section 4.1) are unsuitable for online use. This timing dependence also means that any potential changes to the implementation will impact the timing of threads and the resulting accuracy of the algorithm. This dependence is problematic if we want to test map point selection approaches based on the quality of the selected map points, as this dependence will penalise more computationally expensive approaches. Evaluating the relative performance of offline approaches against map point selection approaches suitable for online use requires further consideration.

The design choices of using an initial pose-graph optimisation and a limited number of bundle adjustment iterations attempt to strike a good compromise between the real-time nature and accuracy of the algorithm. This scheme of first using the pose graph estimates to continue online estimation, while bundle adjustment is performed in parallel, means that the impact of map point selection on the SLAM estimation is only measured after bundle adjustment finishes. Mur-Artal *et al.* [6] found that only ten iterations of bundle adjustment are often necessary after the initial pose-graph optimisation for ORB-SLAM, and this parameter choice was also used in ORB-SLAM 2. A limited number of bundle adjustment iterations can prevent estimation from converging to a new estimate after the map has changed due to map point selection. As a result, the limited number of iterations

diminishes the impact of map point selection on the estimated trajectory. The resulting trajectory accuracy is more dependent on the accuracy of the pose-graph heuristic than the bundle adjustment accuracy.

Another property is that some of the existing heuristics of the algorithm are not properly designed for the reduced maps resulting from applying map point selection techniques. For example, ORB-SLAM 2 identifies unnecessary keyframes to remove based on the ratio between map points shared with nearby keyframes and the number of unique map points. This ratio is no longer a good metric in the presence of map point selection. Map point selection approaches generally select a small subset of map points visible in many keyframes. Map point selection can result in this keyframe removal heuristic to mark many keyframes after selecting map points as redundant, as many of the unique map points that are only visible from a small subset of keyframes are typically not selected. Subsequently, the heuristic incorrectly removes large sections of trajectory. For the evaluation of map point selection approaches, the poses must continue to be estimated to evaluate the impact of selection on trajectory estimation.

This subsection motivated our choice for ORB-SLAM 2 and provided an overview of the algorithm. As mentioned in this subsection, different aspects of the algorithm stemming from its design are undesirable for evaluating different map point selection approaches. This subsection highlighted these properties and how they impact the evaluation of map point selection approaches.

5.2.2. Modifications to the ORB-SLAM 2 algorithm

Subsection 5.2.1 provides an overview of the properties of the ORB-SLAM 2 algorithm that complicate the evaluation of different map point selection approaches. Addressing these challenges is essential to allow the fair and accurate evaluation of different map point selection approaches using a SLAM implementation. Due to the timing dependence of the algorithm and map point selection approaches, which are prohibitively expensive to use online, the approach of this chapter is to use map point selection offline. The ORB-SLAM 2 algorithm is also modified to improve the ability to evaluate map point selection approaches. This subsection presents an overview of the modifications made to the ORB-SLAM 2 algorithm. This modified version of ORB-SLAM 2, used for evaluating map point approaches, is publicly available[1].

The first modification was to add the ability to store and load the map generated by the ORB-SLAM 2 algorithm. This modification allows the use of map point selection approaches offline. This modification also allows testing the map point selection approach on the same input maps despite the stochastic nature of the ORB-SLAM 2 algorithm. The resulting reduced input map is then loaded from storage and used by the modified

[1] https://github.com/ChristiaanM/ORB_SLAM2_selection

ORB-SLAM 2. This test procedure is described in greater detail in Section 5.3.

The second modification to the ORB-SLAM 2 algorithm was to increase the number of bundle-adjustment iterations and modify the algorithm to wait for this SLAM estimation after loop-closures to finish before processing new frames. This change allows estimation to converge more reliably at the expense of worse real-time performance. We found that this modification allows the algorithm to provide more consistent results, which are beneficial for evaluating the impact of map point selection. This change places a greater emphasis on bundle adjustment accuracy when evaluating map point selection approaches and a reduced emphasis on the accuracy of the real-time approximations and the accuracy of the pose graph estimation.

The third modification was adjusting the keyframe removal heuristic to avoid removing keyframes with a limited number of map points. This ensures that all poses are included in the estimation, and their estimates can continue to be updated by bundle adjustment and influence the resulting evaluation of the map point selection approach. Introducing map point selection into other visual SLAM algorithms would require a similar revision of heuristics that are no longer appropriate in the presence of map point selection.

This section proposes using a modified version of ORB-SLAM 2 to evaluate different map point selection approaches. These modifications were motivated to mitigate the undesirable properties identified in Subsection 5.2.1. The majority of these modifications are intended to allow for more consistent results and facilitate a fair evaluation of the different map point selection approaches despite the potential offline nature of map point selection approaches.

5.3. Test Procedure for Evaluating Map Point Selection Approach

This section describes the offline test procedure used by this dissertation to evaluate different map point selection approaches. When map point selection is implemented online, the map should be reduced whenever the number of map points exceeds a specified map point budget. Such an online implementation requires repeatedly solving the map point selection problem during operation. In contrast, the offline test procedure described in this section approximates the evaluation of map point selection approaches by only applying a map point selection at a single timestep in the robot's trajectory and uses map point selection approaches offline. The aforementioned procedure relies on the modification as described in Subsection 5.2.2 to the ORB-SLAM 2 algorithm that allows storing maps offline. This section first presents the approximate test procedure before discussing the benefits of using this approximation.

Figure 5.1 shows a flow chart of the test procedure. The test procedure segments

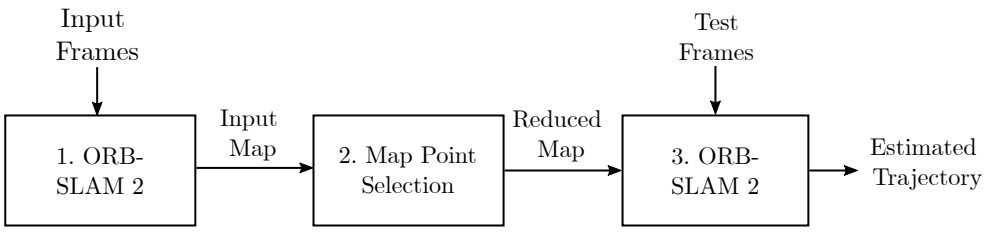

Figure 5.1: This flow chart shows the sequential steps used to evaluate map point approaches with offline selection. An initial input portion of a trajectory is used with a modified ORB-SLAM 2 implementation to generate an input map. A map point selection approach is then applied offline to the input map to simplify the map. Lastly, the modified ORB-SLAM 2 algorithm is started again using the remaining test portion of the trajectory and the reduced map as input.

trajectories from SLAM datasets into two consecutive parts: the input trajectory and the test trajectory. The input trajectory is used with the modified version of ORB-SLAM 2 to create an input map, which is stored. Different map point selection approaches are then applied to the same input map. The reduced maps generated by the different selection approaches then replace the input map in the modified ORB-SLAM 2, after which SLAM is performed on the remaining test trajectory. We detail how we select the input and test trajectories for the KITTI and EuRoC datasets in Sections 5.5 and 5.6, respectively.

Each map point selection approach is evaluated on all the chosen trajectories of a dataset. As mentioned in Subsection 5.2.1, the performance of the ORB-SLAM 2 algorithm is inconsistent. While the changes in Subsection 5.2.2 improve consistency, the stochastic nature of the algorithm is still present for the modified version of ORB-SLAM 2. This inconsistency results in different input maps being generated between different runs of the modified ORB-SLAM 2 algorithm despite using the exact same input trajectory. This dissertation chooses to use a set of five input maps, instead of only a single map, to capture the performance of map point selection approaches over a range of potential realisations of the input map. Using only a single input map will present a more noisy measurement of the trajectory accuracy performance than using a group of input maps. Additionally, we evaluate each reduced input map twice to mitigate variation in the estimated trajectory and improve our ability to measure recall performance. The impact of the variation in maps resulting from the stochastic nature of the algorithm for the KITTI data is investigated in greater detail in Section 5.5.

Despite the approximate nature of this offline test procedure, this procedure has three notable advantages over online evaluation: the first advantage is that it allows for the evaluation of different map point selection approaches on exactly the same maps. This is not possible online due to the significant variation of the ORB-SLAM 2 algorithm between runs even on the same input data. Using the same map is also beneficial when evaluating the impact of the choice of map point selection approach with a limited number of samples. The second advantage is that it allows for the evaluation of approaches that are not suitable

for online applications, such as using the SLAM utility from Section 4.1. This approach serves as a vital point of comparison for approximate information-theoretic approaches from Section 4.2. The third advantage is that it prevents the potential bias due to additional computation from affecting the timing of threads and potentially indirectly impacting the trajectory accuracy due to ORB-SLAM 2's design, as mentioned in Subsection 5.2.1.

This section presents the test procedure used to evaluate different map point selection approaches offline. This test procedure requires separating the trajectory from a SLAM dataset into an "input trajectory" and a "test trajectory". Due to the stochastic nature of the ORB-SLAM 2 algorithm, multiple input maps are generated using the input trajectory. The same group of maps are then evaluated using different map point selection approaches. This test procedure approximates the online application of different map point selection approaches, but importantly facilitates fair comparison between approaches and also allows comparison with approaches not suitable for online applications.

5.4. Baseline Map Point Selection Approaches

This section reviews some baseline map point selection approaches. These baselines serve as the expected upper bound for trajectory accuracy, the expected lower bound or as an uninformative baseline for different map point selection approaches.

The first baseline is including all the map points in the input map (or not using map point selection). We expect to achieve the best trajectory using all the available map points from the input map. The proposed map point selection approaches aim to offer comparable accuracy to this approach at lower input map sizes. We label this approach "full map".

The second baseline is selecting no map points from the input map, which implies discarding the input map. Generally, selecting no map points serves as a lower bound for trajectory accuracy. Selecting map points should offer better trajectory accuracies than selecting no map points. However, the ORB-SLAM 2 algorithm resets if it fails to localise itself in an existing map and evaluating the trajectory for no map points in the input map is not meaningful. Instead of selecting no map points, we only include all the map points visible in the last keyframe of the input map. This is equivalent to starting a new map with the test trajectory and consequently prevents any loop closures with the input trajectory. We label this approach the "empty map" approach.

As a consequence of the aforementioned tracking requirement, all map point selection approaches are modified to use the simple heuristic of including all the map points observed in the last keyframe in the input map. This heuristic ensures that the ORB-SLAM 2 implementation can continue tracking during the test trajectory after the map has been reduced. The cost of this heuristic in map points is often negligible compared to the rest of the map. Applying the same heuristic across the different methods ensures a fair evaluation

of the respective algorithms. Instead, map points are selected from the remaining subset of map point points.

The last important baseline is randomly selecting map points from the input map. For a given map point selection approach to be useful, we expect it to compare favourably to random selection using the same number of map points. We label this approach as the "random" approach. Random selection includes the above-mentioned heuristic of including all map points of the last keyframe in the input map.

This section presents the "full map", "empty map" and "random" performance baselines for different map point selection approaches, which are used to interpret the trajectory accuracy of different map point selection approaches. Evaluating the "full map" and "empty" map baselines also serves as a means to evaluate the measurable impact of map point selection for the EuRoC dataset in Section 5.6.

5.5. Evaluating Map Point Selection using the KITTI Dataset

The previous sections present the metrics for evaluating map point selection algorithms, a modified ORB-SLAM 2 algorithm and a test procedure for offline evaluation. This section discusses the KITTI dataset for evaluating map point selection algorithms using the proposed test procedure from Section 5.3. Not all trajectories in this dataset are useful for evaluating map point selection algorithms. This section chooses an appropriate subset of the KITTI trajectories and evaluates the "full map" baseline performance when using these chosen trajectories with the modified ORB-SLAM version from Subsection 5.2.2.

The KITTI dataset features twenty trajectories of a vehicle driving in an urban environment. These trajectories range up to several kilometres per trajectory. Only the first ten trajectories include publicly available ground truth information; therefore, only these trajectories are considered for evaluating map point selection. For trajectories that do not revisit regions of the map, not selecting map points has a minimal effect on trajectory accuracy. In contrast, it is essential to maintain map points for accurate estimation during large loop closures. This work subsequently chooses to use the subset of trajectories from the KITTI dataset that includes loop closures. Since the ORB-SLAM 2 algorithm could not reliably find the loop closures in the 09 trajectory, we do not include this trajectory [8]. After this elimination, the following KITTI trajectories are chosen for evaluation: 00, 02, 05, 06, and 07.

Each of these trajectories is separated into two parts for testing: an initial input trajectory to build the input map, and a test trajectory to test the map after it has been reduced with a selection algorithm. For the choice of where to separate a trajectory into two different sections, we consider that accurate estimation during loop closures involves

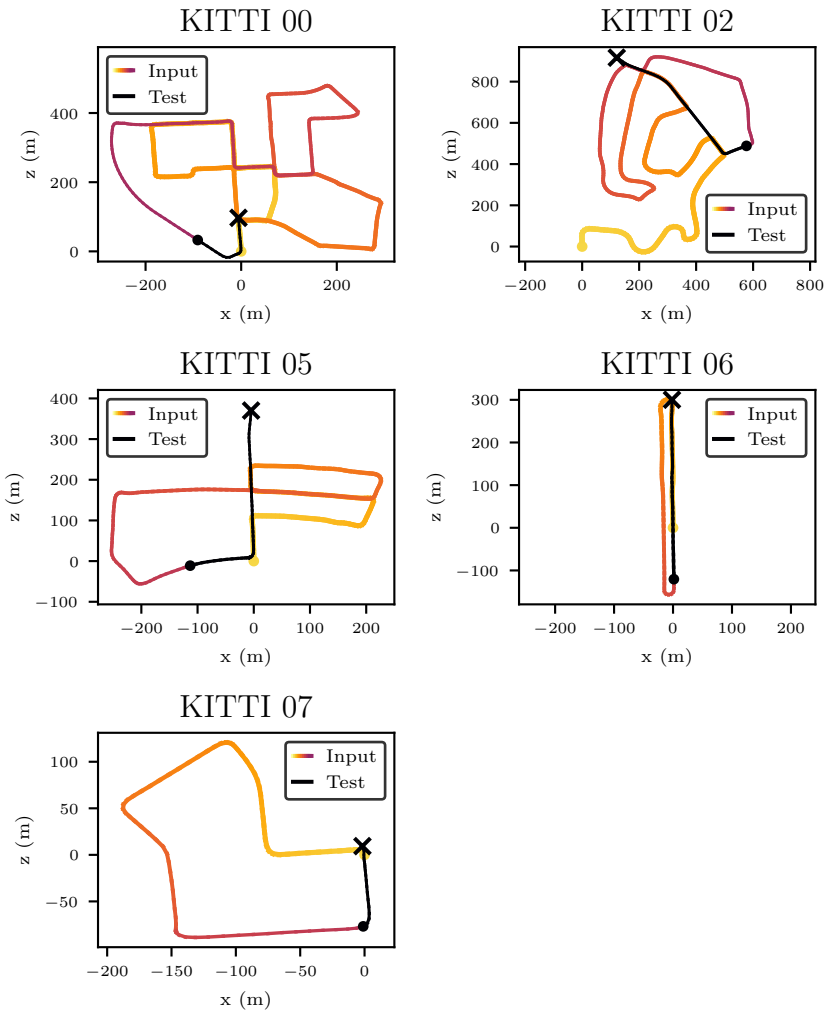

Figure 5.2: The chosen trajectories for the KITTI dataset for evaluating map point selection approaches. The KITTI trajectories labelled 00, 02, 05, 06 and 07 are divided into a test and input trajectory, where the map from the initial input trajectory (visualised with a gradient) is reduced using a map point selection approach and tested on the remaining test trajectory (shown in black). These trajectories are shown using the camera axis of the starting pose of the left camera in the ground truth trajectory.

the full posterior distribution over the full trajectory and map. In addition, drift (the error between the estimated trajectory and the ground truth trajectory) typically increases with time since the last loop closure. We expect the impact of map point to be most challenging before large loop closures when significant drift is present. Additionally, since only the input map is affected by map point selection in this scheme, we also expect it to be more challenging when map points must be chosen for as large of a map as possible. Therefore, we manually choose the frame before a large loop closure to split the two trajectories. Using these guidelines, we choose the input trajectories as the first 4300 frames for the KITTI 00, 4100 for the KITTI 02, 2200 for KITTI 05, 750 for KITTI 06 and 900 for KITTI 07. The resulting input and test trajectories are shown in Figure 5.2.

Various statistics related to the input trajectory and test trajectories are reported in

KITTI	Input Trajectory			Test Trajectory		
Traj.	Frames	Length	Map points	Frames	Length	Loop Closures
00	4300	3510 m	131940	241	211 m	1
02	4100	4363 m	162557	561	702 m	2
05	2200	1731 m	57795	561	472 m	2
06	750	810 m	34075	351	420 m	1
07	900	606 m	23683	201	88 m	1

Table 5.1: KITTI input map and test trajectory statistics. Reported map points are the averages of five maps generated using the same trajectory. The reported lengths are calculated using the cumulative Euclidean distance of the ground truth data for the input and test trajectories.

Table 5.1 to provide an overview of the scale of the trajectories. The length of both the input and test trajectory is listed using the number of frames in the trajectory, and the distance is measured using the ground truth trajectory. The average number of map points for the five input maps generated using these input trajectories are also reported. Lastly, Table 5.1 also includes the total remaining loop closures in the test trajectory.

Due to the stochastic nature of the modified ORB-SLAM 2 algorithm (similar to the original algorithm), maps generated by the algorithm differ between runs. If this variation is significant, it should be considered when evaluating map point selection approaches. The impact of this variation was investigated by generating five different input maps. Each of these input maps was used as input for the modified ORB-SLAM 2 algorithm on the remaining test portion of the trajectory. Figure 5.3 shows relative pose error for ten runs of this experiment for the KITTI 00, 02, 05 and 06 trajectories. For comparison, we also include the trajectory accuracy from using the original ORB-SLAM 2 algorithm.

Figure 5.3 demonstrates that the related pose errors vary significantly between the different input maps. A potential method to mitigate the impact of this variation is to use the same input map when evaluating each map point selection approach. However, the resulting trajectory accuracy might not be a good representative of the performance of the map point selection approach over potential input maps for the trajectory. As a result, this work uses a set of five input maps to test different map point selection approaches.

To characterise the modified ORB-SLAM 2's performance, we evaluate the proposed setup's relative pose error and compare it against the original ORB-SLAM 2 and ORB-SLAM 3 algorithms. These respective algorithms are evaluated on each of the chosen KITTI trajectories. Table 5.2 shows the mean and standard deviation of the relative pose errors. It should be noted that the accuracy of the ORB-SLAM algorithms is hardware-dependent. The results are generated on a personal computer that uses Linux, a Ryzen R9-3900x CPU (with a PassMark score of 32200) and 16 GB RAM.

