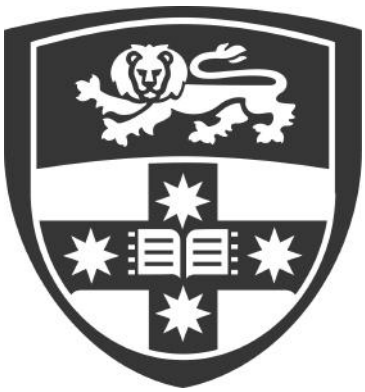
Towards Robust Multi-Robot SLAM in Perceptually Challenging Subterranean Environments

By

ALEX STEPHENS



Faculty of Engineering & IT
School of Aerospace, Mechanical & Mechatronic Engineering
UNIVERSITY OF SYDNEY

A thesis submitted in partial fulfilment of the requirements of the degree of BACHELOR OF ENGINEERING (HONOURS).

JANUARY 2020

Executive Summary

Subterranean exploration is one of the most challenging frontiers for autonomous robotics, and has seen increasing interest in recent years due to potential high-value applications in areas such as disaster response, search and rescue, and planetary exploration. Achieving robust and accurate simultaneous localisation and mapping (SLAM) in subterranean environments is extremely difficult, involving overcoming challenges of uneven illumination, visual obscurants, feature-poor regions and complex topologies.

This thesis presents a series of contributions to Large-Scale Autonomous Mapping and Positioning (LAMP), a SLAM system developed for the DARPA Subterranean Challenge by the CoSTAR team at the NASA Jet Propulsion Laboratory. LAMP is characterised by its incorporation of multiple sensor modalities for odometry and landmark detection, which are fused to provide improved redundancy for localisation and mapping in challenging environments.

As a system that regularly has components and requirements added, the architecture of LAMP must accomodate not only for the current system requirements, but for foreseeable future additions and modifications. We present a new design for the full system architecture of LAMP, in which the software is refactored from the ground up in order to improve its utility for future development. The system is reorganised into logically separated modules which derive from a set of common frameworks and interfaces, producing a system that is minimal, extensible and intuitive to work with.

Meaningful evaluation of the accuracy of a SLAM system is also a challenge in subterranean operations. Typical methods for obtaining ground truth trajectories, such as

motion capture or GPS, are either impossible, impractical or overly expensive in these environments, yet this data is crucial to ongoing evaluation of the system performance during development. This thesis demonstrates a novel technique for performance evaluation of a SLAM system using sparse ground truth data, which can be easily obtained within the practical constraints of subterranean environments. This method is shown to allow for richer analysis of the performance of the system, enabling more effective diagnosis of issues for future development.

A further challenge addressed in this work is that of multi-robot mapping – large localisation drift is often incurred in perceptually challenging environments, which presents difficulties for fusing the data into a consistent map. The pre-existing LAMP system makes use of loop closures to correct for localisation drift within the trajectory of a single robot. This work extends the single-robot capabilities into a broader framework for multi-robot systems, adding a centralised loop closure search algorithm which runs on a base station as data is received from each robot. We demonstrate that this new algorithm is able to produce significant improvements to the quality of multi-robot maps, ensuring that the data from each robot is fused effectively into a consistent master map without excessive additional computational requirements.

The work presented in this thesis remodels the LAMP system into a modular and intuitive platform for future development, with improved pipelines for quantitative performance evaluation and multi-robot data fusion. With each of these contributions, we advance ever closer to achieving the capabilities required for robust multi-robot exploration and mapping of subterranean environments.

Author's Declaration

This is to certify that to the best of my knowledge, the content of this thesis is my own work and has not been submitted for any other degree or purpose. I certify that the intellectual content of this thesis is my own work, and that any contributions from others have been appropriately acknowledged and referenced.

Specifically, the work I contributed includes:

- Researching and writing the literature review.
- Significant input into the architectural and software design for the LAMP system.
- Implementation of the base station software for LAMP.
- Proposal and software design of the process for approximating ground truth trajectories based on prior information, and implementation of the performance analysis pipeline in C++ and Python.
- Design and implementation of the algorithm for centralised inter-robot loop closure search, including refactoring of existing software for single-robot loop closure search in order to allow inputs from multiple robots at the base station.

The work I did not conduct:

- Design and implementation of the full LAMP system.
- Design of the process for acquiring ground truth locations using a total station.
- Design and implementation of the software for single-robot loop closures in LAMP.

.....	.....
Alex Stephens (student)	Dr. Mitch Bryson (supervisor)
School of AMME	Australian Centre for Field Robotics
The University of Sydney	The University of Sydney

ACKNOWLEDGEMENTS

This thesis was completed at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the University of Sydney through the Engineering Sydney Industry Placement Scholarship (ESIPS) program and the Vice Chancellor's Global Mobility Scholarship, and by the National Aeronautics and Space Administration.

First and foremost, I would like to thank my supervisor and team lead at JPL, Dr. Benjamin Morrell, for the incredible opportunities afforded to me under his leadership. His mentorship, support, and willingness to allow me to explore my chosen directions in research were invaluable foundations to my experience at JPL. I thank also Prof. Stefan Williams for opening up the remarkable opportunity to do research at JPL as an undergraduate student.

I am grateful to have been a part of the fantastic CoSTAR team for the DARPA Subterranean Challenge under the direction of our Principal Investigator, Dr. Ali-akbar Agha-mohammadi. Being surrounded by such a brilliant, passionate group of engineers allowed me to develop further as a roboticist than I would have thought possible in six short months. The Perception group – Ben, Kamak, Ed, Nobuhiro, Matteo, Sammi, and Jeremy – were all consistently a pleasure to work with, be it through discussions in group meetings, collaborative coding sessions, or the many late nights of field testing. I thank also our external collaborators, in particular Prof. Luca Carlone and Yun at the Massachusetts Institute of Technology, for our many enlightening discussions.

To my supervisor at the University of Sydney, Dr. Mitch Bryson, I am grateful for the prompt responses to my many emails, and for his guidance and feedback over the course of my thesis.