As shown in the results from Table 5.2, the modified ORB-SLAM 2 algorithm discussed in Subsection 5.2.2 resulted in a reduced standard deviation in trajectory error when

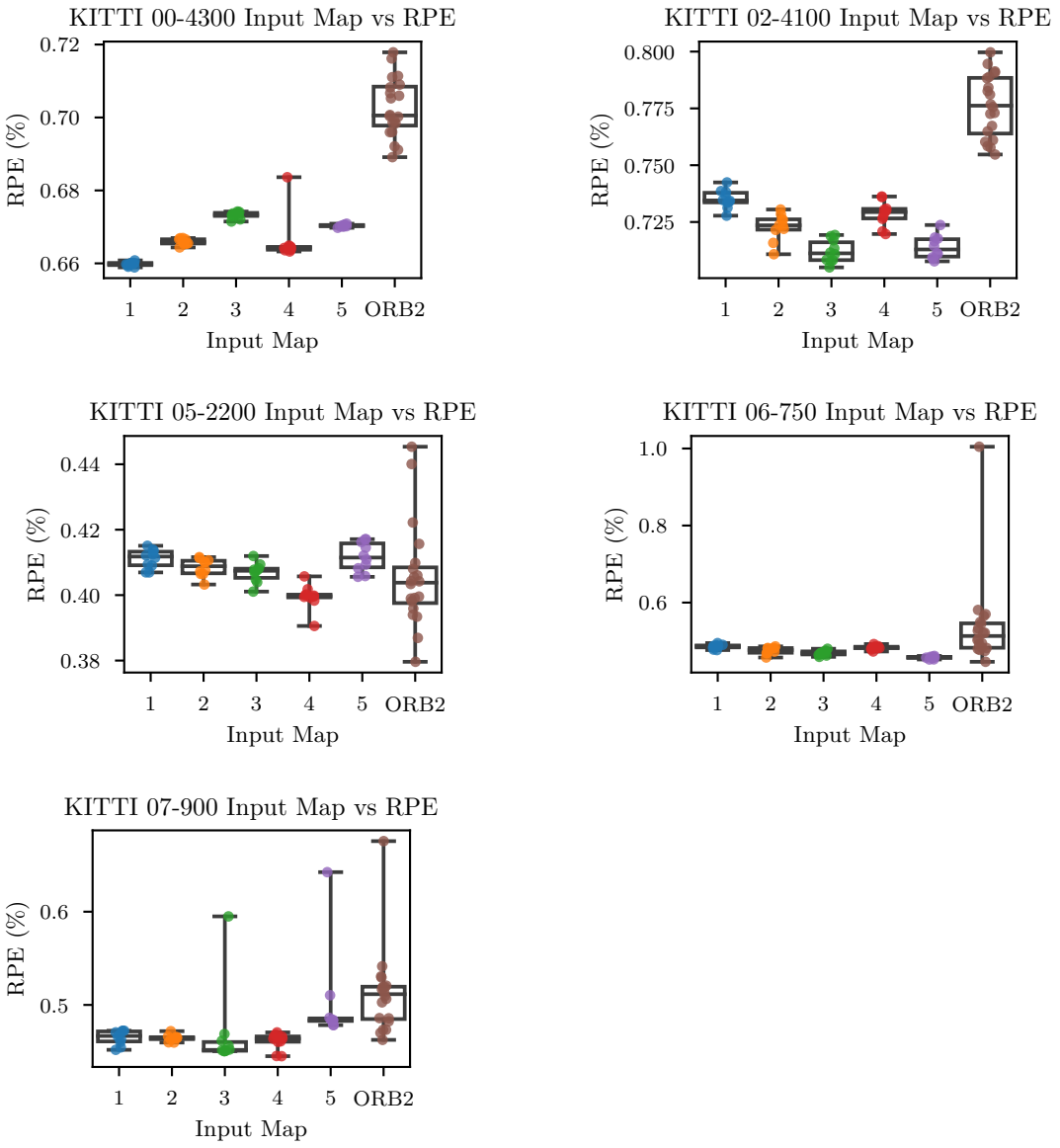

Figure 5.3: This figure shows the variation in trajectory accuracy, measured in relative pose error (RPE) for the whole trajectory with respect to the ground truth trajectory of the modified ORB-SLAM 2 algorithm. Different input maps are generated from different runs of the modified ORB-SLAM 2 algorithm on the same input trajectory (described in Subsection 5.2.2). The modified ORB-SLAM 2 algorithm then uses each input map for the test trajectory, which is repeated ten times. For comparison, this figure includes the variation of the original ORB-SLAM 2 algorithm on the same trajectory.

Trajectory	ORB-SLAM 3	ORB-SLAM 2	Modified ORB-SLAM 2
00	0.717 ± 0.011	0.701 ± 0.008	0.666 ± 0.005
02	0.774 ± 0.016	0.776 ± 0.013	0.723 ± 0.010
05	0.430 ± 0.020	0.404 ± 0.015	0.408 ± 0.005
06	0.538 ± 0.035	0.513 ± 0.113	0.476 ± 0.012
07	0.471 ± 0.036	0.512 ± 0.044	0.465 ± 0.033

Table 5.2: The variation of RPE (%) for different versions of ORB-SLAM on the chosen stereo trajectories of the KITTI dataset. Modifications to the ORB-SLAM 2 algorithm reduced variance at the expense of additional delays. The mean and standard deviation of the RPE are reported for the original ORB-SLAM 2 and ORB-SLAM 3 algorithm, as well as the modified ORB-SLAM 2 version used in this work to evaluate map point selection.

compared to other implementations of ORB-SLAM. The results also show that the modifications improved trajectory accuracy. There are two notable changes that are responsible for improving the trajectory accuracy in Table 5.2. The first change is increasing the number of bundle adjustment iterations. Increasing the number of iterations allows the SLAM estimation to converge more reliably, but this takes longer to compute. The second change is pausing the processing of new camera frames until the bundle adjustment calculation finishes. Normally, ORB-SLAM 2 uses a pose-graph heuristic to approximately correct loop-closures online while calculating the bundle-adjustment estimation problem. This pose-graph heuristic is not well adjusted for maps affected by map point selection. Waiting for the accurate bundle adjustment solution leads to more consistent trajectory accuracy estimates that place a larger emphasis on the bundle adjustment accuracy as opposed to the pose-graph heuristic. While not the motivation behind these changes, these changes also improve the trajectory accuracy for the full map as a consequence at the expense of additional delays in the algorithm. These changes were motivated to improve the ability to more consistently evaluate the impact of map point selection on trajectory accuracy. Despite the improved trajectory accuracy, the additional delays might not be appropriate for all settings. The results further show that ORB-SLAM 3 performs worse than the ORB-SLAM 2 algorithm on the KITTI dataset. ORB-SLAM 3 algorithm modified several heuristics related to loop closures. We suspect that these changes caused the reduced trajectory accuracy for the KITTI dataset.

This subsection chose a subset of the KITTI trajectories to use for the proposed test procedure. The KITTI 00, 02, 05, 06 and 07 trajectories and each of these trajectories were separated into an input and test trajectory. For a map point selection approach to do well for the KITTI trajectories, map point selection must reduce the input maps without negatively impacting either the SLAM estimation or the front-end algorithms. Notably, the selection approach will have to select sufficient map points to continue to identify loop closures and allow accurate bundle adjustment to correct the estimated trajectory.

5.6. Evaluating Map Point Selection using the EuRoC Dataset

Another stereo vision dataset compatible with the ORB-SLAM 2 algorithm is the EuROC dataset [13]. The trajectories in this dataset feature the challenging and dynamic motion of a micro air vehicle in a smaller environment than the KITTI dataset. These trajectories feature one of two environments: either the machine hall or the Vicon room. The different trajectories in these environments feature varying degrees of challenging motion for trajectory estimation. Accurate ground truth is provided using a laser tracker for the machine hall environment, while the Vicon room uses a Vicon motion capture system.

Unlike the KITTI dataset, the trajectories in this dataset often repeatedly cover the same small-scale environment and rely less on the long-term appearance-based loop-closure heuristic. Therefore, we cannot use large-scale loop closures to guide the choice of how to split the trajectory into an input and test trajectory or as a means to eliminate trajectories that are not useful for evaluating map point selection. We instead choose to use the simple scheme of separating the trajectory into two halves for the test and input trajectory.

EuRoC Trajectory	Input Frames	APE (m)		RPE (m)	
		Empty map	Full map	Empty map	Full map
MH01	1841	0.05	0.04	0.56	0.56
MH02	1520	0.04	0.02	0.63	0.63
MH03	1350	0.03	0.03	1.07	1.07
MH04	1017	0.15	0.13	1.00	1.00
MH05	1137	0.13	0.06	0.94	0.94
V101	1456	0.04	0.03	0.33	0.33
V102	885	0.07	0.02	0.78	0.77
V103	1075	0.17	0.04	0.62	0.60
V201	1140	0.07	0.04	0.36	0.36
V202	1174	0.05	0.03	0.71	0.71

Table 5.3: Evaluation of the impact of map point selection on the EuRoC trajectories. Each trajectory in the EuRoC dataset is split into equal input and test portions. The trajectory accuracy of keeping all the map points is compared to discarding all the map points except those in the last keyframe of the input trajectory (labelled "empty map") in the input trajectory. The APE and RPE are reported for the full map and empty map approach.If the empty map performance is close to the full map, then we expect that a trajectory is not sensitive to the map point selection approach.

A test to evaluate the impact of map point selection on trajectory accuracy is to compare the trajectory error of using all the map points in the map to the trajectory error obtained when no map points are selected. We compare the trajectory accuracy of using the "full map" baseline against the "empty map" baseline from Section 5.4. These baselines represent the expected upper and lower bounds of the trajectory accuracy for

different map point selection approaches. A comparison of these baselines across the various EuRoC trajectories shows the measurable impact of map point selection. The absolute and relative trajectory errors are reported for the respective trajectories and map contents in Table 5.3.

The results in Table 5.3 show that the small-scale relative pose error metric over 15 frames, as defined in Section 5.1, is barely affected by map point selection. As a consequence, this dissertation chooses to evaluate the absolute pose error (APE) for the EuRoC dataset. Only some of these trajectories show significant differences between the empty map and full map baselines for the absolute pose error. For many of these input maps, creating a new map during the test trajectory allows for accurate trajectory estimation, and the selection of the map points in the input map often has a minimal impact on the estimated trajectory.

This subsection chooses the trajectories for the EuRoC dataset where the difference between the absolute pose error of the baseline approaches was significant to evaluate map point selection approaches. From the various candidates in Table 5.3, the following trajectories were chosen to use for evaluation: MH05, V102, V103 and V202. These trajectories are also shown visually in Figure 5.4. Due to the heavier emphasis on many small-scale loop closures of the EuRoC dataset and the lack of significant large-scale scale loop closures reliant on the loop-closure detection algorithms, we expect that the chosen trajectories for this dataset will emphasise other components of the SLAM implementation than the chosen trajectories for the KITTI dataset.

	Input Trajectory			Test Trajectory	
Trajectory	Frames	Map points	Length	Length	Frames
MH05	1137	10909	43.4 m	54.0 m	1137
V102	885	6843	40.4 m	35.0 m	826
V103	1075	7271	40.0 m	39.2 m	1075
V202	1174	9869	46.2 m	37.3 m	1175

Table 5.4: EuRoC input map and test trajectory statistics. Reported map points are the averages of five maps generated using the same trajectory. The reported lengths are calculated using the cumulative Euclidean distance of the ground truth data for the input and test trajectories.

Various statistics related to the input trajectory and test trajectory are reported in Table 5.4 to provide a sense of the scale of the trajectories. The length of both the input and test trajectories is listed using the number of frames in the trajectory, and the distance is measured using the ground truth trajectory. Additionally, the average number of map points for the five input maps generated using these input trajectories are also reported.

This chapter detailed our evaluation setup using a SLAM implementation to evaluate different map point selection approaches in Chapter 6. This chapter presented different metrics for evaluating map point selection approaches. This chapter proposed using a

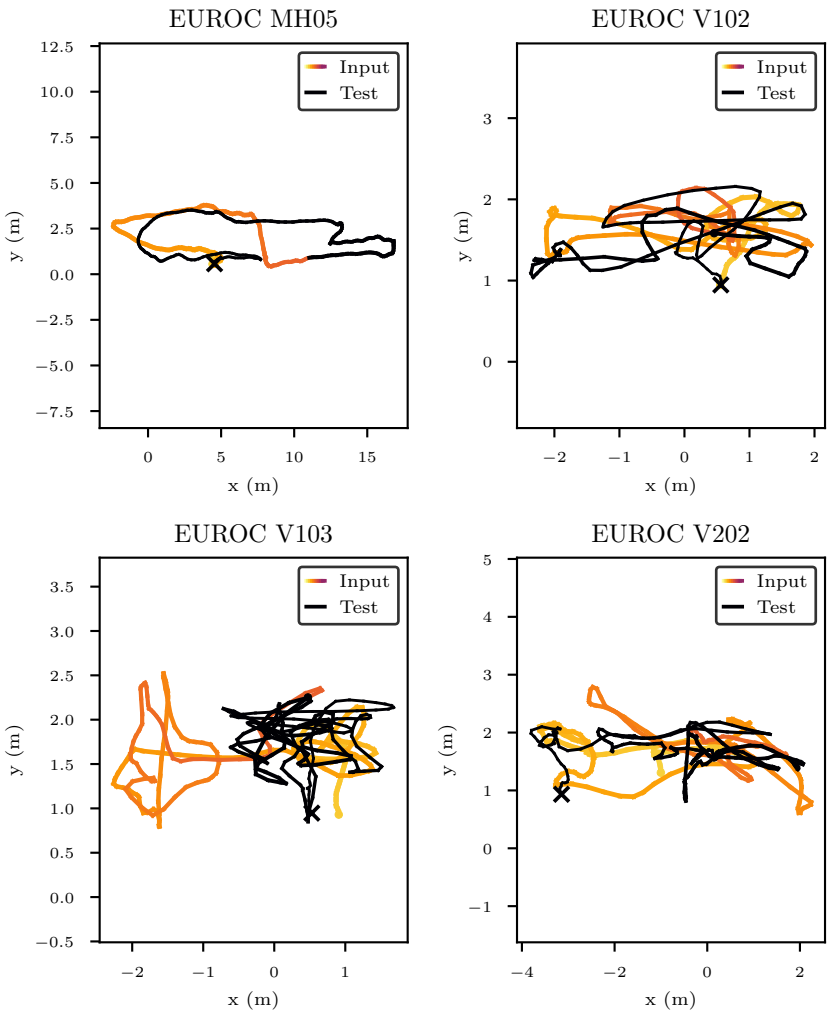

Figure 5.4: The chosen trajectories for the EuRoC dataset for evaluating map point selection approaches. The EuRoC trajectories labelled MH05, V102, V103, V202 are divided into a test and input trajectory, where the map from the initial input trajectory (visualised with a gradient) is reduced using a map point selection approach and tested on the remaining test trajectory (shown in black). These trajectories are shown using the camera axis of the starting pose of the left camera in the ground truth trajectory.

modified version of ORB-SLAM 2, where the modifications were motivated to mitigate some of the undesirable properties of the ORB-SLAM 2 algorithm when evaluating map point selection approaches. An offline test procedure using the modified ORB-SLAM 2 algorithm was also proposed. This test procedure allows fair comparison between different map point selection approaches, including approaches only suitable for offline applications. The test procedure requires choosing input and test trajectories for both the EuRoC and KITTI datasets. Additionally, only a subset of the trajectories used for SLAM are useful for evaluating map point selection approaches. Sections 5.5 and 5.6 chose a subset of trajectories from the respective datasets to evaluate map point selection approaches and detailed the selection of input and test trajectories.

Chapter 6

Offline Experimental Results

This chapter evaluates map point selection with the different utility functions proposed in Chapter 4. This evaluation is performed as described in Chapter 5. The modified ORB-SLAM 2 implementation from Subsection 5.2.2 is used with the offline test procedure from Section 5.3. This chapter evaluates each map point selection approach by reducing the input maps from the input trajectories for either the KITTI or EuRoC datasets, as described in Section 5.5 and Section 5.6, respectively. A selection approach is applied to each of the five input maps with an allocated map point budget. Each reduced map resulting from the selection is then evaluated twice on the remaining trajectory. This results in 10 samples for each trajectory per approach at a tested map point budget. For the random selection baseline, we generate five copies for each input map for a specified map point budget. Evaluating each of these maps twice for the random selection baseline generates 50 samples per map point budget.

The results in this dissertation are generated on a personal computer that uses Linux Mint 21.1, a Ryzen R9-3900x desktop CPU (with a PassMark score of 32200) and 16 GB RAM. The timing results for all selection approaches are measured using a single thread. The computer hardware used not only affects the measured execution time but also trajectory accuracy due to the design of ORB-SLAM 2. Therefore, the trajectory accuracy and timing results should be interpreted by comparing the relative timing or accuracy of different map point selection approaches.

The experimental results in this chapter are divided into three sections, each comparing different groups of map point selection approaches. The first two sections include comparisons against approaches investigated but not included in the final evaluation due to the superior performance of other map point selection approaches.

Section 6.1 evaluates the SLAM utility from Section 4.1 with either lazy-greedy or stochastic greedy and random selection on the KITTI dataset. This experiment has two goals: First, this experiment establishes a reference for the trajectory accuracy and recall performance achievable by the offline approach of using the SLAM utility and the random baseline. These serve as a point of reference for other experiments for interpreting the achieved trajectory accuracy and recall performance. The second goal of the experiment in this section is to evaluate the use of the stochastic greedy algorithm to reduce the computation times of using the SLAM utility compared to the lazy greedy greedy algorithm.

Section 6.2 evaluates the coverage-based map point selection approaches from Section 4.3 using the KITTI dataset against the trajectory accuracy and loop-closure recall of select results from Section 6.1. The goal of this section is to evaluate the different coverage-based approaches presented in this dissertation and choose the best approach for comparison in Section 6.3.

Section 6.3 evaluates the remaining approximate information-theoretic approaches from Section 4.2 and the combined approach from Section 4.4, which assumes loop-closure information is available. These approaches are compared against selected approaches from Section 6.1 and Section 6.2. The evaluation in this section evaluates different map point selection approaches for both the KITTI and EuRoC datasets.

6.1. Evaluation of SLAM Approaches and Random Selection

This section evaluates the SLAM utility from Section 4.1 using either the lazy or stochastic greedy algorithm (described in Section 3.2.2) and the random selection baseline. The approach using the SLAM utility and the lazy greedy algorithm is labelled *lg-SLAM*, while using the stochastic greedy algorithm is labelled *sg-SLAM*. The stochastic greedy algorithm is used with an approximation factor of $\epsilon = 0.05$, which determines the size of the random subsets used for greedy selection and the resulting performance guarantee of the approximate utility (as shown in Subsection 3.2.2). Random selection is one of the performance baselines from Subsection 5.4 and is labelled *random*. The evaluation in this section uses only the KITTI dataset. This section first compares the execution times of different map point selection approaches in Subsection 6.1.1 before evaluating the trajectory accuracy of these approaches in Subsection 6.1.2.

6.1.1. Timing Results for SLAM approaches and Random on the KITTI dataset

This subsection evaluates the computational cost of map point approaches by measuring the average execution times required to reduce each input map for the KITTI dataset to 15% of its original size. Section 5.5 describes the selection of the input trajectories and the creation of the input maps for the KITTI dataset. The average execution time of five runs of using the SLAM utility with either the lazy greedy algorithm or the stochastic greedy algorithm is shown in Figure 6.1. As a point of comparison, we also include the execution time required for the random selection baseline. The timing results from Figure 6.1 are also included in tabular format in Table 6.1.

When evaluating the execution time of these different approaches, we only measure the

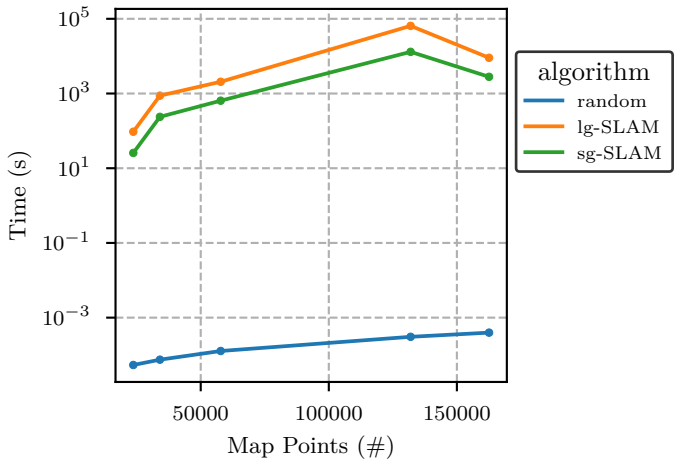

Figure 6.1: Execution times for maximising the SLAM utility using either the lazy greedy algorithm (labelled *lg-SLAM*) or the stochastic greed algorithm (labelled *sg-SLAM*). Random selection is also included for comparison. Input maps are generated using the input trajectory and reduced to 15% of the input map points using the respective map point selection approaches. The reported execution times are the average of five runs of the approach.

Algorithm	KITTI Trajectory 00	02	05	06	07
random	3.07e-04 s	3.97e-04 s	1.28e-04 s	7.50e-05 s	5.40e-05 s
lg-SLAM	17 h 58 min	2 h 30 min	34 min 23 s	14 min 37 s	1 min 34 s
sg-SLAM	3 h 37 min	46 min 39 s	10 min 42 s	3 min 57 s	25.66 s

Table 6.1: Execution times for maximising the SLAM utility using either the lazy greedy algorithm (labelled *lg-SLAM*) or the stochastic greedy algorithm (labelled *sg-SLAM*). Random selection is also included for comparison. Input maps are generated using the input trajectory and reduced to 15% of the input map points using the respective map point selection approaches. The reported execution times are the average of five runs of the approach.

time the algorithm uses to select map points and any pre-computation directly associated with the map point selection approach. For example, we do not include the time spent reading or writing the map to storage or creating relevant SLAM-related data structures (such as storing the observations associated with each camera frame or map point). Time spent on calculating data not typically stored by SLAM implementations is included despite this not being part of the time spent on executing the lazy or stochastic greedy algorithms. For example, our implementation of the SLAM utility first calculates and stores the marginal information matrix over robot poses associated with each map point, which is included in the measurement.

The timing results from Figure 6.1 and Table 6.1 show that both the *lg-SLAM* and *sg-SLAM* approaches are expensive and scale poorly to larger settings. The execution time for shorter trajectories often takes at least multiple minutes and increases to multiple hours for the KITTI 00 trajectory. The implementation used in this dissertation implements

the SLAM utility using a sparse implementation of the Cholesky update. The worst-case computational complexity of evaluating the utility is $O(t^2)$, and optimising the utility with the lazy or stochastic greedy algorithm results in asymptotic complexities of $O(n^2t^2)$ and $O(n^2t)$, respectively, where t is the number of keyframes in the trajectory and n is the number of map points (refer to Subsection 4.1.3 for these complexities).