Finally, I thank my partner, Alice, and my parents, Andreas and Sadhana, for their support during my time at JPL and throughout my undergraduate degree. I have had many incredible opportunities over the years, owing significantly to the constant support and encouragement from the people closest to me, for which I am forever grateful.

Abstract

Recent developments in the capabilities of mobile robots have led to increasingly diverse applications in areas that are too dull, dirty or dangerous for humans. Subterranean environments such as mining facilities, cave networks and the urban underground are some of the most challenging for mobile robots, often involving complex unknown topologies, rough or unstable terrain, and degraded sensing and communication capabilities. The development of collaborative autonomous robots to effectively explore and map subterranean environments would assist in terrestrial applications such as disaster response or search and rescue, as well as provide unprecedented opportunities for exploration of other worlds.

Operations in subterranean environments bring to light many domain-specific challenges for multi-robot systems. Current frameworks for multi-robot simultaneous localisation and mapping (SLAM) are largely untested in environments with limited communications, repetitive structures and few distinct visual features. This thesis contributes to the development of a robust multi-robot SLAM system for subterranean exploration and mapping. Contributions include design and implementation of a modular and extensible system architecture for multi-robot SLAM in subterranean environments; development of a novel methodology for performance evaluation with sparse ground truth data; and implementation and testing of an algorithm for inter-robot data association under computational constraints.

Keywords: multi-robot SLAM, subterranean robotics, ground truth, loop closure.

Table of Contents

List of Figures

Figure **Page**

List of Acronyms and Abbreviations

BLAM	Berkeley Localisation and Mapping
CoSTAR	Collaborative SubTerranean Autonomous Resilient Robots
DARPA	Defense Advanced Research Projects Agency
EKF	extended Kalman filter
GICP	Generalised-ICP
ICP	iterative closest point
IMU	inertial measurement unit
IRM	information roadmap
iSAM	incremental smoothing and mapping
JPL	Jet Propulsion Laboratory
LAMP	Large-Scale Autonomous Mapping and Positioning
LIO	lidar inertial odometry
LOAM	Lidar Odometry and Mapping
MAP	maximum a posteriori
NASA	National Aeronautics and Space Administration
NLLS	nonlinear least squares
PCM	Pairwise Consistency Maximisation
RANSAC	random sample consensus
ROS	Robot Operating System
SfM	structure from motion
SLAM	simultaneous localisation and mapping

TIO	thermal inertial odometry
UML	Unified Modelling Language
UWB	ultra-wideband
VIO	visual inertial odometry
VO	visual odometry
VSLAM	visual SLAM

CHAPTER 1

INTRODUCTION

Recent advances in computer engineering have enabled the development of increasingly sophisticated and capable mobile robots. Historically, robotics was developed for industrial applications in which the environment was organised around the machines. Modern robots are able to explore complex unknown environments and perform tasks that are too dull, dirty or dangerous for humans. This presents opportunities for a diverse array of new applications, ranging from subterranean search and rescue to exploration of other planets in our solar system. These applications come with a host of domain-specific challenges, many of which have only just begun to be properly addressed in current robotics research.

1.1 Background and Motivation

1.1.1 Subterranean Robotics

Underground environments often present significant challenges for exploration, mapping, situational awareness and communication. Tunnels or mines can span many kilometres deep underground, presenting constrained passageways and vertical shafts which may

be prone to hazardous collapses or rock bursts. Urban underground areas can present complex multi-storey layouts which become extremely difficult to traverse and navigate in the event of natural or man-made disasters. Natural cave networks often have irregular terrain and unknown topologies, extending deep underground or even underwater.

The nature of such environments makes them difficult for humans to traverse efficiently, effectively and safely. Search and rescue operations in subterranean environments typically present significant dangers from unstable and degraded structures. In the absence of existing infrastructure, communication deep underground is restricted to short range or line-of-sight, increasing the difficulty of rapid emergency response. Exploration and mapping requires venturing into unknown and potentially hazardous environments, presenting risks to the safety of even the most prepared and skilful explorers.

Figure 1.1. Robotic exploration of challenging subterranean environments. Photos courtesy of Kamak Ebadi, NASA JPL.

The dangers of underground exploration have led to increasing demand for the development of subterranean robotics (Figure 1.1). Today, many types of highly capable robots are available at low cost, allowing them to be readily deployed in scenarios which might otherwise have put humans lives at risk. Robots are well-suited to surveying and mapping tasks that are menial work for humans, and can be purpose-built for effectiveness in any given environment. Robots can also act as vanguards to human exploration of unfamiliar environments, such as in the case of planetary exploration

where robots are able to provide detailed information about places that are as yet inaccessible to humans.

1.1.2 Autonomous Robotics

The availability of powerful and compact computers has opened up new frontiers in the development of intelligent systems. Modern computers can process large data streams in real-time, making complex decisions which allow them to respond effectively to a broad range of scenarios. This has enabled the development of autonomous robotics – robotic systems that are able to perceive and respond to their environments without human input. Autonomous robots are used in industry to transport materials around mines, factories and warehouses, reducing the size of the human workforce required. Autonomous cars can navigate increasingly complex and dynamic environments, offering the potential to reshape urban transportation infrastructure in the coming years.

Autonomous robots also offer significant potential in domains with restricted communications. Subterranean environments typically preclude the use of GPS and long-range communication technologies, as the signals are occluded by the environment. The result is that an autonomous robot can explore much farther than a human-operated one. Robots exploring other planets face large communication latencies due to the light-speed travel time to and from Earth – for the Mars rovers, the round-trip communication time can be as large as 40 minutes. These latencies make real-time remote operation impossible, meaning that robots exploring other bodies in our solar system must be able to navigate and perform tasks without human input.

1.1.3 Multi-Robot Subterranean Exploration

In most robotics applications, two robots working collaboratively can achieve more than one. However, multi-robot systems also have a variety of additional domain-specific advantages in subterranean environments.

Robots in underground environments typically rely on some form of communication relaying in order to talk to the outside world. In the case of a single robot, this would be

in the form of communication nodes periodically dropped by the robot, allowing it to communicate back to a base station at its starting location. A multi-robot system exploring the same area has a twofold advantage over the single robot. Firstly, the robots will be able to carry and drop more communication nodes, producing a communication network with more coverage, greater redundancy and therefore higher robustness. Secondly, the robots themselves can act as communication nodes, increasing the area over which the robots can explore while remaining in communication with the base station.

In time-critical scenarios, multiple robots can explore and map subterranean environments much faster than a single robot. Combining and sharing information from all robots also allows more accurate maps to be produced. Development of multi-robot systems that can coordinate exploration and effectively share information is therefore a significant area of interest in robotics today.

1.1.4 DARPA Subterranean Challenge

The DARPA Subterranean Challenge is a three-year competition that aims to foster development of technologies to rapidly explore, map, navigate and search complex underground environments [1]. The competition consists of three "circuits" as shown in Figure 1.2, comprising mining tunnels, urban underground and natural cave environments respectively, followed by a final circuit which incorporates elements of all three.

Figure 1.2. DARPA Subterranean Challenge domains. Image from [2].

All work in this thesis has been completed as part of the CoSTAR team in the DARPA Subterranean Challenge, a collaboration between NASA Jet Propulsion Laboratory (JPL), Massachusetts Institute of Technology, and the California Institute of Technology.

Competition Details

The primary aim of the competition is to accurately report the locations of artifacts placed throughout the environment. Artifacts must be reported to within 5 metres of their true location within a DARPA-defined global reference frame in order to count towards a team's score. Teams are allowed a single human operator who may interact with the robots through a base station located at the circuit entrance. Teams have one hour to report back as many artifacts as possible.

Technical Challenges

The competition environments are quite large, necessitating the use of multiple robots to effectively explore the full area within the time limit. Since there is only a single operator to issue commands, it is beneficial for the robots to be able to explore the environment autonomously, coordinate, and share information as required.

In order to report artifacts back with the required accuracy, the robots must be able to map the environment with high accuracy, restricting positional drift to a few metres even while exploring many kilometres of the environment. The environments may also be significantly degraded, presenting additional challenges for both terrain traversal and perception of the environment using on-board sensors.

1.2 Aims and Objectives

This thesis contributes to the development of a multi-robot collaborative SLAM system designed for operation in challenging subterranean environments as part of the DARPA Subterranean Challenge. In particular, it aims to address the following key objectives:

1. **Design of a modular and extensible system architecture for multi-robot SLAM in challenging subterranean environments**.

   Current SLAM systems typically make use of limited sensing modalities which are selected as the most suitable for particular purposes or environments. This thesis develops an architecture for a SLAM system that can easily incorporate new components and sensing modalities as needed to address the requirements of diverse and challenging subterranean environments.

2. **Development of a practical and intuitive technique for performance evaluation of SLAM systems within the practical constraints of subterranean operations**.

   In subterranean environments, it is typically impractical and prohibitively expensive to obtain the full ground truth data required for comprehensive performance evaluation. This work develops a technique that works within the practical constraints of such environments to provide richer performance analysis than what is available through standard methods.

3. **Design and implementation of centralised fusion of mapping data from multiple robots**.

   The accuracy of mapping in multi-robot SLAM systems is heavily dependent upon effective fusion of data from multiple robots. This thesis addresses development and implementation of a system for lidar-based inter-robot loop closures, and demonstrates the resultant improved mapping capabilities.

1.3 Outline of Thesis

Each of the chapters in the body of this thesis presents work done on a particular aspect of a larger, more complex system. In order to make the thesis as intuitive and readable as possible, results are presented where appropriate within each of the main chapters, rather than in a single results chapter.

The core chapters of this thesis are as follows:

Chapter	Contents
2	A thorough review of the background literature and current state-of-the-art for multi-robot SLAM, with a focus on technologies suited to subterranean environments.
3	Design and development of an updated framework for LAMP, the multi-robot SLAM system upon which all work in this thesis is built.
4	Development of a new technique for trajectory analysis in subterranean environments using sparse ground truth data. Examples are presented of the types of analysis that becomes available through the technique.
5	Development and implementation of inter-robot lidar loop closure search within LAMP. This chapter demonstrates the enhanced multi-robot mapping capability that becomes available through the addition of inter-robot loop closures.

The thesis ends with a conclusions chapter, which includes discussion of potential avenues for future work in each of the primary research areas.

CHAPTER 2

LITERATURE REVIEW

This thesis contributes to capabilities for multi-robot systems to explore unknown subterranean environments, in particular their ability to share information and achieve accurate collaborative mapping in large, feature-poor environments.

This chapter provides an overview of several key areas that form the foundation of this work, as well as a review of the current state-of-the-art in these areas. Firstly, a general introduction to simultaneous localisation and mapping (SLAM) is provided, followed by an overview of the graph SLAM formulation that forms the basis of the work in this thesis. The subsequent sections review the relevant background and current state-of-the-art in lidar-based SLAM, loop closure methods, and multi-robot systems. The chapter ends with a description of the gaps in the literature that form the basis of the work in this thesis.

2.1 Simultaneous Localisation and Mapping (SLAM)

Autonomous navigation requires a robot to be able to know its position and orientation (its *pose*) within the environment. The process of determining a robot's pose with respect to some predefined frame of reference is known as localisation.

In some scenarios, GPS can be used to provide localisation information. However, in applications indoors, near buildings, or underground, GPS is typically not accurate or reliable enough to be used for navigating obstacles. In such cases, instead of making use of a priori knowledge of the environment, the robot must construct a map of its environment from observations made as it explores. This map can take a variety of forms, from simple features or key points identified within the environment, to occupancy grids or point cloud representations of the full 3D environment.

The problem of a robot acquiring an environment map while simultaneously localising itself with respect to that map is known as the simultaneous localisation and mapping (SLAM) problem. Solutions to the SLAM problem form the foundation of many applications for autonomous mobile robots.