Figure 6.1 shows that the execution time for the SLAM utility does not monotonically increase with the number of map points. The last two data points in this graph are for the KITTI 00 and KITTI 02 trajectories, with 132 940 and 162 557 map points in the input map on average, respectively (refer to Table 5.1 for these average map points). Maximising the SLAM utility was more expensive for the map from the KITTI 00 trajectory than for the KITTI 02 trajectory despite this map having fewer map points. However, the computational cost of evaluating the marginal gain for the SLAM utility depends on the density of the posterior information matrix over robot poses. The input map for the KITTI 02 trajectory does not include any loop closures, while the KITTI 00 trajectory includes several loop closures. These loop closures result in a denser information matrix for the KITTI 00 trajectory, and as a consequence, the execution time for the KITTI 00 trajectory is longer. This increased execution time for the KITTI 00 trajectory is despite the resulting input map containing fewer map points and keyframes than the KITTI 02 trajectory.

The SLAM utility using either stochastic or lazy greedy algorithms quickly becomes prohibitively expensive for online use as both the size of the map and the density of the information matrix increases. The multiple hours required for selection is also a limiting factor for offline evaluation of these map point selection approaches as this dissertation compares the trajectory accuracy of different map point selection approaches against varying map point budgets. The test procedure described in Section 5.3 requires evaluating map point selection algorithms over five input maps over a range of map point budgets. For the maps from the KITTI 00 trajectory, generating the reduced maps can require multiple days of running map point selection algorithms.

A convenient property of the lazy greedy algorithm is that after running the algorithm, the selected set of map points can be used to inexpensively obtain the selected set of map points for smaller budgets. If we store the order in which the lazy greedy algorithm selects map points, the solution for smaller map point budgets is the initial portion of the selected map points up to the newer and smaller budget. As a consequence, using the lazy greedy algorithm only for the largest map point budget also yields all the selected map points for smaller budgets. This dissertation uses this property with the *lg-SLAM* approach to avoid running the lazy greedy algorithm from scratch for each map point budget in the tested range. The stochastic greedy algorithm does not have this property and requires calculating a selection of map points for each map point budget in the range, which is often more computationally expensive than calculating the lazy greedy solution once for

the largest budget.

This subsection shows that using the SLAM utility with either the lazy-greedy or stochastic greedy algorithm is limited to offline applications. This subsection also shows that while the *sg-SLAM* approach is less computationally expensive than the *lg-SLAM* approach, the execution times for these approaches are in the same order of magnitude. For comparison, this subsection also evaluated the computation time required for random selection that requires less than 1 ms to calculate. Random selection and the lazy-greedy algorithm represent the two extremes regarding execution time.

6.1.2. Trajectory Accuracy for SLAM approaches and Random on the KITTI dataset

This subsection evaluates the trajectory accuracy of the *lg-SLAM* and *sg-SLAM* approaches against random selection (labelled *random*) and the *full map* and *empty map* baselines from Section 5.4. The discussion in this subsection is structured as follows: First, the range of chosen map point budgets for the KITTI dataset is presented. Second, this subsection analyses the random selection baseline as this provides insight into the difficulty of different trajectories for map point selection. At the end of this subsection, the trajectory accuracy and recall performance of the *sg-SLAM* and *lg-SLAM* approaches are compared.

As part of the test procedure from Section 5.3, different map point selection approaches are applied to the same set of input maps over varying map point budgets. The map point budgets for evaluation of the KITTI dataset are varied between 3000 and 20000 map points as this was found to capture most of the significant behaviour of the different selection approaches (with the exception of KITTI 02, for which we instead varied the map points between 6000 and 40000 map points). These map point budgets for the input map can be expressed as percentages of the average map points as follows: from 2.3% to 15.2% for the KITTI 00 trajectory, from 3.7% to 24.6% for the KITTI 02 trajectory, from 5.2% to 34.6% for the KITTI 05 trajectory, from 8.8% to 58.7% for the KITTI 06 trajectory and lastly, from 12.7% to 84.4% for the KITTI 07 trajectory.

Evaluating map point selection approaches using the chosen procedure and range of map point budgets is a time-consuming process. The modified ORB-SLAM 2 algorithm for the KITTI dataset (5 trajectories) to evaluate a map point selection approach generates 450 reduced maps (5 per trajectory at 18 map point budgets). Each reduced input map at a specified map point budget is evaluated twice on the test trajectory. Evaluating all the reduced maps takes roughly 17 hours of continually running the modified ORB-SLAM 2 algorithm. The different test trajectories vary in length, but on average, evaluating a test trajectory takes roughly 2 minutes per input map. The time spent on evaluation also does not account for the computation time required to calculate the different map point selections. Due to the time-consuming nature of evaluating map point selection approaches

over a wide range of trajectories and map point budgets, the number of input maps and range of map budgets was chosen as a reasonable compromise between the time required to generate these results and the accuracy of capturing trajectory and recall performance.

Using the test setup mentioned above, the reduced maps from the different map point selection approaches; that is, *lg-SLAM*, *sg-SLAM*, and *random* are evaluated on the remaining trajectories. The trajectory accuracy of each reduced map on the remaining test trajectory is evaluated using the ground truth trajectory. The median of the relative pose error, as well as the first and third quantiles of the relative pose error, are shown in Figure 6.8 and Figure 6.9. These figures also include the average recall for the respective map point selection approaches as this measures the impact of map point selection on loop-closure detection. This figure also includes a comparison against the *empty map* and *full map* baselines. These baselines are not tied to a specific budget, but the performance of these baselines is plotted as a reference for the relative performance of map point selection approaches at a specified budget. Map point budgets are stated as a percentage of the original map points in the input map.

The trajectory accuracy of the random selection indicates the relative difficulty of selecting map points for that trajectory. As shown in Figures 6.2 and 6.3, the map point budget as a fraction of the input map required by random selection to have a trajectory accuracy comparable to the *full map* baseline varies between trajectories. The map point budget for achieving good recall for random selection varies between 15% for the KITTI 00 trajectory and 40% for the KITTI 07 trajectory. This variation in the performance of the random selection baseline suggests that the difficulty of map point selection varies between these two trajectories.

These figures show the importance of measuring loop-closure performance when interpreting the trajectory accuracy results from map point selection approaches. Random selection often has trajectory accuracy close to the *full map* baseline when provided with enough map points to obtain good recall in a given trajectory. Conversely, the random selection approach has a trajectory accuracy comparable to the *empty map* baseline when no loop closures were found. This comparable trajectory accuracy results from ORB-SLAM 2 only updating trajectory estimates for the full trajectory and map points after loop closures. As a result, when random selection fails to achieve good recall at a given budget, it performs poorly. For the KITTI 07 trajectory, 100% recall is only achieved at around 40% of the input map for the KITTI 07 trajectory. The loop closure for the KITTI 07 trajectory is particularly challenging for map point selection approaches due to the minimal overlap between frames in the input portion of the trajectory and when revisiting the area during testing.

The *lg-SLAM* approach, where the SLAM utility is maximised using the lazy greedy algorithm, achieves trajectory accuracy comparable to the *full map* baseline in the KITTI 00 and 02 trajectories for map point budgets larger than 10% and consistently outperforms

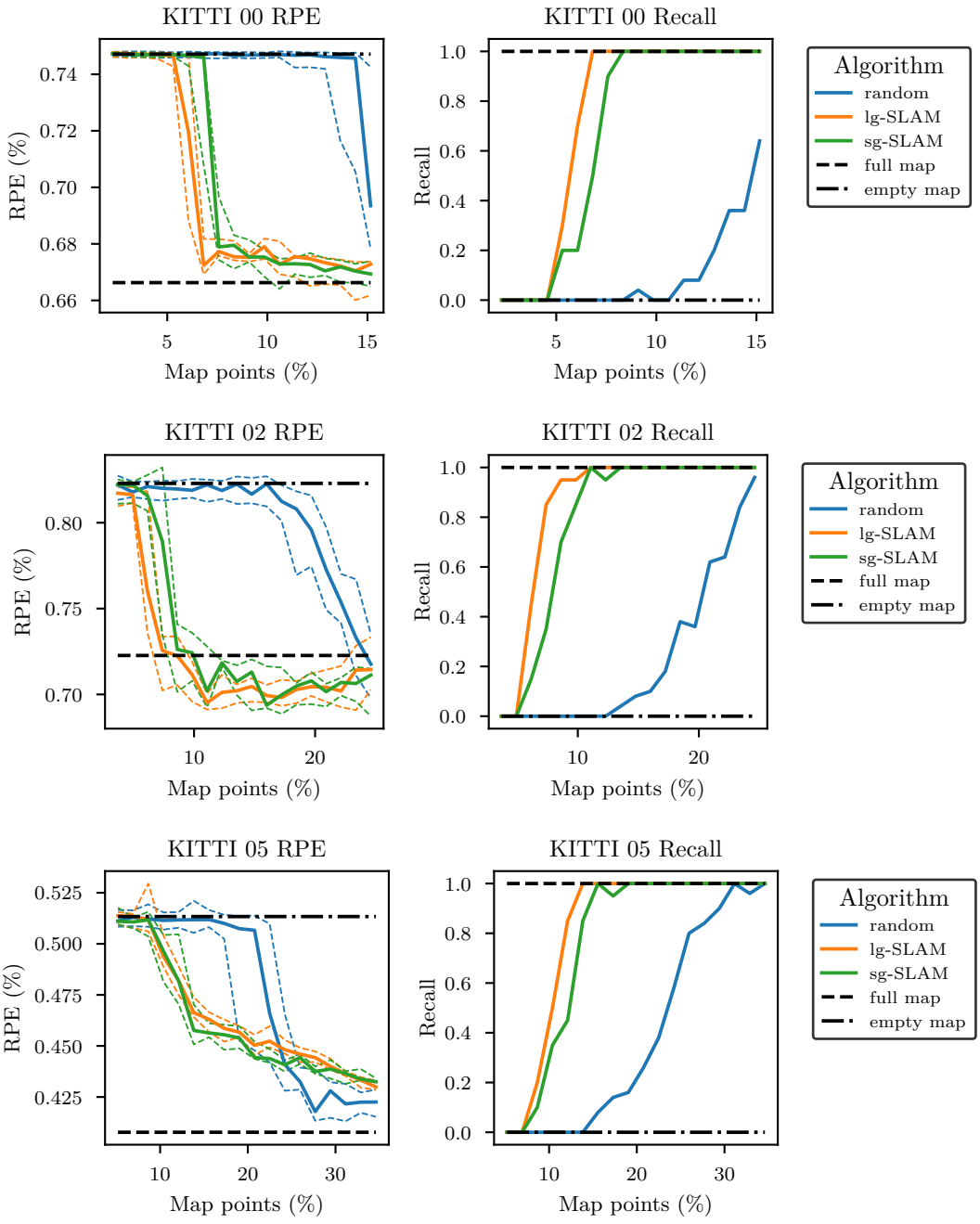

Figure 6.2: Relative pose error (RPE) and recall for the KITTI 00, 02 and 05 trajectories as a function of the map point budget. Proposed approaches are compared against the *empty map*, *full map* and *random* baselines.

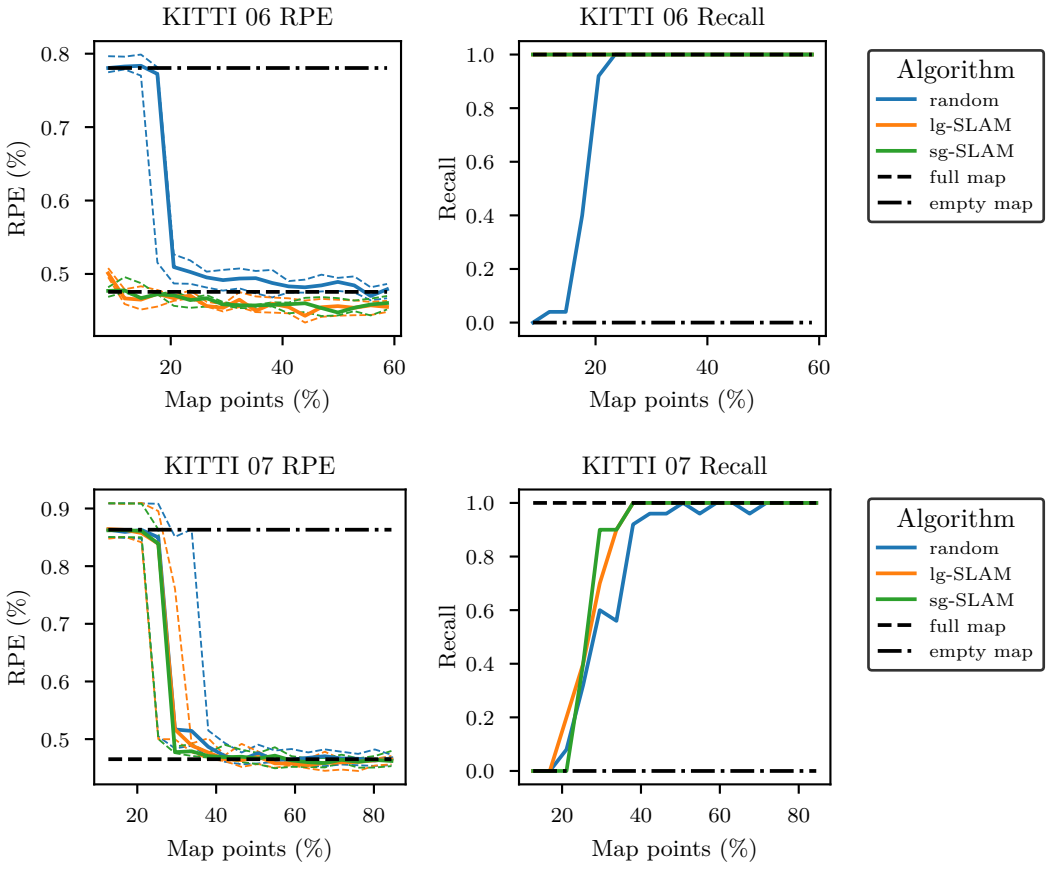

Figure 6.3: Relative pose error (RPE) and recall for the KITTI 06 and 07 trajectories as a function of the map point budget. Proposed approaches are compared against the *empty map*, *full map* and *random* baselines.

random selection. For the KITTI 05 trajectory, the *lg-SLAM* approach outperforms random selection for the range between 10% and 25% map points, but random selection achieves a higher trajectory accuracy for larger maps. The *lg-SLAM* approach has a trajectory accuracy comparable to the *full map* baseline for the entire range of map point budgets for the KITTI 06 trajectory. For the KITTI 07 trajectory, the *lg-SLAM* approach is comparable to the random selection. This comparable performance to random for the 05 and 07 trajectories indicates that this utility function did not capture all aspects of the map point utility accurately. Recall performance explains the poor trajectory accuracy for the KITTI 07 trajectory. However, for the KITTI 05 trajectory, the trajectory accuracy is less than the *full map* baseline despite 100% recall. This suggests another underlying cause for the trajectory accuracy of this approach for the KITTI 05 trajectory.

The SLAM utility makes two important simplifying assumptions that are potentially violated in practice: First, it assumes that the information gain using only current and past measurements is a good approximation of the information gain of the posterior distribution that includes future measurements. This assumption will not hold for all trajectories. The second assumption is that the chosen model of the posterior distribution is an accurate model for the impact of map points on trajectory accuracy. Various practical effects, for example, linearisation errors, missed loop-closures or outliers, result in the baseline SLAM approach not accurately modelling the utility of map points in all cases. As a result, approaches based on the SLAM utility will not necessarily always outperform alternatives in practice.

As indicated in Figures 6.2 and 6.3, the trajectory accuracy of using the stochastic greedy algorithm with the SLAM utility, or *sg-SLAM*, was similar to using the lazy greedy algorithm with the SLAM utility, or *lg-SLAM*. Given the comparable trajectory accuracy, this is a stochastic approximate alternative to the lazy greedy algorithm. The stochastic greedy algorithm requires multiple runs to characterise the trajectory accuracy of the map point selection approach. The stochastic nature makes this approach less practical for an offline baseline despite its improved execution time. The stochastic greedy also does not have the property of cheaply calculating the selection for smaller map point budgets using the solution of large map point budgets.

This section compared map point selection using the SLAM utility function with either the lazy or stochastic greedy algorithms against the random selection baseline. The results showed that the budget of map points needed for random selection to match the *full map* baseline varies between the different datasets. Therefore, some trajectories are more challenging for map point selection approaches than others. The SLAM utility remains expensive to use even if optimised with the stochastic greedy algorithm. This dissertation consequently uses the lazy greedy algorithm as using a deterministic map point selection approach simplifies the comparison against other map point selection approaches. This work also uses the property of the lazy greedy algorithm mentioned in Subsection 6.1.1 to

cheaply calculate the set of map points for lower budgets for the SLAM utility, using the selected set for higher map point budgets.

6.2. Evaluation of Coverage-based Approaches

This section experimentally evaluates the coverage-based approaches mentioned in Section 4.3 on the KITTI dataset. Subsection 4.3.1 details an existing coverage-based integer program proposed by Dymczyk *et al.* [14], which has yet to be used for SLAM. This dissertation evaluates its applicability to the SLAM problem. This utility is optimised using the commercial Gurobi [64] optimiser, and we label this approach *ip-wcover*. Initial experimentation showed that the coverage parameter $b_{\text{cover}} = 100$ and trade-off parameter $\lambda = 25$ work well for the datasets in this evaluation (refer to Subsection 4.3.1 for a more detailed description of these parameters). Subsection 4.3.2 proposes maximising an equivalent utility function using greedy techniques, and maximising this weighted coverage utility with the lazy greedy algorithm is labelled *lg-wcover* with the same parameters. As an alternative to these coverage-based approaches, Subsection 4.3.3 proposed maximising the minimum number of map points in a frame using the SATURATE algorithm, and we label this approach *sat-cover*.

This section is structured as follows: The computational cost of these approaches is compared in Subsection 6.2.1. Subsection 6.2.2 evaluates the trajectory accuracy of the coverage-based approaches and compares it against relevant baselines from Section 5.4 and the *lg-SLAM* approach from Section 4.1.

6.2.1. Timing Results for Coverage-based Approaches on the KITTI dataset

The computational cost of the coverage-based approaches is evaluated by measuring the average execution time required to select 15% of the total map points in each of the KITTI trajectories. The average of five runs of the algorithm is recorded in Table 6.2 and shown visually against the average number of map points in each input map in Figure 6.4.

	KITTI Trajectory				
Algorithm	00	02	05	06	07
ip-wcover	1.02 s	1.09 s	0.34 s	0.28 s	0.14 s
lg-wcover	0.04 s	0.04 s	0.01 s	0.01 s	0.01 s
sat-cover	0.97 s	1.02 s	0.32 s	0.20 s	0.12 s

Table 6.2: Execution times of coverage-based map point selection approaches for the KITTI trajectories. Input maps are generated using the input trajectory and reduced to 15% map points of the input map using the respective map point selection approaches. The reported execution times are the average of five runs of the approach.

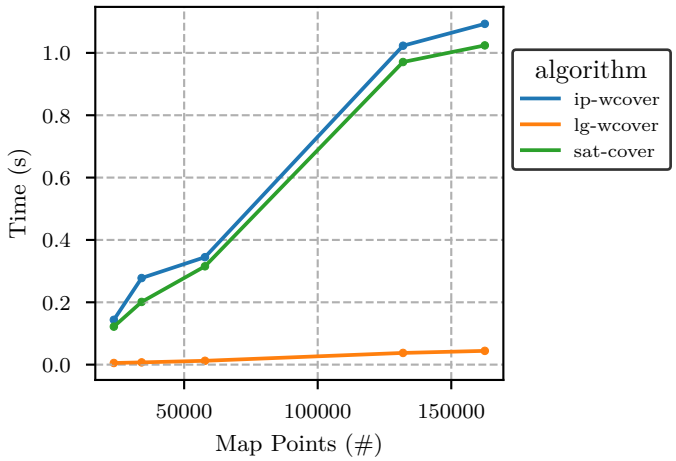

Figure 6.4: Execution times for coverage-based map point selection approaches on the KITTI trajectories. Maps are generated using the first input portion of the trajectories and reduced to 15% map points using the respective map point selection approaches. The reported execution times are the average of five runs of the approach.

Table 6.2 shows that all of the coverage-based approaches have execution times in the order of seconds or even lower. These execution times are orders of magnitude less than the minutes or hours needed for approaches based on the SLAM utility from Section 6.1.

Execution times for the *ip-wcover* approach varies between 0.14 seconds in the smaller KITTI 07 trajectory to 1.09 seconds for the KITTI 02 trajectory, respectively. Integer programming is NP-hard and, therefore, has a non-polynomial complexity in general. We expect that for larger problems, this approach would be outperformed by approaches with polynomial-time complexity. However, for these problems, execution times remain inexpensive. Using the lazy greedy algorithm to maximise the same utility improves execution times significantly, with the *lg-wcover* approach generally being two orders of magnitude faster and requiring only 44.2 ms for the largest KITTI 02 dataset.

As described in Subsection 4.3.3, the *sat-cover* approach requires multiple iterations to maximise the minimum number of map points in a frame. Since this approach has a polynomial computational complexity, we expect that this approach should outperform the integer-programming-based approach for problems of a sufficient scale. However, the execution times of these two coverage-based approaches are comparable for these datasets.

6.2.2. Trajectory Accuracy for Coverage-based Approaches on the KITTI dataset

This subsection evaluates the trajectory accuracy and loop closure performance of the coverage-based techniques on the KITTI dataset. The resulting median relative pose error (RPE) and loop-closure recall are shown in Figure 6.5 and 6.6. For ease of comparison, these figures also include the trajectory accuracy results for using the SLAM utility with

the lazy greedy algorithm and random selection from Subsection 6.1.

The integer program model by Dymczyk *et al.* [14] labelled *ip-wcover* achieves good recall for the KITTI trajectories, as shown in Figures 6.5 and 6.6. This approach was proposed to select map points for localisation and visual odometry and continue to provide good recall on this dataset. The trajectory accuracy results for the *ip-wcover* approach is surprisingly competitive for the KITTI 00 and 02 trajectories at map point budgets between 10% and 15%, as depicted in Figure 6.5. However, this is not the case when this approach is applied over a wider range of map point budgets for the KITTI 00 and KITTI 02 trajectories. Additionally, this approach also has poor trajectory accuracy for the KITTI 05 as shown in Figure 6.5 and KITTI 06 and KITTI 07 in Figure 6.6. Despite its good recall performance in these trajectories, the *ip-wcover* approach performs poorly regarding trajectory accuracy, often performing worse than random selection or even has a worse trajectory accuracy than the *empty map* baseline. The ORB-SLAM 2 algorithm only performs bundle adjustment; that is, SLAM estimation with the full map and trajectory when a loop is detected. For these cases where the accuracy is worse than the *empty map* baseline, the estimate of the trajectory before performing the loop closure was more accurate than the updated estimate from bundle adjustment with the selected subset of map points. The *ip-wcover* approach also often fails to deliver competitive trajectory accuracy at higher budgets and rarely approach the accuracy of the *full map* baseline. This approach is often outperformed by random selection at the map point budgets at the top end of the tested range, as seen in the KITTI 02, 05, 06 and 07 trajectories. While the *ip-wcover* approach is capable of detecting loop closures at smaller map point budgets than Section 6.1, this approach does not consistently allow for the accurate correction of the map resulting from those loop closures. This trajectory accuracy is in contrast to the *lg-SLAM* approach and random selection, for which the trajectory accuracy is often limited by recall performance.

The *lg-wcover* approach uses the same utility function as the *ip-wcover* approach, but optimises it using the lazy greedy algorithm. The trajectory accuracy and recall of this approach compare well to using integer programming techniques. The *lg-wcover* approach results in the same excellent recall performance when compared to the random selection baseline or the *lg-SLAM* approach. The trajectory accuracy of the *lg-wcover* approach is largely identical to that of the *ip-wcover* approach is often outperformed by random selection and consistently outperformed by the *lg-SLAM* approach. These results demonstrate that the *lg-wcover* approach serves as an inexpensive approximation of the original *ip-cover* with comparable performance for both trajectory accuracy and loop-closure recall.

The *sat-cover* approach generally outperforms other coverage-based approaches in terms of SLAM trajectory accuracy. The *sat-cover* approach uses the SATURATE algorithm, which uses an iterative search to maximise the minimum number of map points. This

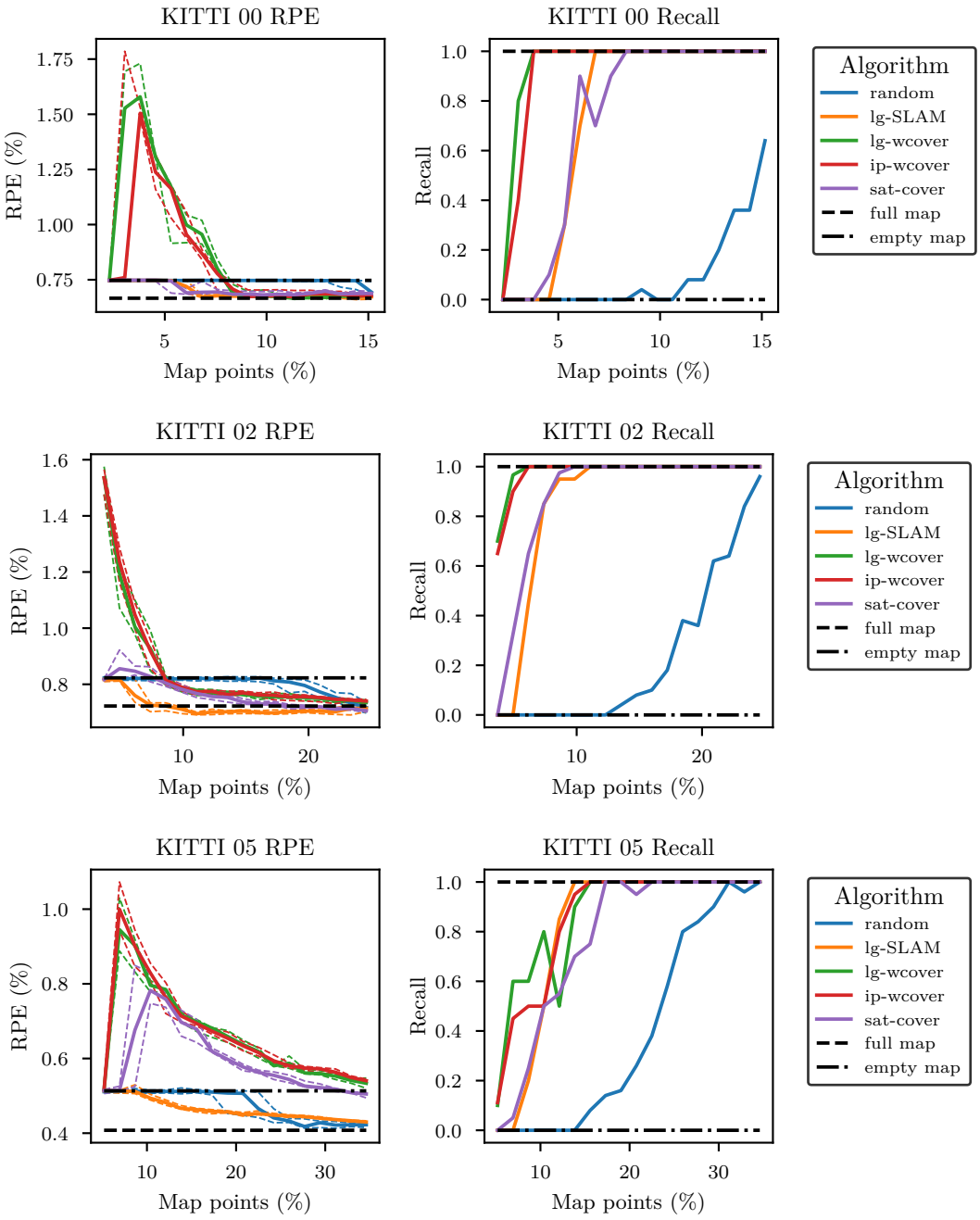

Figure 6.5: RPE and recall for the KITTI 00, 02 and 05 trajectories as a function of the map point budget. Coverage-based approaches are compared against the *empty map*, *full map* and *random* baselines.

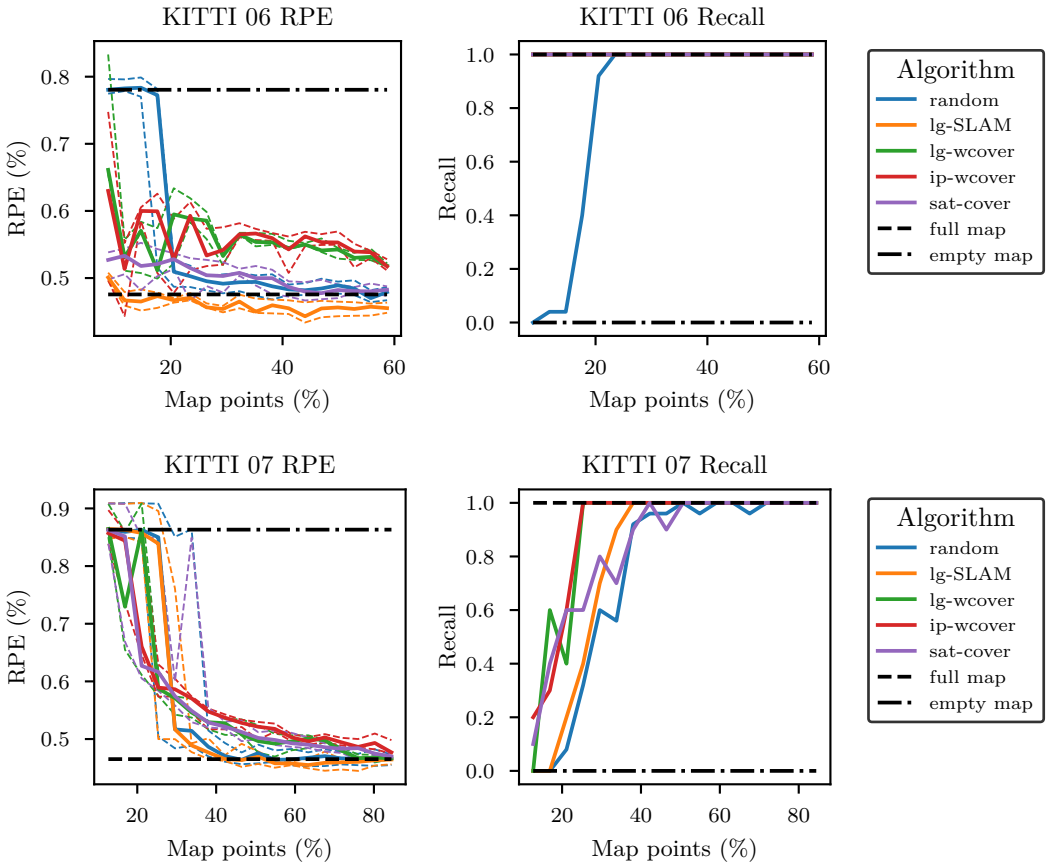

Figure 6.6: RPE and recall for the KITTI 06 and 07 trajectories as a function of the map point budget. Coverage-based approaches are compared against the empty map, full map and random baselines. All approaches except random selection achieved 100% recall for the KITTI 06 trajectory.

approach is less competitive in terms of loop-closure performance when compared to the *ip-wcover* and *lg-wcover* approaches. While the *sat-cover* approach obtains better trajectory accuracies for the KITTI trajectories, the trajectory accuracy of this approach still compares poorly to the *lg-SLAM* approach and the random selection baseline.

The *sat-cover* approach provides further insight into the selection of the *lg-SLAM* approach. While the SATURATE algorithm uses an iterative search to select a uniform selection of map points, the *lg-SLAM* approach instead uses the SLAM utility function that assigns lower utility values to redundant map points to encourage selecting a diverse set of map points. It is noteworthy that the *lg-SLAM* and *sat-cover* approaches have comparable recall. This demonstrates that *lg-SLAM* approach already selects a reasonably diverse set of map points for loop closure recall and is only outperformed by the two best coverage-based approaches in terms of recall (*ip-wcover* and *lg-wcover*).

The results in this section show that the *ip-* and *lg-wcover* approaches obtained excellent recall and are inexpensive, but these approaches did not allow for accurate SLAM estimation. The results from this subsection suggest that coverage-based approaches are reasonable approximations to model the utility of map points for loop-closure recall. This observation motivates the combined approach developed in Section 4.4. The *lg-wcover* approach serves as an inexpensive approximation of the *ip-wcover* approach. While these approaches are not suitable for capturing the map point utility for the overall SLAM implementation, coverage-based approaches serve as a point of comparison for loop-closure performance. Therefore, this dissertation chooses to include the *ip-cover* approach as a representative coverage-based approach for comparison.

6.3. Evaluation of Approximate Information-theoretic and Combined Map Point Selection Approaches

This section evaluates different the approximate information-theoretic map point selection approaches on both the KITTI and EuRoC datasets. The localisation utility from Subsection 4.2.1 is maximised using the lazy greedy algorithm and is labelled *lg-local*. The odometry utility from Subsection 4.2.2 is maximised using the lazy greedy algorithm and labelled *lg-odom*.

Sections 6.1 and 6.2 showed that the *lg-SLAM* approach obtain the best trajectory accuracy of the different map point selection approaches thus far. Consequently, this approach is included in this section to serve as a point of comparison for trajectory accuracy. Section 6.2 shows that the existing weighted coverage-based approach from Subsection 4.3.1 obtains good recall performance for the KITTI trajectories, and this map point selection approach is included in this section to serve as a point of comparison for loop-closure recall. This approach is labelled *ip-wcover* and uses the same parameters $b_{\text{cover}} = 100$ and

$\lambda = 25$ as in Section 6.2.

For the KITTI trajectories that feature large-scale loop closures, we also test the impact of extending the odometry utility with additional loop-closure information as proposed in Section 4.4. Initial experimentation showed that $b_{\text{cover}} = 250$ works well for the combined approach. This combined approach maximised with the lazy greedy algorithm is labelled *lg-odom+cover*. This combined approach is aimed at improving trajectory estimation accuracy for datasets with large-scale loop closures. Due to the lack of large-scale loop closures in the EuRoC dataset, we do not evaluate the performance of the combined approach on the EuRoC dataset.

The evaluation of the aforementioned map point selection approaches proceeds as follows: First, the execution times are compared for both the KITTI and EuROC datasets in Subsection 6.3.1. Secondly, the trajectory accuracy of different map point selection approaches for the KITTI dataset is evaluated in Subsection 6.3.2. Lastly, the trajectory accuracy of different approaches is compared for the EuRoC dataset in Subsection 6.3.3.

6.3.1. Timing Results for KITTI and EuRoC Datasets

This subsection evaluates the execution times of the different map point selection approaches on the chosen trajectories of the KITTI and EuRoC datasets. Each input map is reduced to 15% of the initial map points using the respective map point selection approaches. The average execution times for five runs of each map point selection approach for both the EuRoC and KITTI datasets are shown in Figure 6.7. The execution times are also shown separately for the EuRoC dataset in Table 6.4 and for the KITTI dataset in Table 6.3.

This subsection also includes the timing results for using the stochastic greedy algorithm with either the localisation utility or the odometry utility. We label these approaches as *sg-local* and *sg-odom* and use the stochastic greedy algorithm with an approximation factor of $\epsilon = 0.05$. These results show the potential reduction in execution times obtained by using the stochastic greedy algorithm.

	KITTI Trajectory				
Algorithm	00	02	05	06	07
ip-wcover	1.02 s	1.09 s	0.34 s	0.28 s	0.14 s
lg-local	1.00 s	1.14 s	0.38 s	0.23 s	0.13 s
lg-odom	0.84 s	0.95 s	0.33 s	0.20 s	0.11 s
lg-odom+cover	0.80 s	0.91 s	0.31 s	0.19 s	0.10 s
sg-local	0.27 s	0.31 s	0.10 s	0.06 s	0.03 s
sg-odom	0.27 s	0.28 s	0.10 s	0.07 s	0.03 s

Table 6.3: Execution times for different map point selection approaches for the KITTI trajectories. Input maps are generated using the input trajectory and reduced to 15% map points of the input map using the respective map point selection approaches. The reported execution times are the average of five runs of the approach.

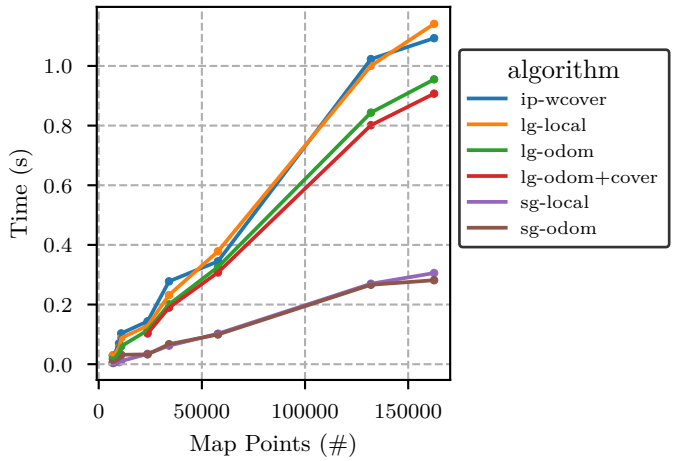

Figure 6.7: Execution times for online map point selection approaches versus the total number of map points in the input map. Listed execution times are for selecting 15% of the total number of map points for several trajectories in both the EuRoC and KITTI datasets.

	EuRoC Trajectory			
Algorithm	MH05	V102	V103	V202
ip-wcover	0.10 s	0.03 s	0.03 s	0.07 s
lg-local	0.09 s	0.03 s	0.03 s	0.06 s
lg-odom	0.06 s	0.02 s	0.02 s	0.04 s
sg-local	0.01 s	0.00 s	0.00 s	0.01 s
sg-odom	0.03 s	0.01 s	0.01 s	0.02 s

Table 6.4: Execution times for different map point selection approaches for the EuRoC trajectories. Input maps are generated using the input trajectory and reduced to 15% map points of the input map using the respective map point selection approaches. The reported execution times are the average of five runs of the approach.

Figure 6.7 shows that the execution times for many approaches are typically below 1 second. Execution times for many of the approaches increase roughly linearly with increasing map size at varying rates. This increase is despite the varying worst-case asymptotic computational complexities of these approaches. As noted in Subsection 3.2.2, the stochastic greedy algorithm requires $O(n)$ function evaluations, while the lazy greedy algorithm requires $O(n^2)$. These observations align with the empirical observation that using the lazy-greedy algorithm often results in significantly reduced execution times despite its poor worst-case computational complexity.

Given that the odometry and localisation utilities are already suitable for online use even when using the lazy greedy algorithm, we continued to use the lazy greedy algorithm for deterministic selection and stronger performance guarantees. This is despite the reduced execution times of using the stochastic greedy algorithm with these utility functions (as shown in *sg-odom* and *sg-local* versus *lg-odom* and *lg-local*, respectively. When using the odometry or localisation utilities in larger environments with limited computing, it might

be necessary to use the stochastic greedy algorithm and adjust the degree of approximation, ϵ, based on the processing time available.

Execution times for the *lg-odom* and *lg-local* approaches using the lazy greedy algorithm were largely comparable. The *lg-odom* approach was slightly faster, with a maximum average execution time of 0.95 s for the KITTI 02 dataset. In comparison, the *lg-local* approach required 1.14 s for this trajectory. The combined *lg-odom+cover* approach from Section 4.4 adds an additional loop-closure coverage term to the definition of the odometry utility. Execution times for this combined approach were largely comparable to that of *lg-odom*, where lazy greedy was used on the utility without this additional term.

All approaches in this subsection have execution times in the order of a second or less. These execution times are far shorter than the typical time required to perform SLAM estimation or traverse environments of this scale. These relatively short execution times show that these methods are viable for online use for SLAM maps of this scale. The following subsections evaluate the resulting trajectory accuracy and, therefore, the quality of the maps produced by different approaches.

6.3.2. Trajectory Accuracy for the KITTI Dataset

This subsection evaluates the trajectory accuracy of the different map point selection approaches. The results for the SLAM approach and random selection from Section 6.1 are included for comparison, as well as the results from the *ip-wcover* approach from Section 6.2.

The median of the relative pose error, as well as the first and third quantiles, are shown in Figure 6.8 and Figure 6.9. These figures also include the average recall for the respective map point selection approaches. The *empty map* and *full map* baselines are not tied to a specific budget, but the performance of these baselines is plotted as a reference for the relative performance of map point selection approaches at a specified budget.

A criterion by which to evaluate different map point selection approaches is the number of map points needed to achieve trajectory accuracy comparable to the *full map* baseline. Figures 6.8 show that the *lg-local* approach needed more map points than the *lg-SLAM* approach to achieve trajectory accuracy comparable to the *full map* baseline. As shown in Figure 6.9, the *lg-local* approach offered better trajectory accuracy than the *lg-SLAM* approach in the KITTI 05 trajectory and slightly better than the trajectory accuracy of the full map in the KITTI 06 trajectory. Similar to the SLAM approach, the *lg-local* approach does worse for the KITTI 07 trajectory, with performance comparable to the random selection. For these trajectories, using the localisation utility is an approximation of the SLAM utility, often offering slightly worse trajectory accuracy at a significantly reduced computational cost.