2.1.1 Probabilistic Formulation of SLAM

The *full* SLAM problem can be described concisely as follows: given a series of observations $z_{1:t}$ and control inputs $u_{1:t}$, the agent (usually a robot) aims to estimate the map m as well as its trajectory $x_{1:t}$ up to the time t.[1] This is commonly formulated as a posterior probability distribution:

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \tag{2.1}$$

Rather than estimating the full probability distribution, most SLAM algorithms aim to estimate the most likely trajectory and map (the mode of the posterior) in a process known as maximum a posteriori (MAP) estimation [3]. This computation requires an observation model, which describes the probability of making a particular observation given the current pose and map

$$p(z_t \mid x_t, m) \tag{2.2}$$

[1] An alternate form is the *online* SLAM problem, in which only the map and most recent pose are estimated. However, the focus of this thesis will be on solutions to the full SLAM problem.

as well as a motion model, which describes the probability of a transition to a particular state from the previous state under the latest control input

$$p(x_t \mid x_{t-1}, u_t) \tag{2.3}$$

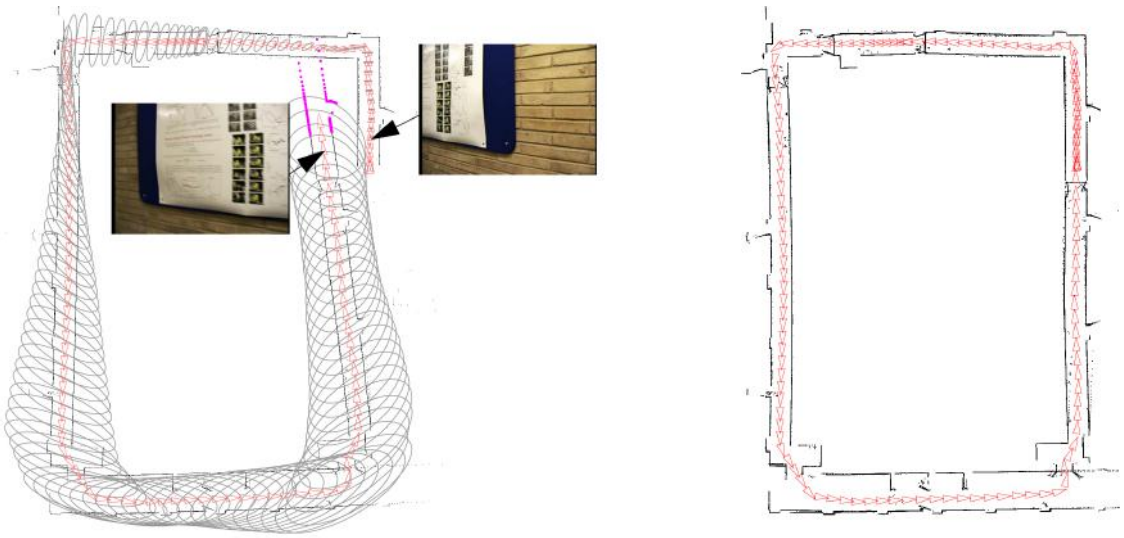
SLAM implementations will often use odometry in order to measure the state transitions, rather than directly making use of the control inputs, thereby completely decoupling the SLAM system from the control system.

With these models known, the posterior in Equation 2.1 at time t can be computed recursively in terms of the posterior at time $t-1$ using Bayes' theorem, given a new observation z_t and control input u_t [4]. Computation of the full posterior is typically infeasible due to high dimensionality, so most practical systems use approximations to make the problem tractable [3]; in typical applications, robots run a SLAM algorithm in real-time, processing incoming observations and control inputs at discrete time-steps and incrementally updating the map and trajectory estimates.

2.1.2 Loop Closure

As a robot travels from its starting location, it accumulates uncertainty in its map and pose estimates, due to the covariances associated with the probabilistic models and sensor measurements. These uncertainties grow without bound as the length of the trajectory increases. If the robot recognises a location that it has previously visited, it can use this information to adjust the map estimate, counteracting the drift that has occurred since the location was first visited. This is known as a loop closure. Detection of loop closures is extremely valuable, since a correct detection will cause all affected covariances to decrease monotonically [5].

Recognition of a previously visited location can be achieved in a variety of ways, based on stored information from any type of exteroceptive sensor. Vision-based methods are popular, since they can be easily tailored to recognise locations in the particular environment in which the robot is deployed – Figure 2.1 shows an example of a visual loop



(a) (b)

Figure 2.1. An example of a loop closure removing accumulated errors. (a) The robot has travelled around the loop, accumulating significant errors which have caused its position estimate to drift. It recognises its starting location from some previously seen features on the wall. (b) A correction is applied to the map based on the detected loop closure, resulting in a much more accurate map. Images from [6].

closure. Loop closures can also be found by matching lidar point clouds [7], recognising map misalignment [8], or in general by storing and indexing any feature of interest in the environment.

Section 2.4 gives a detailed review of loop closure methods in the context of lidar-based SLAM, which are of most relevance to this thesis.

2.1.3 Visual SLAM

SLAM implementations have been successfully demonstrated with a variety of sensors. However, visual SLAM (VSLAM), in which a camera is used as the primary sensor, has seen significant research interest due to the low cost, low energy requirements, and lightweight nature of these sensors.

Visual Sensing

While VSLAM systems may rely purely on a standard monocular RGB camera, they typically incorporate stereo cameras or depth (RGB-D) cameras to remove scale ambiguity and provide direct access to 3D information [9]. The 3D structure of an environment can also be reconstructed from successive 2D images in a process known as structure from motion (SfM), in which correlated features in successive images are used to reconstruct the intervening motion in 3D space [10]. Many VSLAM systems will also make use of an inertial measurement unit (IMU) to provide an additional baseline for the motion of the system (discussed further in Section 2.3.2).

Feature Selection

Traditional visual SLAM algorithms rely on distinctive features – also known as landmarks – which can be identified in successive frames and used to construct a map of the environment. This approach is known as indirect, or feature-based, visual SLAM.

The choice of features used in a VSLAM is based on a range of factors, and can have a large effect on the performance of the algorithm in different environments. In man-made environments, which typically have very structured geometries, corners are a popular choice as features due to their invariance (similar appearance from different directions, and under different lighting conditions) and ease of extraction [9]. However, in natural environments, well-defined corners may be less common, and more complex features such as edges, textures or other domain-specific interest points may be more appropriate.

An alternative to feature-based visual SLAM that has grown in popularity in recent years is direct visual SLAM, which infers motion directly from the geometry of the pixel intensities between frames. State-of-the-art algorithms such as LSD-SLAM [11] demonstrate the improved robustness achieved by direct methods, which make use of *all* information in the image rather than only extracted features. These methods come with increased computational cost, but can provide significant performance improvements in environments with few distinct features, or features that are not known a priori.

Strengths and Weaknesses

Visual SLAM has seen great interest due to the suitability of the camera as a sensor on mobile robots. Cameras are lightweight, cheap, low-powered, and capture rich information about the scenes they observe. Many examples of highly successful VSLAM implementations exist in the literature.

However, performance of VSLAM algorithms can suffer significantly in the presence of variable lighting conditions, occlusions, or a lack of suitable features in the environment [9]. For subterranean exploration, while visual odometry may provide a helpful contribution to the odometry backbone of a SLAM system, the usefulness of a pure VSLAM solution is limited by the characteristics of the environment.

2.1.4 Point Cloud SLAM

SLAM solutions based on 3D point clouds have also been implemented with great success, often able to produce extremely accurate and detailed maps of the full environment. An in-depth overview and exploration of the current state-of-the-art in lidar SLAM is presented in Section 2.3.

2.2 Graphical Model of SLAM

2.2.1 Preliminaries: EKF-SLAM

Many of the earliest widely successful SLAM algorithms were based on the extended Kalman filter (EKF), forming a class of algorithms referred to under the label of EKF-SLAM. These methods typically solve the *online* SLAM problem, estimating the map and most recent state only.

EKF Formulation

In the EKF formulation of SLAM, the state vector is given by

$$X = \begin{bmatrix} \mathcal{R} \\ \mathcal{M} \end{bmatrix} = \begin{bmatrix} \mathcal{R} \\ \mathcal{L}_1 \\ \vdots \\ \mathcal{L}_n \end{bmatrix} \tag{2.4}$$

where $\mathcal{R}$ is the robot state, and $\mathcal{M} = \begin{pmatrix} \mathcal{L}_1 & \dots & \mathcal{L}_n \end{pmatrix}$ is the set of states of the n current landmarks [12]. Initially, there are typically no landmarks, so $n = 0$ and the state vector is simply $X = \mathcal{R}$. The initial pose is typically used to define the origin, so the EKF is initialised with both the state vector and covariance matrix as zeros, representing certainty in the knowledge of the initial state.

Motion Update

In a typical EKF, the state vector X is updated at each time step based on the control input u, a process known as the motion update. However, the map $\mathcal{M}$ in EKF-SLAM is time invariant, so only the robot state component of the state vector is altered by the motion update. Likewise, the only terms in the covariance matrix affected by the motion update are those relating to the robot state – robot motion provides no information about the covariances between landmarks.

Landmark Observation

The measurement update in EKF-SLAM occurs whenever the robot observes a landmark. If the robot has seen the landmark before, the measurement correction is performed much like a standard EKF, typically affecting all components of the state estimate.

If the landmark has not been seen before, however, the measurement update must add a new state to X, and correspondingly increase the size of the covariance matrix. This step is not present in standard Kalman filter applications. In some cases, such as when bearing-based sensors are used, the first observation of a landmark may only be a *partial observation* – one which does not provide enough information for the observation

model to be invertible. In such cases, a prior term can be used to augment the model and allow the update to be performed similarly to the observable case [13].

Limitations and Computational Complexity

Since the dimension of the state vector X grows with each new landmark, the time complexity of the measurement update step in EKF-SLAM is $\mathcal{O}(n^2)$, growing quadratically with the number of observed landmarks [12]. This update complexity limits EKF-SLAM to sparse maps, meaning poor data associations can have a large impact on the state estimate, and many environments will not provide suitable features at all.

Additionally, EKF-SLAM applies only to the online SLAM problem, since the addition of a new pose to the state vector at each time step would cause the computation time to grow rapidly and without bound [3]. EKF-SLAM is therefore largely limited to environments with sparse, easily distinguishable landmarks. It is also applicable only to scenarios where the most recent state and not the full trajectory estimate are of interest.

2.2.2 Graph-Based SLAM

In the last 20 years, an alternative SLAM approach based on a sparse graphical representation of the posterior has risen in popularity, pioneered by works including Bosse et al. 2003 [14] and Thrun et al. 2004 [15]. This graph-based approach solves the full SLAM problem, exploiting the graph structure to achieve far better overall performance than the EKF-based approaches that previously dominated the field. These ideas follow in the footsteps of earlier work on structure from motion and bundle adjustment in the field of computer vision, which addressed similar estimation problems over many variables using sparse optimisation methods [16].

Factor Graphs and Factorisation

Graph-based SLAM encodes the posterior in a *factor graph*, a bipartite graph representing a factorisation of the posterior density function. Vertices in the graph are either *variables* or *factors*. Variable vertices represent the states being estimated, such as robot

poses and locations of landmarks. Factors in the graph correspond with factors in the posterior distribution. Variable nodes can only connect to factor nodes, and vice versa. [2] In the SLAM context, factors encode information about the relative transform between two states, as well as the confidence in this transform, using a Gaussian model.