The *lg-odom* approach offers comparable trajectory accuracy to the *lg-SLAM* approach

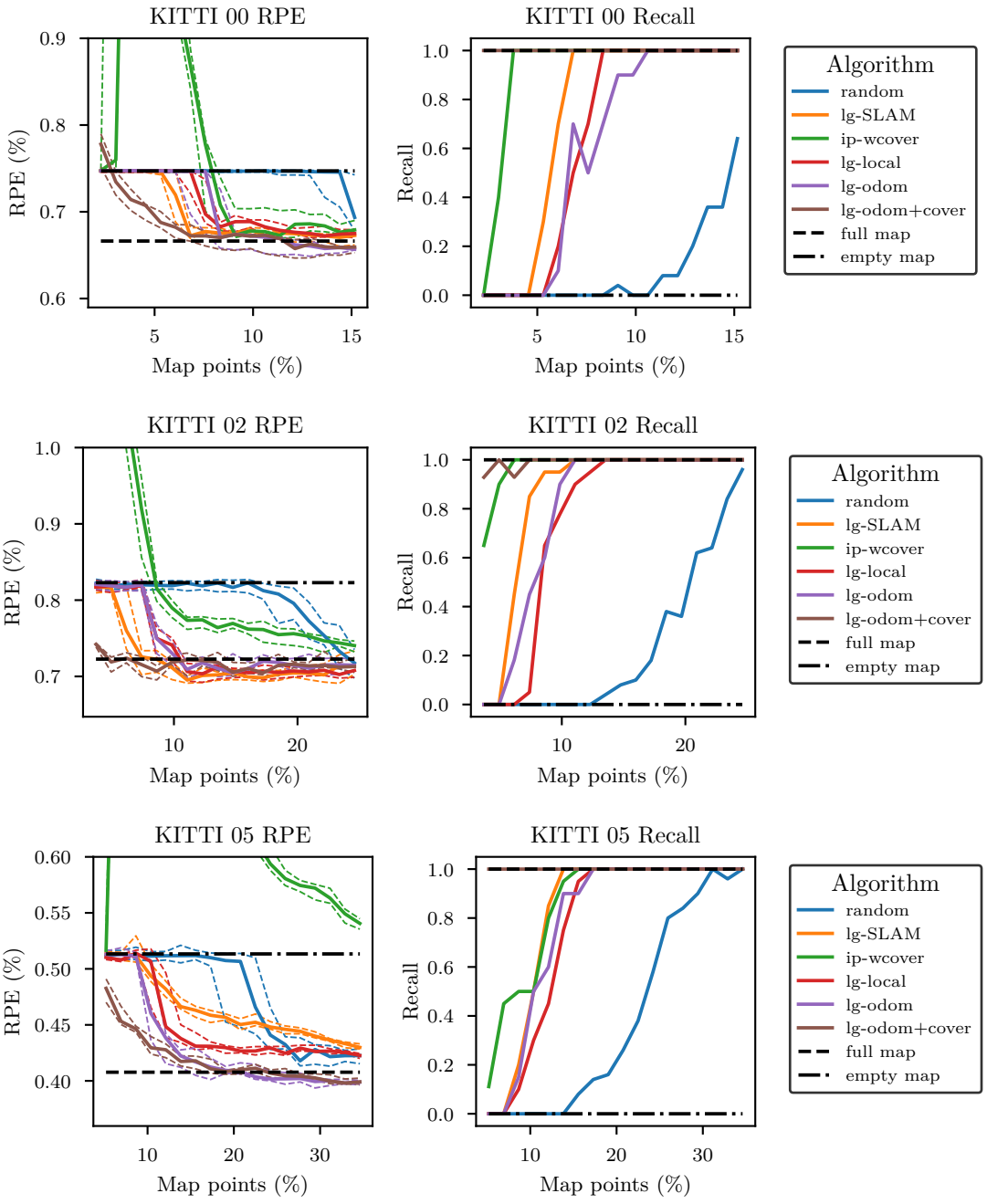

Figure 6.8: RPE and recall for the KITTI 00, 02 and 05 trajectories as a function of the map point budget. Proposed approaches are compared against the *empty map*, *full map* and *random* baselines and alternative integer program approach labelled *ip-wcover* [14].

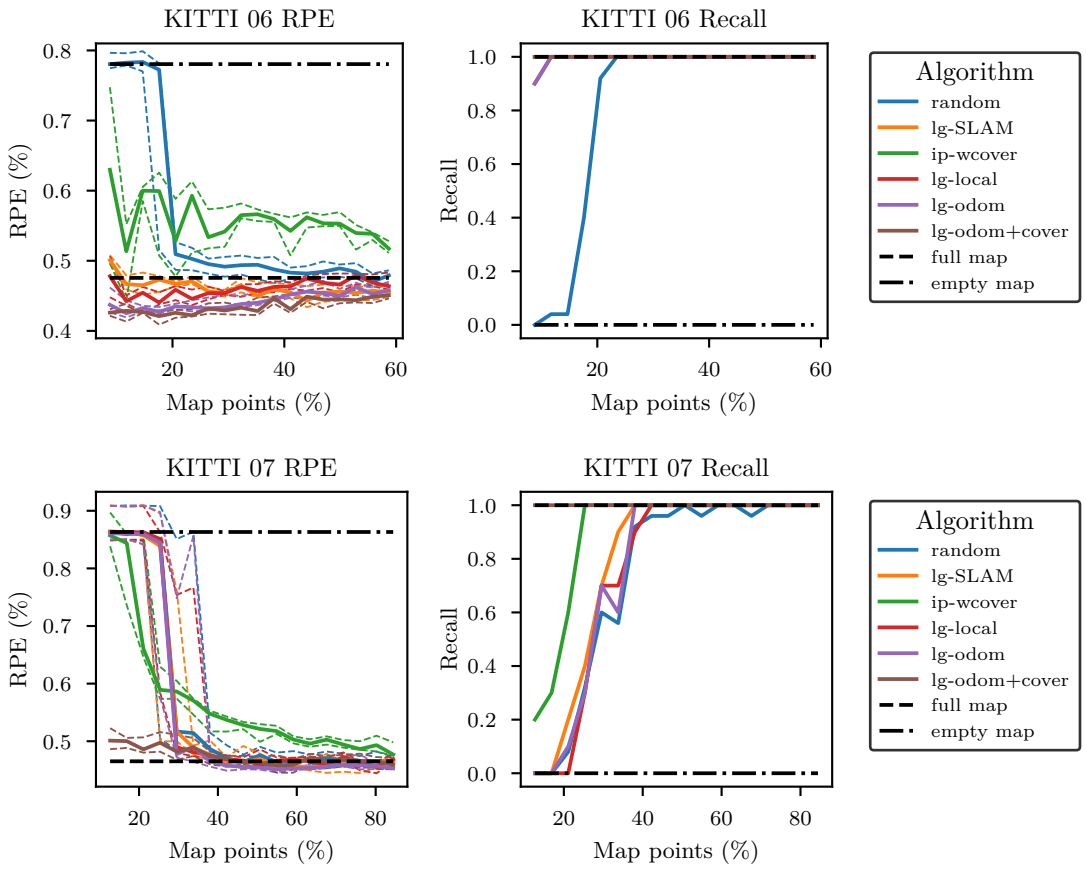

Figure 6.9: RPE and recall for the KITTI 06 and 07 trajectories as a function of the map point budget. Proposed approaches are compared against the empty map, full map and random baselines and alternative integer program approach labelled *ip-wcover* [14]. All approaches achieved 100% recall for the KITTI 06 trajectory, except the *random* and *lg-odom* approaches.

at higher map point budgets in the KITTI 00 trajectory. However, it is outperformed by the *lg-SLAM* and *lg-local* approaches at a small range of map point budgets between 5% and 10%. The worse trajectory accuracy of the *lg-odom* approach for the KITTI 00 trajectory coincides with worse recall performance at the aforementioned range of map point budgets. This trend continues for the KITTI 02 trajectory, where using the *lg-odom* approach requires larger maps than the *lg-SLAM* approach to achieve trajectory accuracies comparable to the full map. The *lg-odom* approach consistently outperformed the *lg-local* and *lg-SLAM* approaches and the random selection baseline for the KITTI 05 trajectory. The *lg-odom* approach also outperforms the previously mentioned approaches for the KITTI 06 dataset, where it does notably better than the *full map* baseline at lower budgets. Lastly, this approach performs similarly to *lg-SLAM* or *lg-local* approaches and random selection for the KITTI 07 trajectory, where poor loop-closure performance limited trajectory accuracy. For the KITTI dataset, the *lg-odom* approach offers improved trajectory accuracy compared to the *lg-local* at a similar computational cost. The *lg-odom* approach is, therefore, the recommended approximate approach between these two options.

The trajectory accuracy for the *lg-odom* approach is very comparable to the *lg-SLAM* approach in terms of trajectory accuracy, except for map point budgets where recall performance limits the trajectory accuracy.

If we assume knowledge of the loop-closure locations, Figures 6.8 and 6.9 show that the *lg-odom+cover* approach improves trajectory accuracy compared to the *lg-odom* approach for smaller map point budgets where trajectory accuracy is limited by loop-closure recall. Note that since the KITTI 06 trajectory already achieved good recall performance over the tested range of map point budgets, the combined approach provides no advantage of the *lg-odom* approach. The *lg-odom+cover* approach outperforms alternatives both in terms of trajectory accuracy and recall for the KITTI 00, 02, 05 and 07 trajectories. For larger map point budgets, this approach obtains comparable trajectory accuracy to the *lg-odom* approach. The *lg-odom+cover* approach is the only approach that could consistently close loops for the KITTI 07 trajectory at map point budgets as low as 15%. The trajectory accuracy and loop-closure recall of the *lg-odom+cover* approach demonstrate that considering loop-closure detection during map point selection can offer significantly improved map point selection performance.

The results in Figures 6.8 and 6.9 show that the information-theoretic approaches offer superior trajectory accuracy compared to alternatives. However, in some cases, loop-closure performance plays a significant role in the performance of a map point selection approach. For these cases, the *lg-odom+cover* approach outperforms all alternatives in both trajectory accuracy and recall performance.

Appendix B presents two supporting investigations for these results. The first investigation is a qualitative visualisation of the RPE where the *lg-odom* approach outperformed the *lg-SLAM* approach in terms of trajectory accuracy. The first investigation presents a visual representation of the RPE for the KITTI 05 trajectory at a map point budget of 20% where the *lg-odom* approach has a trajectory comparable to the full map. This investigation also visualises the errors for the KITTI 06 trajectory with a map point budget of 12% where the *lg-odom* approach outperforms the full map baseline. This first investigation qualitatively shows that the *lg-odom* improved trajectory errors throughout the entire trajectory. The second investigation evaluates the choice to evaluate map point selection approaches using an input trajectory before large loop-closures using the KITTI 05 trajectory. This second investigation shows that the chosen 2200 frames are not necessarily the choices for the most significant separation between map point selection approaches if the map point budget scales with the size of the input map. However, the choice of 2200 frames for the KITTI 05 trajectory allows for reasonable separation between approaches, and the trajectory accuracy of the information-theoretic approaches is insensitive to the choice of evaluation frame.

Figures 6.10 and 6.11 compare using the relative pose error (RPE) against the absolute pose error (APE) for the KITTI trajectories. These figures provide a comparison between

using these two metrics for the KITTI dataset and for ease of comparison, we do not include recall in these figures. Relative pose error for the KITTI dataset is evaluated on distances between 100 and 800 meters as described in Section 5.1 and recommended for the KITTI dataset [12].

Figures 6.10 and 6.11 show that the relative performance of map point selection approaches is largely similar, whether using the large-scale RPE or APE. APE appear to be slightly less sensitive to map point selection approaches, with many approaches having trajectory accuracies closer to the *full map* baseline. Map point selection approaches that managed to obtain errors close the full map using RPE continue to do so when using APE.

This subsection evaluated different map point selection approaches using the KITTI dataset. This subsection found that the *lg-odom* and *lg-odom+cover* offer trajectory accuracy comparable to the offline *lg-SLAM* approach. When loop-closure information is available, the combined *lg-odom+cover* approach outperforms all alternatives. These observations are supported by both the RPE results and the APE trajectory accuracy results.

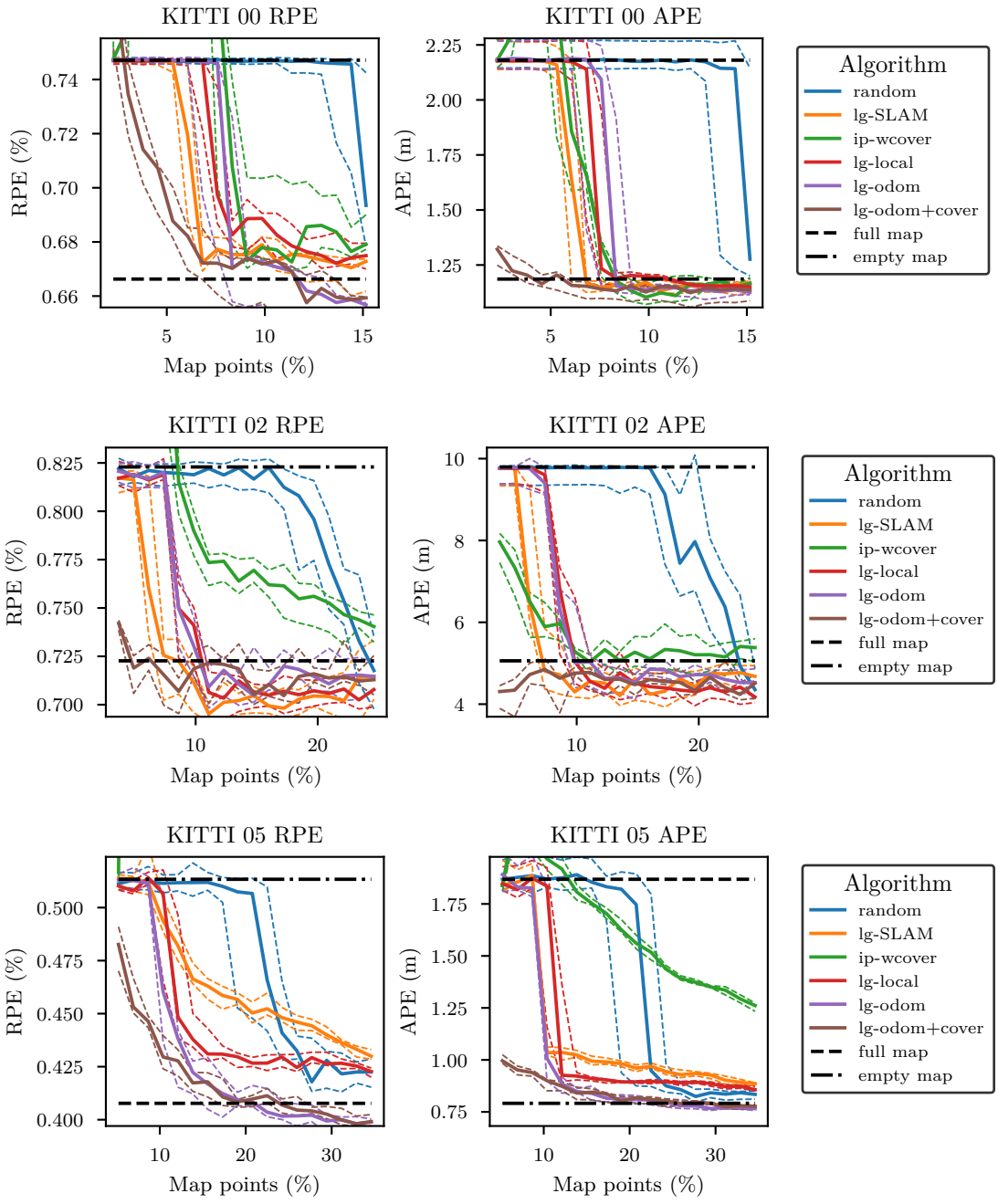

Figure 6.10: RPE and APE for the KITTI 00, 02 and 05 trajectories as a function of the map point budget. Proposed approaches are compared against the *empty map*, *full map* and *random* baselines and alternative integer program approach labelled *ip-wcover*. [14].

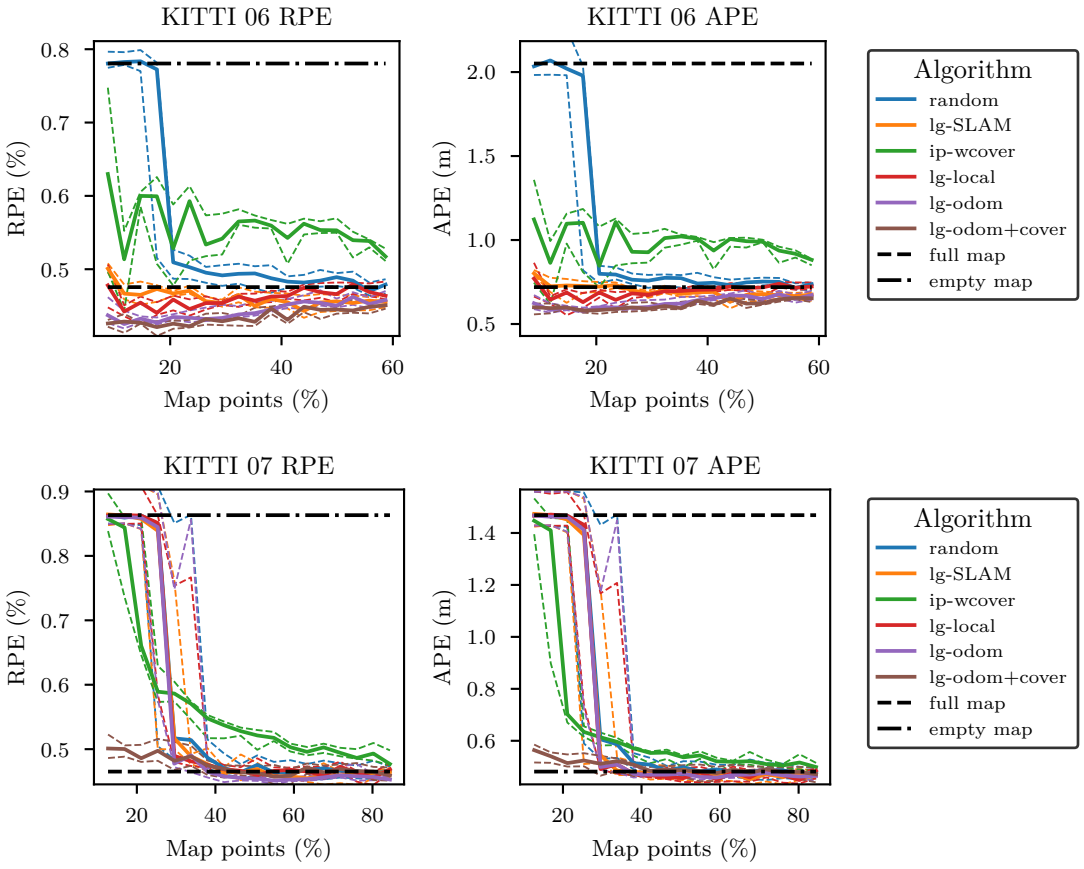

Figure 6.11: RPE and APE for the KITTI 06 and 07 trajectories as a function of the map point budget. Proposed approaches are compared against the *empty map*, *full map* and *random* baselines and alternative integer program approach labelled *ip-wcover*. [14].

6.3.3. Trajectory Accuracy for the EuRoC Dataset

This subsection evaluates the performance of the different map point selection approaches on the EuRoC dataset that features dynamic motion and numerous shorter-term loop closures in room-sized environments.

Section 5.6 evaluated the difference between the *empty map* and *full map* baselines and found that the smaller scale relative pose error definition typically used for the EuRoC datasets (error over 15-frames) was typically not significantly affected by map point selection. The KITTI metric of evaluating over 100 to 800 meters is also not appropriate due to the shorter length of these trajectories. As recommended by Section 5.6, this subsection evaluates the absolute pose error (APE) for the EuRoC dataset. The APE error is commonly used to measure the accuracy of SLAM problems for this dataset. Due to the lack of explicit large-scale loop closures, we do not evaluate recall performance or evaluate using the *lg-odom+cover* approach from Section 4.4 on this dataset.