For a system with variables X and measurements Z (here we use Z to encompass both odometry and landmark measurements), the posterior is $p(X \mid Z)$. This can be factorised using Bayes' rule into terms that only depend on a subset of the variables. In the factor graph, each factor ϕ_i is a function only of the variables that it is adjacent to – its adjacency set $X_i \coloneqq \mathcal{N}(\phi_i)$. The global factorisation is the product of the factors [17]:

$$\phi(X) = \prod_i \phi_i(X_i) \tag{2.5}$$

This factorisation will form the basis of the inference process described later.

Constructing the Pose Graph

At the start of the robot's journey, the factor graph – often referred to in the robotics context as a *pose graph* – consists of a single node representing its initial pose. As the robot travels, new variable nodes are added periodically (for example, every metre that the robot traverses), separated by factors that store the intervening transforms. These factors may come from visual odometry, point cloud registration, IMU readings, or any combination of these or other methods.

When a new landmark is observed, a vertex is created for it, along with a factor connecting it to the most recent pose. Repeat observations add more factors connecting subsequent poses to the same landmark vertex.

An example of a pose graph is shown in Figure 2.2.

[2] Equivalently to the bipartite graph formulation, one can also use vertices to represent variables only, with edges representing factors. This formulation is often more convenient, and will be used in this thesis.

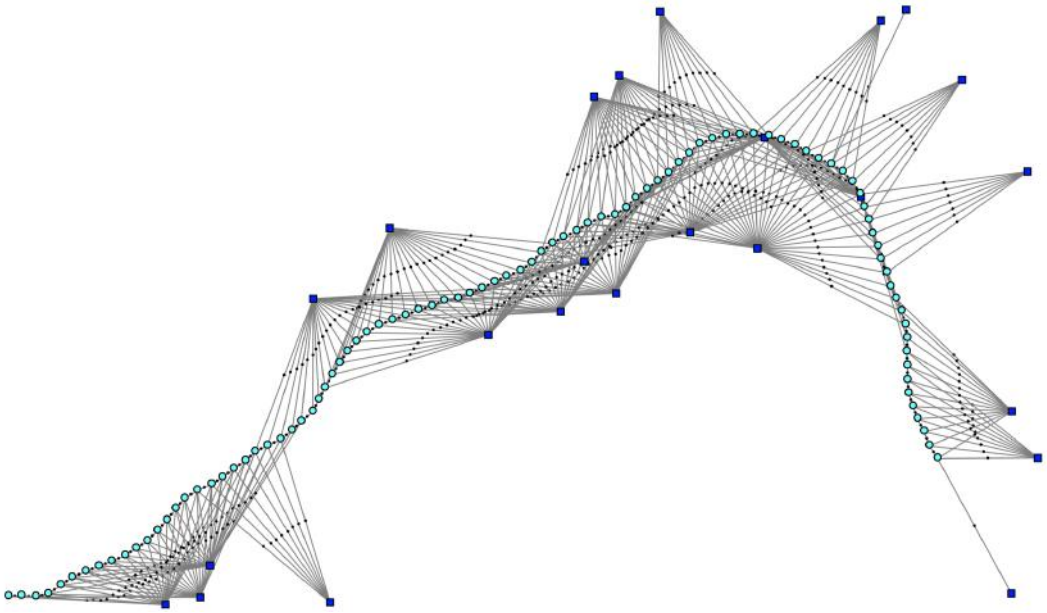


Figure 2.2. An example of a pose graph. Light blue markers represent poses, dark blue observed landmarks, and black dots represent factors. Vertices are rendered at their ground truth 2D locations for visualisation purposes. Image from [17].

Inference from the Graph Representation

The pose graph does not directly provide information about any of the states associated with poses or landmarks – only relative constraints between them are stored in the graph. Graph-based SLAM therefore requires an additional inference process in order to extract the trajectory and map estimates. For this reason, graph-based methods may be thought of as a *lazy SLAM* method, in contrast to the *proactive* EKF-SLAM approach which maintains up-to-date estimates of all states at all times [3].

The inference problem is to extract the most likely state X from the factorisation of $p(X \mid Z)$ in Equation 2.5. This is the MAP estimate, given by

$$X^{\text{MAP}} = \arg\max_X \phi(X) \tag{2.6}$$

$$= \arg\max_X \prod_i \phi_i(X_i) \tag{2.7}$$

Measurements are modelled by Gaussian factors of the form

$$\phi_i(X) \propto \exp\left\{-\frac{1}{2}\|h_i(X_i) - z_i\|^2_{\Sigma_i}\right\} \tag{2.8}$$

where $\|\cdot\|_{\Sigma_i}$ denotes the Mahalanobis distance, a distance metric which is scaled based on the inverse covariance matrix Σ_i^{-1}. The term $h_i(X_i)$ is the measurement prediction, and z_i are the mean values of the corresponding Gaussian factors [17].

Taking the negative logarithm of Equation 2.8, the expression for the MAP estimate in Equation 2.7 becomes

$$X^{\text{MAP}} = \arg\min_X \sum_i \|h_i(X_i) - z_i\|^2_{\Sigma_i} \tag{2.9}$$

The measurement prediction functions $h(\cdot)$ are typically nonlinear, so Equation 2.9 implies that the graphical SLAM formulation using Gaussian factors is equivalent to a nonlinear least squares (NLLS) problem. Methods such as the Gauss-Newton algorithm or the Levenberg–Marquardt algorithm can be used to find the globally optimal estimate X^{MAP}, provided a relatively accurate initial estimate can be computed [17, 18].

Comparison with EKF-SLAM

A key motivator for the adoption of graph-based SLAM methods was their computational efficiency, typically much better than filtering-based methods [19]. Additionally, since graph SLAM methods construct the map using all available data, the accuracy of the linearisation and data association techniques can be improved – that is, data associations and estimates of past states can be revised based on new data [3]. In applications where the aim is to explore and accurately map an environment, this aspect of solving the full SLAM problem is highly advantageous.

However, graph-based methods also have the disadvantage that the size of the graph grows linearly over time, unlike EKF-SLAM in which the dimension of the state vector depends only on the number of observed features. Graph-based SLAM is therefore more

suitable for exploring environments of fixed size, whereas EKF-SLAM can operate over longer time periods without needing to store additional information. In certain applications, it can also be helpful to have the data association probabilities (the covariance matrix) and best state estimate accessible at all times, which is not the case in graph SLAM due to the additional inference step [3].

Thus, while graph-based methods are by no means universally better than EKF-SLAM, they are preferred in many applications due to their improved accuracy and computational efficiency.

2.2.3 Optimisation Frameworks

The benefits of graph-based SLAM over filtering-based methods have made it the dominant paradigm in modern SLAM systems. However, making it computationally viable for large problems has required substantial developments in optimisation techniques.

The basic graph optimisation process operates over the entire trajectory, resulting in increasing computational complexity and therefore degraded performance as the robot explores farther. Incremental solvers such as iSAM [20] and iSAM2 [21] were a key step toward efficient graph-based SLAM, using incremental variable re-ordering and fluid re-linearisation to achieve vastly improved performance [22].

Currently, there are several open-source libraries for efficient graph optimisation which are widely used in the field, including GTSAM [18], Ceres Solver [23] and g2o [24].

2.3 Lidar SLAM

SLAM solutions based on point clouds construct a full map of the environment, using the 3D point measurements themselves as features. These methods typically make use of lidar sensors, which provide either 2D line-scan measurements or full 3D measurements with large fields of view. These sensors are typically much heavier than cameras, restricting their use on some types of aerial drones, but lidar-based SLAM has been demonstrated with great success on land-based robots. In lidar SLAM, the most recent

point cloud can be registered with the previous point cloud to produce transforms that act as odometry, or registered directly with the global map to incrementally update it.

This section provides an overview of the state-of-the-art in point cloud registration, odometry and mapping techniques that form the foundations of lidar SLAM. Finally, it provides an overview of data compression methods for point cloud SLAM, in particular with respect to potential applications in subterranean environments.

Point Cloud Registration

While point cloud SLAM may rely on feature extraction to construct the map, more typically the full point cloud is used. Each time a new point cloud is acquired, it is compared with the previous point cloud in order to extract the spatial transformation between the two corresponding poses, a process known as point cloud registration [25]. An example of point cloud registration is shown in Figure 2.3.

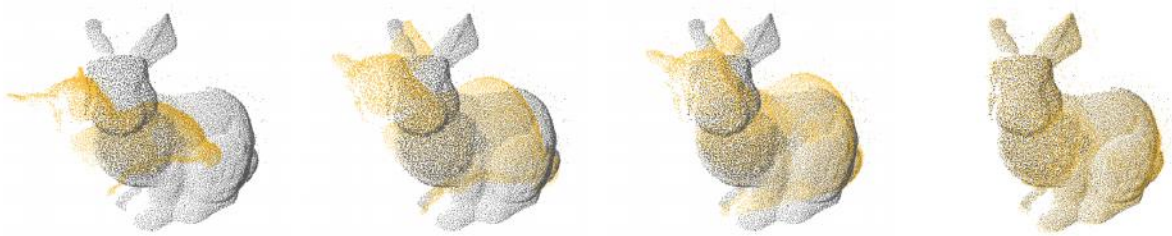


Figure 2.3. A point cloud registration algorithm, starting from initialisation (left), and results after 1, 10 and 20 iterations (right). Image from [26].

Point cloud registration is typically achieved by iterative methods. A popular such method is the iterative closest point (ICP) algorithm proposed by Besl et al. [27], which uses the following process to compute the transform from a source cloud to a reference cloud:

1. For each point in the source cloud, find the closest point in the reference cloud.
2. For some chosen error metric (typically, the sum of squared distances between paired points), compute the transform to the source cloud that will minimise the total error across all pairs.

3. Apply the computed transform to the source cloud.

4. Iterate until either the error or the magnitude of the transforms is sufficiently small.

Many variants of ICP have been proposed over the years, notably the Generalised-ICP (GICP) algorithm by Segal et al. [28], which integrates point-to-point and point-to-plane ICP methods into a unified probabilistic framework. The GICP algorithm maintains similar computational performance to standard ICP, while improving its ability to deal with noise and outliers.

2.3.1 Odometry in SLAM

Odometry, the process of determining the motion of a robot over time, is an essential component of higher-level algorithms for localisation, mapping and navigation. Many methods for odometry exist, each with advantages and failure modes that are highly application-specific.

Wheel Encoders

For wheeled robots, the position of the robot can be estimated based on the rotation of its wheels. Optical encoders are used to measure the incremental rotation of the drive wheels, which is processed to account for the steering geometry of the vehicle [29].

Wheel odometry is a simple, computationally lightweight method which in many scenarios achieves extremely accurate odometry. However, its accuracy is naturally affected by the presence of wheel slip. Scenarios in which the traction of the wheels is poor, such as the wet, muddy or uneven surfaces that are often found in subterranean environments, can easily make wheel encoders infeasible as a sole source of odometry [30].

Visual Odometry

In order to overcome some of the challenges of wheel odometry, visual odometry (VO) is often adopted instead of (or in conjunction with) wheel odometry. Much like VSLAM (dis-

cussed in Section 2.1.3), visual odometry makes use of feature extraction and matching between frames in order to reconstruct the motion of the robot.

While visual odometry is unaffected by the terrain over which the robot traverses – and indeed is applicable to legged and flying robots, whereas wheel odometry is not – it is not a universal solution. Visual odometry is highly reliant on the presence of suitable features in the environment, and can be affected heavily by smoke, fog, blurring and textureless surfaces [9].

Lidar Odometry

Lidar odometry, also known as point cloud odometry, computes the incremental motion of the robot by comparing consecutive point cloud scans. The Lidar Odometry and Mapping (LOAM) system demonstrated by Zhang et al. [31] uses feature extraction to match consecutive point clouds, and is considered the state-of-the-art in lidar-based odometry. LOAM extracts edge and planar points from the scans, matching them at high frequency to approximate the motion of the robot. Many variations on this exist, making use of techniques including continuous 3D lidar sweeps [26], learning-based feature selection [32], fusion with visual odometry [33, 34], or full feature-independent point cloud registration [35].