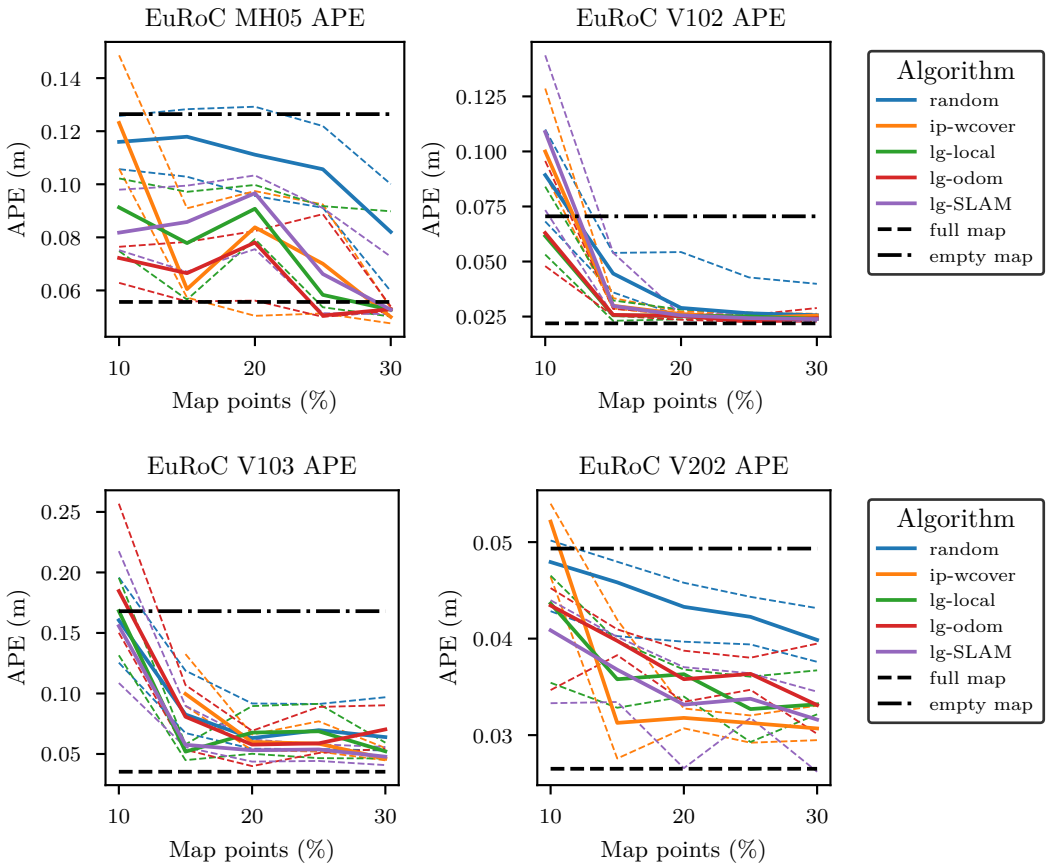

Figure 6.12: Evaluation of map point selection approaches on the chosen EuRoC trajectories. The median and the first and third quantiles of the absolute pose error (APE) are shown over the tested range of map point budgets. The EuRoC dataset does not feature similar large-scale loop closures as the KITTI dataset; therefore, we do not report recall for this dataset.

Similar to the KITTI dataset, we evaluate different map point selection approaches by

first generating different input maps using an initial input portion of the trajectory. The different map point selection approaches are applied to these input maps and the reduced input maps are tested with the modified ORB-SLAM on the remaining test trajectory. The estimated trajectory resulting from using these reduced input maps are then compared to the ground truth trajectory. Similar to the KITTI dataset, each input trajectory is used to create five input maps. These five input maps are evaluated twice. The reported trajectory errors at a listed map point budget are, therefore, for 10 samples, with the notable exception of random selection, where this process is repeated five times for 50 samples. For each map point selection approach at a listed map point budget, we report the median APE of the trajectory, and the first and third quantile for each of the chosen EuRoC trajectories in Figure 6.12.

Experimental results from Figure 6.12 show that most of the tested map point selection approaches outperformed random selection over the tested range of map point budgets. However, the difference between trajectory accuracy of the different map point selection approaches, that is, *ip-wcover*, *lg-SLAM*, *lg-odom* and *lg-odom*, are often within the upper and lower bounds of the trajectory accuracies. This suggests that there is no approach that consistently outperforms alternatives.

For the EuRoC trajectories, particularly the V103 and MH05 trajectories in the 10% to 20% range, trajectory accuracy improves in a less consistent fashion with the number of selected map points than previously observed for the KITTI dataset. We suspect that due to the lack of large-scale loop closures, the effect of other components of the SLAM implementation (such as tracking or mapping) more significantly impacts the trajectory accuracy. These effects are unmodelled by the utility function, and the trajectory accuracy of the overall SLAM implementation is not necessarily monotone with increasing map points. If, for example, a very limited number of map points was selected in the region, the tracking module will insert more keyframes due to the limited number of tracked map points, which could potentially result in improved accuracy. Similarly, selecting map points influences the co-visibility graph used for local SLAM, affecting which keyframes are included in the local SLAM estimation.

The trajectories in the EuRoC dataset traverse the same environment multiple times and, unlike the KITTI dataset, do not have a limited number of large-scale loop closures. For the KITTI dataset, maintaining a good set of map points from the input map was essential since these map points must be estimated with sufficient accuracy to achieve accurate loop closure in the future. The EuRoC dataset features many small loops through the same environment. New map points can be added during this test trajectory that assists in accurate estimation during the many loops that follow, and as a result, the EuRoC dataset is less sensitive to the map point selection approach when evaluated with the proposed test procedure.

Any of the tested map point selection approaches can reduce the map for the EuRoC

dataset by 30% of the map, resulting in a decrease in trajectory accuracy of only 2 cm in the tested scenarios. Considering the accuracy and execution times, all the approaches except SLAM perform similarly for the EuRoC dataset. These results demonstrate that these map point selection approaches perform well in scenarios other than the KITTI dataset. However, not all datasets or trajectories are useful for comparing different map point selection approaches.

This chapter showed that approximate information-theoretic approaches have trajectory accuracy comparable to the *lg-SLAM* approach in both the KITTI and EuROC datasets while having execution times comparable to the *ip-wcover* approach. The trajectory accuracy of these approximate information-theoretic approaches is consistently comparable to or better than alternatives and is computationally inexpensive.

Chapter 7

Online Evaluation of Map Point Selection Approaches

This chapter evaluates different map point selection approaches that have been implemented in an online configuration alongside the original ORB-SLAM 2 algorithm. The purpose of these experiments is to demonstrate that the map point selection approaches that have previously been tested offline can be used in online scenarios. This chapter also shows that the changes made to facilitate the evaluation of different map point selection approaches in Subsection 5.2.2 are not necessary for map point selection approaches to obtain trajectory accuracies comparable to not using map point selection. This chapter does not detail an optimised integration of these techniques into ORB-SLAM 2. Instead, this experiment demonstrates how map point selection will be used alongside a SLAM implementation.

Previously, Chapter 5 proposed evaluating different map point selection approaches offline and this approach was used to generate the results presented in Chapter 6. This offline evaluation was motivated by two reasons: to enable the comparison between techniques unsuitable for online applications and to minimise the effect of properties of the specific SLAM implementation from impacting the evaluation of these techniques. This chapter instead evaluates different map point selection approaches online using an implementation of map point selection alongside the original ORB-SLAM 2. Map point selection is implemented in a separate thread, and only approaches suitable for online evaluation are included in this chapter. This chapter, therefore, does not evaluate the approaches based on the SLAM utility. This chapter first presents the software implementation in Section 7.1, while Section 7.2 presents the experimental results of using this implementation.

7.1. Online Map Point Selection Implementation

This section describes the changes to the original ORB-SLAM 2 implementation (not the modified implementation from Chapter 5) with map point selection used for online selection. Similar to offline evaluation, we use ORB-SLAM 2 as a representative algorithm for evaluating map point selection. ORB-SLAM 2 separates the online SLAM problem into

threads that address different sub-problems of the online SLAM implementation. When introducing map point selection, it is necessary to consider how to integrate it into the SLAM implementation and the data flow between the respective threads.

The execution times required for map point selection impact how these map point selection approaches can be implemented alongside the original ORB-SLAM 2 algorithm. The approximate information-theoretic approaches require execution times of the order of a second for larger maps, as shown in Section 6.3, and these execution times are in the same order as the other components within the ORB-SLAM 2 implementation. Therefore, while these execution times are practical for an online application, they are not negligible in terms of the existing components. As mentioned in Subsection 5.2.1, the accuracy of ORB-SLAM 2 is dependent on the timing of different threads. Introducing the calculation of map point selection into any of the existing threads would introduce additional delays to those threads and impact performance. Therefore, the design choice was made to implement map point selection in a new, separate "map point selection" thread. This decision avoids delays and is suitable in scenarios where additional CPU cores are readily available.

Figure 7.1 shows how the map point selection thread is integrated with the original ORB-SLAM 2 software implementation. This figure also depicts the usual flow of data for ORB-SLAM 2 between the different existing submodules, each using different threads. The tracking thread first processes camera frames. The "local mapping" thread chooses a subset of the keyframes from the "tracking" thread to refine the local pose positions and local map point position estimates. The "loop closing" thread performs loop-closure detection. If a loop closure is found, it performs an initial update of the robot poses and maps point positions after loop closures using pose-graph estimation. Lastly, if the "loop closing" thread detects a loop closure, the "bundle adjustment" thread is started. This "bundle adjustment" thread performs bundle adjustment and refines estimates after the initial pose-graph estimation. Each thread reads the current shared map, performs operations on its copy of the map data, and updates the shared map after the thread has completed processing relevant frames passed to it by the other threads.

The creation and estimation of new map points occur within the "local mapping" thread, which carries out approximate SLAM estimation using only local observations. The time required for the "local mapping" thread is dependent on the environment but is typically performed multiple times a second. The typical time needed for the "local mapping" thread is less than the maximum potential time for map point selection. Consequently, it is reasonable to expect the insertion of multiple keyframes and their associated map points while map point selection is in progress. Since map point selection is implemented in parallel, it will be necessary to reduce the map to a budget less than the true hardware limitations. Choosing a lower value provides a buffer for these newly created map points, while map point selection is performed in parallel. The size of this buffer should be adjusted

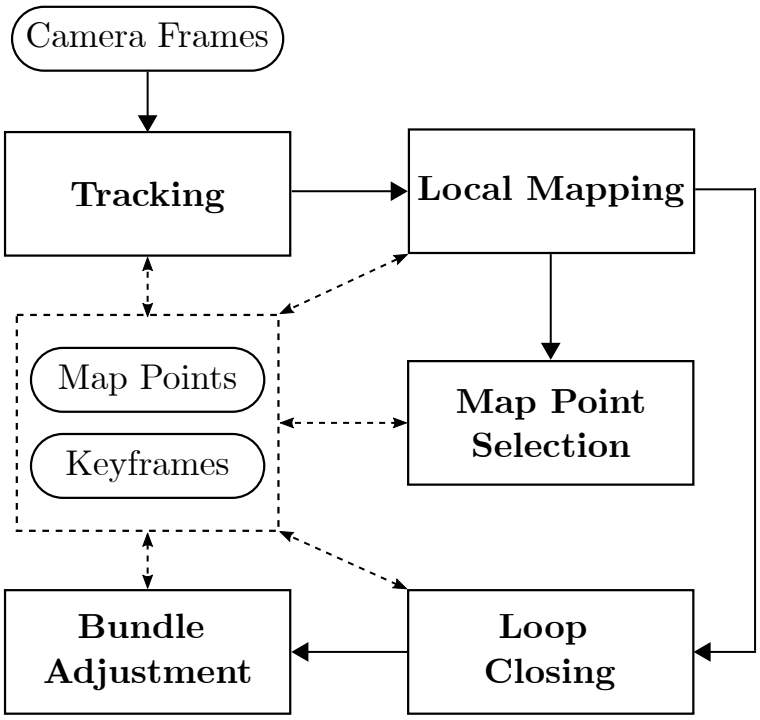

Figure 7.1: A diagram of the threads of the software implementation. This diagram depicts the separate threads of the software implementation in blocks, while shared data is depicted in ovals. The solid lines depict the sequential flow of the processing of a new camera frame between the threads, with only some camera frames being passed to the next thread. The "local mapping" thread starts the newly added "map point selection" thread and, similar to other threads, pushes updates to the shared map once the computation is completed. Different threads read from and update the shared map data asynchronously. Dashed-arrows represent this asynchronous flow of map data. See Mur-Artal and Tardós [8] for more details on the functioning of the original ORB-SLAM 2 algorithm.

based on the various factors. These factors include the rate at which new map points are inserted, the computation time required for map point selection, and the consequences of exceeding the budget. By employing this approach, we can effectively restrict the size of the map while accommodating the asynchronous nature of the algorithm implementation.

The map point selection thread is started by the "local mapping" thread whenever the map points exceed the budget and uses a copy of the shared data; that is, map points, keyframes and associated map point observations to formulate the map point selection problem. When map point selection finishes, the map points not selected are removed from the shared map used by the other threads. If the current map exceeds the budget, map point selection reduces that map's size back to the budget.

This section presented an overview of the implementation of map point selection alongside the existing ORB-SLAM 2 algorithm for online evaluation. Implementing map point selection in a separate thread mitigates some challenges with using map point selection alongside ORB-SLAM 2. However, it does not address all of the properties mentioned in Subsection 5.2.1. ORB-SLAM 2 relies on the pose-graph heuristic for accurate trajectory

estimates and a limited number of bundle-adjustment iterations. Consequently, we expect a less pronounced difference in the trajectory accuracy of different map point selection approaches during this online evaluation.

7.2. Online Results

This section evaluates different map point selection approaches using the online ORB-SLAM 2 implementation mentioned in Section 7.1. This section first presents the different map point selection approaches used for this evaluation, before presenting the experimental evaluation that results from these choices.

The different map point selection approaches from Section 6.3 are also included in the online evaluation in this section. These map point selection approaches include the existing coverage-based approach from Subsection 4.3.1 maximised with integer programming techniques with coverage parameter $b_{\text{cover}} = 100$ and trade-off parameter $\lambda = 25$, labelled *ip-wcover*. The localisation and odometry functions from Section 4.2 maximised with the lazy greedy algorithm are labelled *lg-local* and *lg-odom*, respectively. This section includes the combined odometry function with a coverage function from Section 4.4 using the lazy greedy algorithm labelled *lg-odom+cover*. The baseline of not applying map point selection labelled *full map* is also included in this evaluation. It is important to note that this chapter does not use the modified version of ORB-SLAM 2 from Chapter 5. Subsequently, the *full map* baseline obtains a worse trajectory accuracy than achieved by the modified ORB-SLAM 2 implementation for the offline evaluation.

For the purposes of demonstration, this section also includes a test where we halved the loop-closure threshold of ORB-SLAM 2 from 40 map points to 20 map points, and this approach is labelled *lg-odom(LT)*. This approach represents an alternative to the *lg-odom+cover* where rather than modifying the map point selection approach to improve map point selection for loop-closure detection, the SLAM implementation can be modified to improve recall performance.

This section evaluates each of the above-mentioned map point selection approaches online using the implementation from Section 7.1 on the KITTI trajectories with loop closures detectable by ORB-SLAM 2; that is, the trajectories used in the offline evaluation. In contrast to the offline evaluation, which only applied map point selection approaches on the input maps and evaluated the resulting map on the test trajectory, this online evaluation uses the full KITTI trajectory for the evaluation. Whenever the map point budget exceeds the specified threshold, map point selection is applied to select a subset of map points. For this evaluation, the map point budget is chosen as either 10,000, 20,000 or 30,000 map points. The online evaluation is repeated five times for each trajectory, and the median RPE and APE are shown in Tables 7.1 and 7.2.

Tables 7.1 and 7.2 show that the proposed map point selection approaches allow using

KITTI 00	RPE (%)			APE (m)		
map points	10k	20k	30k	10k	20k	30k
ip-wcover	0.82	0.77	0.73	1.8	1.6	1.5
lg-local	0.76	0.69	0.70	2.0	1.3	1.3
lg-odom	0.76	0.69	0.69	2.3	1.3	1.3
lg-odom+cover	0.71	0.71	0.69	1.3	1.3	1.2
lg-odom(LT)	0.72	0.70	0.68	1.3	1.3	1.2
full map		0.70			1.3	

KITTI 02	RPE (%)			APE (m)		
map points	10k	20k	30k	10k	20k	30k
ip-wcover	1.02	0.89	0.86	6.1	6.9	7.3
lg-local	0.86	0.80	0.83	11.0	7.0	10.1
lg-odom	0.83	0.82	0.79	8.5	7.9	6.9
lg-odom+cover	0.84	0.81	0.78	5.7	5.4	6.0
lg-odom(LT)	0.84	0.80	0.80	6.9	6.0	6.0
full map		0.76			5.7	

KITTI 05	RPE (%)			APE (m)		
map points	10k	20k	30k	10k	20k	30k
ip-wcover	0.46	0.44	0.42	1.0	1.0	0.9
lg-local	0.47	0.40	0.39	1.0	0.8	0.8
lg-odom	0.52	0.38	0.39	2.0	0.7	0.7
lg-odom+cover	0.38	0.40	0.39	0.7	0.8	0.7
lg-odom(LT)	0.37	0.39	0.38	0.7	0.7	0.7
full map		0.40			0.8	

Table 7.1: RPE and APE for KITTI 00, 02 and 05 trajectories using online selection.

KITTI 06	RPE (%)			APE (m)		
map points	10k	20k	30k	10k	20k	30k
ip-wcover	0.54	0.53	0.54	0.9	0.8	0.8
lg-local	0.51	0.54	0.51	0.7	0.8	0.9
lg-odom	0.49	0.50	0.48	0.9	0.7	0.7
lg-odom+cover	0.47	0.51	0.49	0.6	0.8	0.7
lg-odom(LT)	0.47	0.48	0.47	0.7	0.8	0.7
full map		0.51			0.8	

KITTI 07	RPE (%)			APE (m)		
map points	10k	20k	30k	10k	20k	30k
ip-wcover	0.50	0.50	0.49	0.6	0.5	0.5
lg-local	0.84	0.51	0.51	1.4	0.5	0.6
lg-odom	0.68	0.49	0.50	0.5	0.5	0.6
lg-odom+cover	0.50	0.49	0.51	0.6	0.5	0.4
lg-odom(LT)	0.50	0.49	0.51	0.5	0.5	0.5
full map		0.50			0.5	

Table 7.2: RPE and APE for KITTI 07 trajectory using online selection.

a subset of the available map points while obtaining trajectory accuracy close to that of the full map. The *lg-odom* and *lg-local* approaches match the APE and RPE of the full map using only a limited subset of map points (typically at map point budgets of 20,000 or 30,000) for all trajectories except the KITTI 02 trajectory. Similar to the offline results, more map points are needed for the KITTI 02 trajectory. For some map point budgets and trajectories, notably at 10,000 for the KITTI 05 and KITTI 07 trajectories, the ability to perform loop closures impacts trajectory accuracy. This results in a sharp decrease in trajectory accuracy for the *lg-local* and *lg-odom*. Similar to the offline results, this online comparison finds that the *lg-odom* approach generally offers improved trajectory accuracies compared to the *lg-local* approach.

For the KITTI 07 trajectory at the 10,000 map point budget, where *lg-local* and *lg-odom* approaches missed loop closures, *ip-wcover* offers better trajectory accuracy than the *lg-local* and *lg-odom* approaches. Except for these two cases, the *ip-wcover* approach is generally outperformed by the *lg-local* or *lg-odom* approaches. The relative performance of the *ip-wcover* is similar to the performance of this approach in offline results, except that the difference between the trajectory accuracy of the *ip-wcover* and *lg-odom* approaches is less pronounced than for the offline results.

The combined *lg-odom+cover* approach uses additional information about future loop closures. Similar to the offline results, this results in improved recall performance, which results in significantly improved trajectory accuracy for the KITTI 07 trajectory and 05 trajectory at a map point budget of 10,000 map points. This combined approach consistently outperforms or is comparable to the best alternative approaches across all trajectories and map point budgets. The trajectory accuracy of the *lg-odom+cover* approach is almost always comparable to the *full map* baseline, except for the KITTI 02 trajectory. However, *lg-odom+cover* still obtained the best trajectory accuracy at this map point budget compared to alternative map point selection approaches for the KITTI 02 trajectory.

The results from the *lg-odom(LT)* approach demonstrate a potential avenue where the *lg-odom* approach is used, but online ORB-SLAM 2 implementation is modified by lowering the required threshold of map point matches for accepting loop closures. This approach is in contrast to the *lg-odom+cover* approach, where map point selection is modified to better compensate for the front-end requirements of the SLAM implementation. Reducing the loop-closure detection threshold is sufficient to allow the detection of loop closures for the *lg-odom* approach at map point budgets of 10,000 for the KITTI 05 and 07 trajectories. This allows the *lg-odom(LT)* approach to obtain trajectory accuracies comparable to the *lg-odom+cover* approach and the *full map* baseline without requiring additional loop-closure information. Appendix C presents offline results using the same loop-closure threshold adjustment with the *lg-odom* approach as an alternative to the *lg-odom+cover* approach. The offline results from Appendix C shows that reducing the

loop-closure threshold boosts recall for the *lg-odom* approach, which allows for improved trajectory accuracy for this approach at low map point budgets, provided the loop-closure detection algorithm is sufficiently robust.

The results from this section indicate that using the *lg-odom+cover* approach is recommended, provided additional loop-closure information is available, which might only be available for some applications. If this information is unavailable, we recommend modifying the SLAM implementation to lower the loop-closure threshold and using the lazy greedy algorithm with the odometry utility or the *lg-odom(LT)* approach. Suppose neither loop-closure information nor the ability to modify the SLAM implementation is available. In that case, we cannot use the *lg-odom+cover* or *lg-odom(LT)* approaches, and the recommended map point selection algorithm depends on the application. The *lg-odom* generally outperforms or is competitive with the remaining alternatives over the range of map point budgets, except the KITTI 05 and KITTI 07 trajectories with 10,000 map points and is generally recommended over the *lg-odom* or *lg-local* approaches. However, for some applications, recall performance in these two specific cases might be more critical than trajectory accuracy in general. In that case, the coverage-based *ip-wcover* is now worth considering. Unlike in the offline results, the trajectory accuracy for this approach is closer to that of the *lg-odom* and *lg-local* approaches.

The discrepancy in trajectory accuracy between the techniques is less pronounced than for the offline results, as shown by comparing the difference between RPE error for the *ip-wcover* and the *lg-odom* approaches in the online results versus the offline results in Chapter 6. As described in Subsection 5.2.2, the offline results use a modified ORB-SLAM 2 implementation with an increased number of bundle adjustment iterations, and the modified implementation also pauses the processing of new frames until the bundle adjustment calculation is completed. These modifications allowed the full SLAM estimation to more consistently converge to the estimate given by the selected set of map points. These online results were generated without these modifications. Pose-graph estimates are used until the "bundle-adjustment" thread finishes, and only a limited number of bundle-adjustment iterations are used with the initial pose-graph estimates as the starting point. As a consequence, the accuracy of the pose-graph estimates for the online results has a more significant impact on the estimated trajectory than for the offline results. This greater emphasis on pose-graph estimation ultimately leads to map point selection approaches that allow for accurate trajectory estimates in the offline results, such as the *lg-odom* approach, to have more comparable trajectory accuracies to approaches which do not allow for accurate trajectory estimates in the offline results, such as the *ip-wcover* approach.

The results in this section demonstrate that online map point selection allows for reducing the number of map points in the map, often with a minimal impact on trajectory accuracy, provided an appropriate map point budget is chosen. Map point selection can

be implemented alongside an existing SLAM implementation online and already allows performing SLAM with reduced maps. However, changes to the SLAM implementation can result in significantly improved trajectory accuracy when map point selection is used as part of the SLAM implementation. These changes can include adjusting the bundle-adjustment iterations (as demonstrated by the offline results in Chapter 6) or the loop-closure detection (as demonstrated by the *lg-odom(LT)* approach) to compensate for the sparse maps produced by map point selection approaches. For an optimised implementation of ORB-SLAM 2 alongside map point selection, it could be beneficial to revisit all heuristic choices made by ORB-SLAM 2 that will be impacted by map point selection. These heuristics include not only loop-closure detection and the bundle-adjustment iterations, but also heuristics which govern choices based on the number of map points, for example, keyframe insertion or the co-visibility graph used by pose-graph estimation that uses the number of shared map points.

Chapter 8

Conclusion

This chapter summarises the findings of this dissertation in Section 8.1. This summary adds context to the discussion on the contributions of the research in Section 8.2. Lastly, Section 8.3 presents a discussion of potential future research avenues.

8.1. Research Summary

Existing sparse visual simultaneous localisation and mapping (SLAM) algorithms do not limit the storage space or memory required to store the map and associated visual feature information. This is undesirable for large-scale applications when dealing with resource-constrained platforms. While there are existing approaches for map point selection for localisation or visual odometry, this dissertation investigates the more challenging and novel problem of selecting map points for visual SLAM. Specifically, this dissertation proposes using map point selection algorithms to limit the map's total number of map points and associated feature information. The budgeted selection of map points is used as an abstraction of all the data associated with the map.

This dissertation proposes novel approaches to the map point selection problem for visual SLAM based on information-theoretic utility functions. Moreover, the dissertation investigates the novel application of coverage-based approaches for map point selection, previously used for localisation and visual odometry, to the SLAM problem. The proposed utility functions are maximised using greedy algorithms (typically the lazy greedy algorithm from Subsection 3.2.2). These greedy algorithms allow for the development of computationally efficient approaches that maximise the utility of the selected subset of map points. This dissertation evaluates map point selection approaches offline in both the KITTI and the EuRoC datasets, before demonstrating approaches online on the KITTI dataset.

The first utility proposed is an information-theoretic approach that maximises the information gain of the SLAM posterior distribution. Experimental results show that maximising this utility with the lazy greedy algorithm (the approach labelled *lg-SLAM*) allows for selecting less than 40% of the total map points, often with a minimal impact on the trajectory accuracy. However, using this utility for selection when using greedy

algorithms is computationally expensive, often requiring multiple hours to calculate a selection. At very small map point budgets, the trajectory accuracy of this approach is limited by the ability to detect loop closures.

This dissertation also proposes approximate information-theoretic utility functions based on approximating the SLAM posterior with simpler problems based on the localisation or odometry problems. Using these approximate utility functions with the lazy greedy algorithm led to computationally inexpensive approaches with trajectory accuracies comparable to those based on the SLAM utility during map point selection. These approximate approaches reduce execution times from hours to often less than a second. Results showed that the odometry utility allows for more accurate trajectory estimates than the localisation utility.

Additionally, this dissertation investigates coverage-based approaches for map point selection. An existing coverage-based approach using integer programming techniques previously used for the related problems of selecting map points for localisation and visual odometry is evaluated (labelled *ip-wcover*). This work adapts the coverage-based approach to be compatible with greedy algorithms to improve the asymptotic complexity (labelled *lg-wcover*). The greedy adaptation reduced execution times by orders of magnitude, with minimal effect on performance. Furthermore, this work proposes an alternative coverage-based approach to maximise the minimum coverage (labelled *max-mincover*). The existing *ip-cover* approach and its greedy adaptation *lg-cover* result in good loop-closure recall, but often with poor trajectory accuracy. Selecting map points for visual SLAM requires choosing a set of map points that allow for accurate loop-closure detection and bundle adjustment.

Lastly, a combined utility function of the odometry utility and a coverage-based function that assumes information about the locations of future loop closures is available. The combined utility function with the lazy greedy algorithm (labelled *lg-odom+cover*) either outperforms alternatives or is equivalent to the best alternative regarding estimated trajectory accuracy and loop-closure recall over the tested range of map point budgets.

8.2. Summary of Contributions

This section presents the contributions of this dissertation. The highlights of these contributions have been published in a peer-reviewed article in the Elsevier journal *Robotics and Autonomous Systems* [1].

The main contribution of this dissertation is the development of novel information-theoretic approaches for map point selection. The first of the information-theoretic approaches is the SLAM utility maximised with the lazy greedy algorithm, also known as the *lg-SLAM* approach. This novel map point selection approach allows robots to select a subset of map points over a wide range of map point budgets with minimal impact

on the estimated trajectory accuracy. No existing approach or random selection offers competitive trajectory accuracy to the *lg-SLAM* approach. Furthermore, the *lg-SLAM* approach serves as the basis for other approaches to approximate and as an offline point of comparison for estimated trajectory accuracy at reduced map point budgets. The second and third novel information-theoretic approaches developed by this dissertation are based on approximating the SLAM utility with either localisation or odometry problems. These localisation and odometry utilities maximised by the lazy greedy algorithm significantly improve the asymptotic computational complexity compared to using the SLAM utility, reducing execution times by multiple orders of magnitude on the tested datasets from hours to often less than a second. The experimental results show that the *lg-odom* approach outperforms the *lg-local* approach based on trajectory accuracy, and generally has comparable trajectory accuracy to the offline *lg-SLAM* approach. This map point selection approach allows for the best trajectory accuracy on average, while being as computationally inexpensive as the *ip-wcover* approach. The *lg-odom* approach now enables online map point selection with trajectory accuracy comparable to the offline *lg-SLAM* approach.

A second contribution is demonstrating that the front-end of a visual SLAM algorithm can significantly impact map point selection performance. As part of this contribution, this dissertation shows that if additional information is available regarding the location of future loop closures, a combined approach can be used. The combined approach uses the odometry utility and coverage utility for the loop-closure frames which leverage the lazy greedy algorithm (labelled *lg-odom+cover*). The combined approach improves loop-closure detection for the *lg-odom* approach, resulting in improved robustness at lower map point budgets. The combined approach surpasses alternatives in terms of trajectory accuracy or has trajectory accuracy similar to the best alternatives. The combined approach demonstrates the value of additional loop-closure information for map point selection, and also serves as a useful point of comparison for future map point selection approaches that do not use loop-closure information.

A third contribution of this dissertation is adapting an existing SLAM implementation to evaluate different map point selection approaches. From the techniques and results presented in this dissertation, it is clear that the design and performance of selection algorithms heavily depend on both the front- and back-end of the SLAM implementation. This work informs research regarding the desired behaviour of SLAM algorithms if it is to be integrated with map point selection. This dissertation also provides an example of modifying ORB-SLAM 2 to better compensate for the effects of map point selection. The modified version of ORB-SLAM 2 and all the map point selection approaches used in this dissertation are publicly available to facilitate the development of future map point selection algorithms.

A fourth contribution of this dissertation is adapting and evaluating the coverage-based

approaches for selecting map points for SLAM. These coverage-based approaches were previously used for localisation or visual odometry problems, but their application to the SLAM map point selection problem is novel. The experimental results for coverage-based approaches generally obtain excellent loop-closure detection performance at low computational cost. However, these approaches do not generally allow for accurate SLAM estimation. This contribution expands the knowledge of the performance of these coverage-based approaches in different applications.

Overall, the map point selection approaches developed in this work can significantly reduce sparse maps without impacting trajectory estimation accuracy. Moreover, this dissertation proves map point selection approaches benefit from changes in both the front-end and back-end components of the SLAM implementation. These results clearly demonstrate that it is essential to consider both the front- and back-end components when designing map point selection approaches for SLAM and vice versa.

8.3. Future Work

This section highlights potential future avenues for map point selection approaches based on the work presented in this dissertation. These directions include providing access to loop-closure information needed for the combined approach, evaluating approaches on a broader range of datasets and SLAM implementations, and developing an optimised ORB-SLAM 2 implementation for map point selection.

A limitation of this work is that the combined map point selection approach (labelled *lg-odom+cover*) assumes that information about the location of future loop closures is available, which is typically not the case for current SLAM implementations. Such information could be obtained by application-specific sources not typically available to the general-purpose visual SLAM algorithms. Future work to predict the location of future loop closures includes using the robot's planned trajectory or information about the environment topology. For example, identifying scenes as road intersections could be used to increase the likelihood of a loop closure at that location for vehicle applications. Another future option is to formulate the selection of loop-closure keyframes as a new optimisation problem by selecting a subset of spatially diverse keyframes where loop-closure performance is essential.

The experimental results in this dissertation are generated using modified versions of ORB-SLAM 2 on a range of datasets with which ORB-SLAM 2 is designed to function. Therefore, the experimental results in this dissertation are not necessarily representative of the performance of the proposed map point selection approaches on all datasets or SLAM implementations. Nevertheless, the research found that the novel information-theoretic approaches perform well in a wide range of scenarios. These approaches also do not require adjusting any parameters, with the exception of the combined approach that has a single

parameter. The map point selection approaches developed in this work are ready and simple to apply to a wider range of datasets or visual SLAM implementation in future research.

This dissertation demonstrates the online application of different map point selection approaches. This research focused on developing different innovative map point selection approaches rather than the optimised integration of map point selection with ORB-SLAM 2 specifically. This dissertation identified relevant properties of the ORB-SLAM 2 algorithm which would be beneficial to address when developing an optimised online implementation. Future work developing such an optimised implementation could yield improved trajectory accuracy and recall performance when using these map point selection approaches online. In developing an optimised ORB-SLAM 2 implementation for map point selection, researchers can revisit the loop-closure detection algorithm to improve recall for the sparse map produced by map point selection, ideally without compromising loop-closure precision.

The map point selection approaches developed by this dissertation are readily available and suitable for online applications, and reduce maps with minimal effect on trajectory accuracy. Additionally, this research provides a basis for future research to develop improved map point selection approaches and optimised SLAM implementations.

Bibliography

[1] Müller, C.J. and van Daalen, C.E.: Map point selection for visual slam. *Robotics and Autonomous Systems*, vol. 167, p. 104485, 2023. ISSN 0921-8890.

[2] Thrun, S.: Probabilistic robotics. *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[3] Zaffar, M., Ehsan, S., Stolkin, R. and Maier, K.M.: Sensors, SLAM and long-term autonomy: A review. In: *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pp. 285–290. 2018.

[4] Davison, A.J., Reid, I.D., Molton, N.D. and Stasse, O.: MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[5] Klein, G. and Murray, D.: Parallel tracking and mapping for small AR workspaces. In: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234. 2007.

[6] Mur-Artal, R., Montiel, J.M.M. and Tardós, J.D.: ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[7] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I. and Leonard, J.J.: Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[8] Mur-Artal, R. and Tardós, J.D.: ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[9] Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L.: Speeded-up robust features (surf). *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008. ISSN 1077-3142. Similarity Matching in Computer Vision and Multimedia.

[10] Rublee, E., Rabaud, V., Konolige, K. and Bradski, G.: Orb: An efficient alternative to sift or surf. In: *2011 International Conference on Computer Vision*, pp. 2564–2571. 2011.

[11] Strasdat, H., Montiel, J. and Davison, A.J.: Visual SLAM: Why filter? *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[12] Geiger, A., Lenz, P. and Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361. 2012.

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

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

[15] Van Opdenbosch, D., Aykut, T., Alt, N. and Steinbach, E.: Efficient map compression for collaborative visual SLAM. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 992–1000. 2018.

[16] Cheng, W., Lin, W., Zhang, X., Goesele, M. and Sun, M.-T.: A data-driven point cloud simplification framework for city-scale image-based localization. *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 262–275, 2017.

[17] Lynen, S., Sattler, T., Bosse, M., Hesch, J., Pollefeys, M. and Siegwart, R.: Get out of my lab: Large-scale, real-time visual-inertial localization. In: *Robotics: Science and Systems*. 07 2015.

[18] Bürki, M., Gilitschenski, I., Stumm, E., Siegwart, R. and Nieto, J.: Appearance-based landmark selection for efficient long-term visual localization. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4137–4143. 2016.

[19] Lynen, S., Zeisl, B., Aiger, D., Bosse, M., Hesch, J., Pollefeys, M., Siegwart, R. and Sattler, T.: Large-scale, real-time visual–inertial localization revisited. *The International Journal of Robotics Research*, vol. 39, no. 9, pp. 1061–1084, 2020.

[20] Kopitkov, D. and Indelman, V.: No belief propagation required: Belief space planning in high-dimensional state spaces via factor graphs, the matrix determinant lemma, and re-use of calculation. *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1088–1130, 2017.

[21] Zhao, Y. and Vela, P.A.: Good feature selection for least squares pose optimization in vo/vslam. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1183–1189. 2018.

[22] Carlone, L. and Karaman, S.: Attention and anticipation in fast visual-inertial navigation. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3886–3893. 2017.

[23] Grisetti, G., Kümmerle, R., Stachniss, C. and Burgard, W.: A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[24] Hartley, R.I. and Zisserman, A.: *Multiple View Geometry in Computer Vision*. 2nd edn. Cambridge University Press, ISBN: 0521540518, 2004.

[25] Triggs, B., McLauchlan, P.F., Hartley, R.I. and Fitzgibbon, A.W.: Bundle adjustment — a modern synthesis. In: Triggs, B., Zisserman, A. and Szeliski, R. (eds.), *Vision Algorithms: Theory and Practice*, pp. 298–372. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[26] Strasdat, H., Davison, A.J., Montiel, J. and Konolige, K.: Double window optimisation for constant time visual SLAM. In: *2011 International Conference on Computer Vision*, pp. 2352–2359. 2011.

[27] Galvez-López, D. and Tardos, J.D.: Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[28] Campos, C., Elvira, R., Rodríguez, J.J.G., M. Montiel, J.M. and D. Tardós, J.: ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM. *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[29] Engel, J., Schöps, T. and Cremers, D.: LSD-SLAM: Large-scale direct monocular SLAM. In: *Computer Vision – ECCV 2014*, pp. 834–849. Springer International Publishing, 2014. ISBN 978-3-319-10605-2.

[30] Li, Y., Snavely, N. and Huttenlocher, D.P.: Location recognition using prioritized feature matching. In: *Computer Vision – ECCV 2010*, pp. 791–804. Springer Berlin Heidelberg, 2010.

[31] Cao, S. and Snavely, N.: Minimal scene descriptions from structure from motion models. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 461–468. 2014.

[32] Camposeco, F., Cohen, A., Pollefeys, M. and Sattler, T.: Hybrid scene compression for visual localization. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7645–7654. 2019.

[33] Park, H.S., Wang, Y., Nurvitadhi, E., Hoe, J.C., Sheikh, Y. and Chen, M.: 3D point cloud reduction using mixed-integer quadratic programming. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 229–236. 2013.

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

[35] Schneider, T., Dymczyk, M., Fehr, M., Egger, K., Lynen, S., Gilitschenski, I. and Siegwart, R.: Maplab: An open framework for research in visual-inertial mapping and localization. *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1418–1425, 2018.

[36] Dissanayake, G., Durrant-Whyte, H. and Bailey, T.: A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In: *2000 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1009–1014. 2000.

[37] Zhang, S., Xie, L. and Adams, M.: Entropy based feature selection scheme for real time simultaneous localization and map building. In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1175–1180. 2005.

[38] Choudhary, S., Indelman, V., Christensen, H.I. and Dellaert, F.: Information-based reduced landmark SLAM. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4620–4627. 2015.

[39] Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J. and Krause, A.: Lazier than lazy greedy. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, p. 1812–1818. AAAI Press, 2015. ISBN 0262511290.

[40] Bentley, J.L.: Multidimensional binary search trees used for associative searching. *Commun. ACM*, vol. 18, no. 9, p. 509–517, sep 1975. ISSN 0001-0782.

[41] Gionis, A., Indyk, P., Motwani, R. *et al.*: Similarity search in high dimensions via hashing. In: *Vldb*, 6, pp. 518–529. 1999.

[42] Fischler, M.A. and Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, vol. 24, no. 6, p. 381–395, jun 1981. ISSN 0001-0782.

[43] Cummins, M. and Newman, P.: FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[44] Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K. and Burgard, W.: g^2o: A general framework for graph optimization. In: *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613. 2011.

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

[46] Ila, V., Polok, L., Solony, M. and Svoboda, P.: SLAM++ - a highly efficient and temporally scalable incremental SLAM framework. *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 210–230, 2017.

[47] *The OpenCV Reference Manual*. Itseez, 3rd edn, October 2023.

[48] Krause, A. and Golovin, D.: Submodular function maximization. *Tractability*, vol. 3, no. 71-104, p. 3, 2014.

[49] Nemhauser, G.L. and Wolsey, L.A.: Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research*, vol. 3, no. 3, pp. 177–188, 1978.

[50] Nemhauser, G.L., Wolsey, L.A. and Fisher, M.L.: An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978. ISSN 1436-4646.

[51] Minoux, M.: Accelerated greedy algorithms for maximizing submodular set functions. In: *Optimization techniques*, pp. 234–243. Springer, 1978.

[52] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J. and Glance, N.: Cost-effective outbreak detection in networks. In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 420–429. 2007.

[53] Edelkamp, S. and Schrödl, S.: *Heuristic search: theory and applications*. Elsevier, 2011.

[54] Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C.: *Introduction to algorithms*. MIT press, 2022.

[55] Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C. and Kleiner, A.: On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, nov 2009. ISSN 0929-5593.

[56] Golub, G.H. and Van Loan, C.F.: *Matrix computations*. 3rd edn. The Johns Hopkins University Press, 1996. ISBN 080185413.

[57] Brookes, M.: The matrix reference manual. 2020.
Available at: http://www.ee.imperial.ac.uk/hp/staff/dmb/matrix/intro.html

[58] Henderson, H.V. and Searle, S.R.: On deriving the inverse of a sum of matrices. *SIAM Review*, vol. 23, no. 1, pp. 53–60, 1981.

[59] Stewart, G.W.: *Matrix algorithms: volume 1: basic decompositions*. SIAM, 1998.

[60] Gill, P.E., Golub, G.H., Murray, W. and Saunders, M.A.: Methods for modifying matrix factorizations. *Mathematics of computation*, vol. 28, no. 126, pp. 505–535, 1974.

[61] Chen, Y., Davis, T.A., Hager, W.W. and Rajamanickam, S.: Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, vol. 35, no. 3, pp. 1–14, 2008.

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

[63] Vazirani, V.V.: *Approximation algorithms*, vol. 1. Springer, 2001.

[64] Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual. 2022. Available at: https://www.gurobi.com

[65] Krause, A., McMahan, H.B., Guestrin, C. and Gupta, A.: Robust submodular observation selection. *Journal of Machine Learning Research*, vol. 9, no. 93, pp. 2761–2801, 2008.

[66] Krause, A., McMahan, H.B., Guestrin, C. and Gupta, A.: Robust submodular observation selection. *Journal of Machine Learning Research*, vol. 9, no. 12, 2008.

[67] Wolsey, L.A.: An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, vol. 2, no. 4, pp. 385–393, Dec 1982. ISSN 1439-6912.

[68] Sturm, J., Engelhard, N., Endres, F., Burgard, W. and Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 573–580. 2012.

[69] Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C. and Kleiner, A.: On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, nov 2009. ISSN 0929-5593.

[70] Umeyama, S.: Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[71] Krause, A. and Golovin, D.: Submodular function maximization. In: Bordeaux, L., Hamadi, Y. and Kohli, P. (eds.), *Tractability: Practical Approaches to Hard Problems*, pp. 71–104. Cambridge University Press, 2014.

[72] Zhang, H. and Vorobeychik, Y.: Submodular optimization with routing constraints. *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, Feb 2016.

[73] Sviridenko, M.: A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, vol. 32, no. 1, pp. 41–43, 2004. ISSN 0167-6377.

Appendix A

Submodular maximisation subject to a knapsack constraint

A generalisation of the submodular maximisation problem for map point selection is the case where the cost of selecting elements in a set is unequal. The cost of each element with index i is defined as c_i, and the total cost of the selected elements $c(S)$ should not exceed the budget b, or

$$S^* = \underset{S \subseteq V, c(S) \leq b}{\arg\max}\ f(S) \tag{A.1}$$

This formulation is known as the submodular maximisation problem with knapsack constraints (or modular costs). The submodular knapsack problem is a generalisation of the classical 0-1 knapsack problem for which the objective $f(S)$ is linear instead of submodular (as noted by [71]). In Subsection 3.2.2, we assumed that relevant hardware constraints could be translated to a budget defined as a specified number of map points. Knapsack constraints allow for a more general scenario where the cost of map points is unequal.

These constraints are relevant to the map point selection problem for SLAM implementations, where the required memory or storage varies between different map points and cannot be translated to a cardinality constraint. This commonly occurs in scenarios where the cost of storing map point observations is high, for example, if map point features are stored for each past observation rather than only storing a singular feature per map point. In these implementations, selecting map points with more observations will be more expensive, as these map points require storing more features. However, the exact cost associated with map points is SLAM implementation dependent.

A common naive adaptation of the lazy greedy algorithm for knapsack constraints is to modify the lazy greedy algorithm to choose the element with the highest cost/benefit, or $\frac{f(S \cup \{x_i\})}{c_i}$. This naive approach has no worst-case performance guarantees and can, therefore, perform arbitrarily poorly in the worst case. Leskovec *et al.* [52] shows that this naive algorithm can be improved if we instead use the best solution between this naive cost-benefit heuristic and a greedy approach that chooses the elements with the highest marginal gain, until no more elements fit in the remaining budget. This variation returns a set with a utility that is at worst within $\frac{1}{2}(1 - e^{-1})$ from the optimal set. This

same algorithm has also been shown to generalise to cases where costs are submodular or approximately submodular by Zhang and Vorobeychik [72]. Algorithm 5 details this generalised cost-benefit algorithm for submodular maximisation problems subject to a knapsack constraint.

Algorithm 5: Generalised Cost-Benefit Greedy

input : $f : 2^V \to \mathbb{R}_+$ and k
output : S

$S_c \leftarrow \emptyset$
$V_c \leftarrow V$
while $V_c \neq \emptyset$ **do**
    $m_i \leftarrow \underset{m_i \in V_c}{\operatorname{argmax}} \frac{f(m_i|S_c)}{c(m_i|S_c)}$
    if $c(S_c \cup \{m_i\}) \leq b$ **then**
        $S_c \leftarrow S_c \cup \{m_i\}$
    $V_c \leftarrow V_c \setminus \{m_i\}$
$S_g \leftarrow \emptyset$
$V_g \leftarrow V$
while $V_g \neq \emptyset$ **do**
    $m_i \leftarrow \underset{m_i \in V_g}{\operatorname{argmax}} f(m_i|S_c)$
    if $c(S_c \cup \{m_i\}) \leq b$ **then**
        $S_g \leftarrow S_g \cup \{m_i\}$
    $V_g \leftarrow V_g \setminus \{m_i\}$
if $f(S_c) \geq f(S_g)$ **then**
    $S \leftarrow S_c$
else
    $S \leftarrow S_g$

The generalised cost-benefit greedy algorithm requires two executions of an algorithm similar to the lazy greedy algorithm, which requires $O(n^2)$ function calls. This results in a similar asymptotic complexity of $O(n^2 \times \mathrm{g}(n))$, where $\times \mathrm{g}(n)$ is the cost of evaluating the weighted marginal gains. It should be noted that there are alternative algorithms with stronger guarantees for the submodular maximisation problem with knapsack constraints, such as the algorithm proposed by Sviridenko [73]. The aforementioned approach guarantees a function value within $1 - e^{-1}$ of the optimal, but this algorithm requires $O(n^5)$ function evaluations instead of the $O(n^2)$ of generalised cost-benefit greedy. This increased asymptotic complexity renders the approach impractical for the scale of problems investigated in this dissertation. Due to the comparable scalability between the generalised the cost-benefit and lazy greedy algorithms, cost-benefit greedy is likely applicable to applications where the lazy greedy algorithm results in acceptable execution times.

Appendix B

Supporting Investigations for Approximate Information-theoretic Results

This appendix investigates the performance of the approximate information-theoretic approaches from Section 6.3 in greater detail. Section 6.3 showed that the *lg-odom* approach outperformed the *lg-SLAM* approach for the KITTI 05 dataset and improved trajectory accuracy beyond the full map for the KITTI 06 trajectory. Subsection B.1 investigates these results qualitatively. Additionally, Chapter 5 argued for using the frame before large loop closures to separate the input and test trajectories. Subsection B.2 investigates this choice in greater.

B.1. Visualisation of trajectory errors

This section presents a qualitative evaluation of the RPE to visualise the impact of map point selection on the estimated trajectory. For this qualitative assessment, we calculate the change in errors between using the full map versus using a map point selection approach. Note that the KITTI relative pose errors are calculated over distances from 100 to 800 meters. For this visualisation, we use only the errors over 100 meters. This visualisation is done in Figures B.1 and B.2 for the KITTI 05 and KITTI 06 trajectories, respectively. In both of these Figures, we compare the trajectory errors of the *lg-odom* approach versus the *lg-SLAM* approach at map point budgets where the odometry approach achieved trajectory errors lower than the full map.

As shown in Figures B.1 and B.2, the *lg-odom* approach managed lower RPE for large portions of the trajectory than the *full map* baseline. This reduction in errors is also not just restricted to the test portion of the trajectory, but includes portions of the input trajectory. This quantitatively shows how these map point selection techniques not only improved the local SLAM accuracy during the test portion of the trajectory, but allowed lower errors after loop closures occurred. While this demonstrates that map point selection can boost trajectory accuracy, the utility functions are monotone and do not capture this

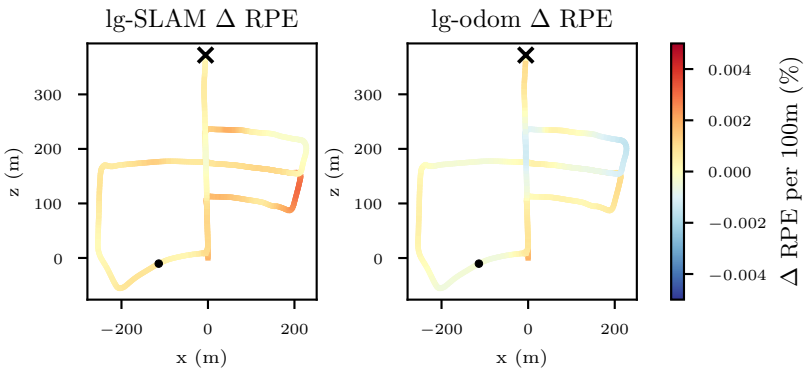

Figure B.1: Qualitative visualisation of the impact of map point selection on the relative pose error for the odometry and SLAM map point selection approaches for the KITTI 05 trajectory. This visualisation is for a input map budget of 20%.

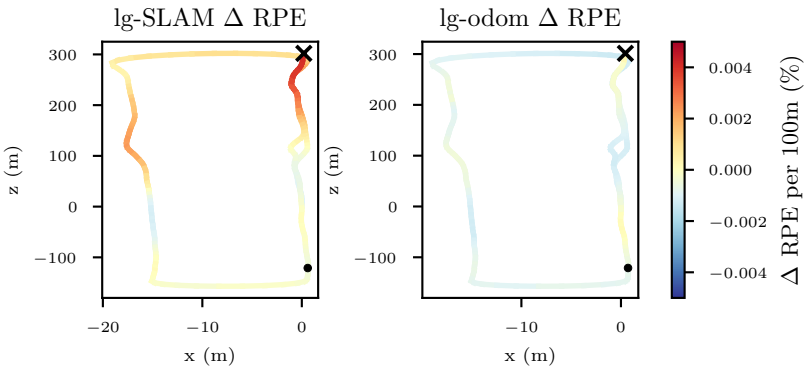

Figure B.2: Qualitative visualisation of the impact of map point selection on the relative pose error for the odometry and SLAM map point selection approaches for the KITTI 06 trajectory. This visualisation is for a input map point of 12%.

behaviour. Therefore, if given a larger map point budget, greedy algorithms will continue to choose more map points even if the best trajectory accuracy is obtained for a smaller map.

B.2. Impact of the evaluation frame for the KITTI trajectory

This section investigates the impact of the evaluation frame; that is, the frame at which to separate a trajectory into an input and test trajectory. Section 5.5 argued that splitting an input and test trajectory before a large loop closure is a good choice as the estimated trajectory likely includes significant drift, and accurately updating the estimation involves large portions of the map. The validity of this assumption is tested on the KITTI 05 dataset. A subset of candidate frames along the trajectory are chosen at locations that are before and after loop closures. The chosen candidate frames: 1250, 1350, 1600, 1900,

2200 and 2550, are shown in Figure B.3.

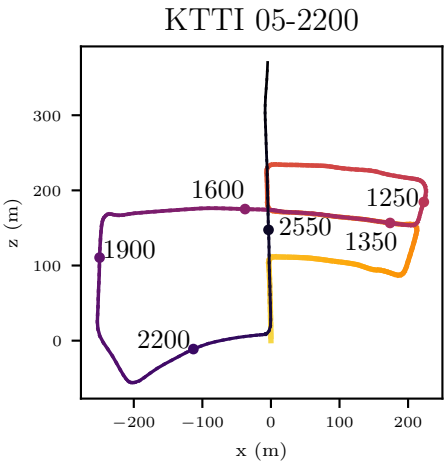

Figure B.3: Different potential frames at which to evaluate map point selection approaches.

For each of these evaluation frames, the relative pose error of different map point selection approaches are evaluated. This shows how the evaluation frame impacts the relative performance and variation between different approaches. The *lg-local*, *lg-odom* and *ip-wcover* approaches are evaluated using a map point budget of 15%. Additionally, the empty map and full map approaches are included for comparison. We do not include random selection as this approach failed to close loops and therefore performed equivalent to the empty map approach. The median of the relative pose errors are shown in Figure B.4.

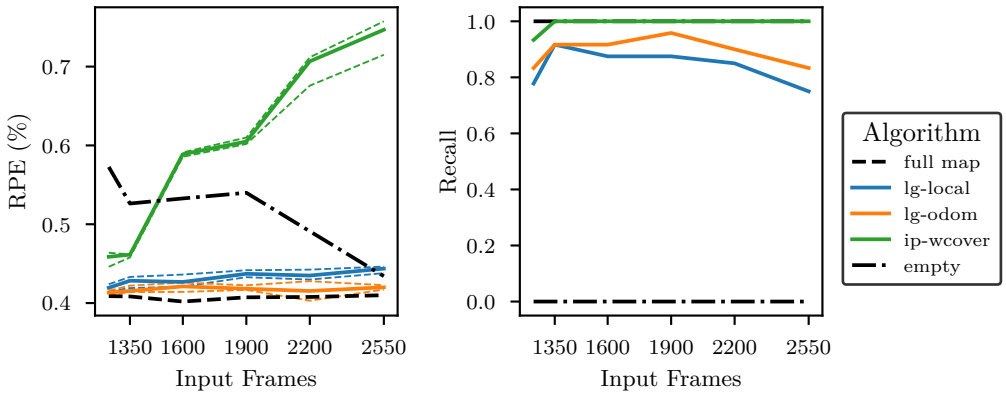

Figure B.4: KITTI 05 Evaluation Frames. Comparison of different frames at which to evaluate selection algorithms

As noted in Section 6.3, if loop-closures are not detected, the trajectory estimates of the ORB-SLAM 2 algorithm (and the modified ORB-SLAM 2 by extension) are not updated. Suppose the initial trajectory estimate is very accurate. In that case, the empty map approach can outperform approaches that detect loop closures but do not allow for accurate trajectory estimation, as shown in Figure B.4.

The chosen evaluation point of 2200 frames for the KITTI 05 is not the most challenging evaluation frame that results in the largest separation between the techniques if the map point budget scales with the size of the input map. However, the choice of using the frame at 2200 results in a reasonable separation between the respective approaches, and the information-theoretic approaches are insensitive to the evaluation point.

Appendix C

Lower Threshold Experimental Results

In the experimental results from Section 6.3, the *lg-odom* approach will sometimes perform poorly in terms of loop-closure recall. This can be improved in one of two ways: by improving the map point utility to capture the utility of map points for loop-closures, as shown by the combined approach from Section 4.4, or by adjusting the loop-closure front-end, as shown in this Appendix.

As shown in Figure C.1 and C.2, reducing the loop-closure threshold improves recall. This change leads to improved trajectory accuracy for the tested trajectories. For sufficiently robust front-ends, this offers a compelling alternative and does not require additional information like the techniques presented in Section 4.4. However, reducing the threshold for loop-closures is not appropriate in all settings, as this can result in incorrect loop-closures, which results in irrecoverable failure for SLAM implementations not robust to such errors (such as ORB-SLAM 2). This appendix shows that a mild reduction in this threshold is a simple and practical approach to gain some additional performance out of the map point selection approaches in this work, if loop-closure information is not available. The disadvantage of this approach is that loop closures still often limit the ability to reduce the maps significantly even with these reductions.

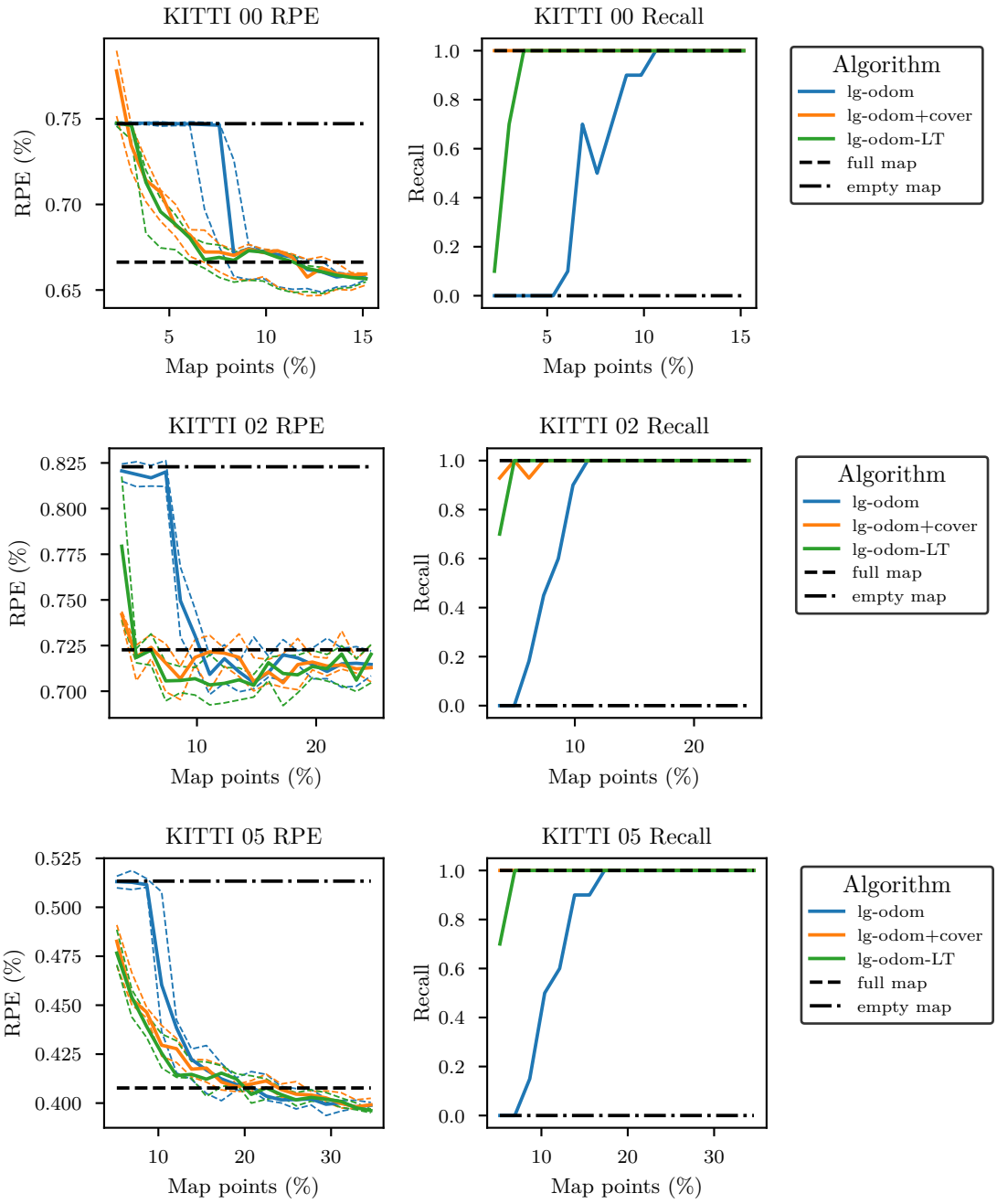

Figure C.1: RPE and recall for the KITTI 00, 02 and 05 trajectories as a function the map point budget.

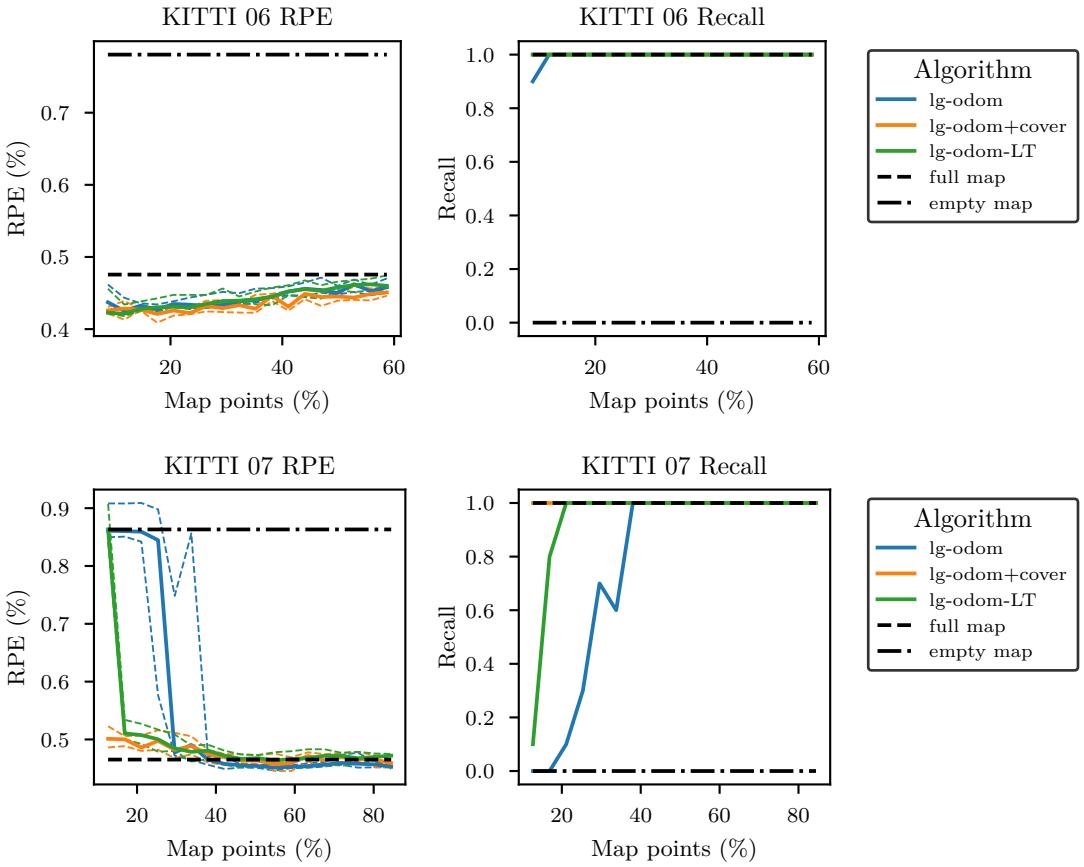

Figure C.2: RPE and recall for the KITTI 06 and 07 trajectories as a function of the map point budget.