Compared with visual odometry, 3D lidar makes use of a much larger field of view (often a full 360° horizontally), is agnostic to illumination, and less sensitive to viewing angles [36]. However, featureless or self-similar corridors in which lidar scans are invariant under translations of the robot along the corridor can cause lidar odometry to fail, as no motion can be registered by comparing point clouds in these scenarios. This is often referred to as *lidar slip*. Point cloud odometry also tends to be more computationally intensive than visual odometry.

2.3.2 IMU Integration

Many SLAM systems integrate an IMU in order to improve localisation and sensing performance. Xue et al. [37] demonstrate that the addition of an IMU can assist at

multiple stages of the lidar odometry pipeline, including point cloud correction, ICP initialisation, and state estimation. The LIO-mapping algorithm from Ye et al. [38] demonstrates use of lidar-IMU fusion for improved state estimation, particularly in conditions of fast motion or insufficient features where lidar on its own might typically fail.

Motion Correction of Point Clouds

Widely used 3D lidar sensors such as the Velodyne VLP-16 [39] typically operate at frequencies of 5-20 Hz, acquiring each scan in a circular sweep around a 360° field of view. Robot motion can therefore lead to distortion of the point clouds, which becomes more pronounced at higher velocities.

Motion distortion can be corrected for in real-time if the velocity of the robot is known. Standard IMUs can provide acceleration measurements at frequencies of 100 Hz or higher [37], which can additionally be upsampled and filtered to precisely characterise the motion of the robot during each point cloud measurement [40]. With an accurate motion model, the point cloud can be transformed point-by-point to mitigate motion distortion and improve the overall quality of the measurements. Zhang et al. [31] demonstrate that the inclusion of an IMU in LOAM to correct for motion distortion produces a distinct improvement in performance.

ICP Initialisation for Lidar Odometry

The convergence of iterative closest point methods to a global minimum is highly dependent on the use of a reasonable initial guess for the translation and rotation offsets [27]. The motion model obtained with the assistance of an IMU can provide the initialisation for ICP, using this information as the best prior estimate of the relative transform between the two point clouds. Using this initialisation method, not only is the algorithm more likely to converge to the correct transform, it will also take fewer iterations to do so [37, 41].

Motion Priors

Inertial measurements can also directly inform localisation and mapping, improving the accuracy of the state estimates constructed by vision, lidar, or other sensing modalities. In filtering-based methods such as EKF-SLAM, the IMU readings can be incorporated into the measurement update of the extended Kalman filter, improving the robustness of the state estimate [42].

In the factor graph SLAM formulation, IMU measurements can be added directly to the graph as a factor between two poses [43]. Suppose at time t, a pose graph is updated with a new factor connecting the current pose x_t to the previous pose x_{t-1} using the transform from lidar or visual odometry. An additional factor can be added between x_{t-1} and x_t storing the integrated motion of the robot computed using all IMU readings between times $t-1$ and t. The IMU readings then appear in the optimisation problem which is solved to obtain the map and trajectory estimates.

2.3.3 Lidar Mapping

In graph-based lidar SLAM, each node in the pose graph is typically associated with a point cloud scan; after performing graph optimisation, these point clouds can all be rendered in a single global reference frame to produce a full map of the environment.

Zhang et al. demonstrated with their LOAM algorithm that the point cloud map can actually be used to increase the accuracy of lidar odometry. Where standard lidar odometry relies only on scan-to-scan matching – comparing the latest scan with the previous one – LOAM additionally performs scan-to-submap point cloud registrations for more accurate odometry. Based on the scan-to-scan odometry estimate, LOAM performs another matching step between the most recent scan and the region of the full map with which this scan intersects. In this way, a higher degree of consistency between nearby point clouds is enforced, resulting in more accurate odometry and therefore more accurate localisation and mapping [31].

2.3.4 Feature-Based Approaches

Modern 3D laser scanners have high data rates, capable of registering millions of data points per second [39]. This presents a challenge for real-time SLAM algorithms, particularly in scenarios involving large-scale maps or communication with other agents. On-board computers used on ground and aerial robots contribute significantly to their weight and power requirements, making optimisation of complex and data-heavy operations an important consideration.

Surfels

A method for data compression of point clouds that has seen some interest is the surface element, or *surfel* representation. Although the concept originates as a graphical rendering technique [44], it has been demonstrated as a method for 3D point cloud compression [45] and applied successfully to real-time lidar SLAM [46, 47]. Generically, a surfel is an object associated with a point in 3D space, which may have attributes such as a surface normal (orientation), associated planar region, texture, colour, or others [44].

(a) (b)

Figure 2.4. Two different types of surfel maps: (a) surfels representing covariances of collections of points as ellipses, coloured by orientation [48], and (b) surfels representing planar and non-planar (curved) surfaces [49].

For SLAM applications, surfels have primarily been used to represent planar surfaces [46], curved surfaces [49], or sets of lidar points from within a 3D grid represented

by their mean and covariance [47] – examples are shown in Figure 2.4. The planar surface approaches, while having demonstrated some success in particular scenarios, rely on the presence of regular geometries and planar features in the environment, making them unsuitable for underground exploration. Covariance-based approaches have the potential for applications in more diverse environments, but the current literature is likewise largely focused on indoor or feature-rich outdoor environments. By nature of data compression, any such method of compressing point clouds will not preserve the full accuracy of uncompressed lidar SLAM, but it is largely an open question which compression methods will prove computationally effective in subterranean environments.

SegMatch

An approach to point cloud compression that attempts to reduce reliance on the presence of distinct objects in the environment is the SegMatch algorithm presented by Dubé et al. [50]. The SegMatch algorithm implements the pipeline for point cloud compression and place recognition shown in Figure 2.5, and is described below. This pipeline, while explained here in terms of SegMatch, is common to many other descriptor-based methods for 3D point cloud compression [51].

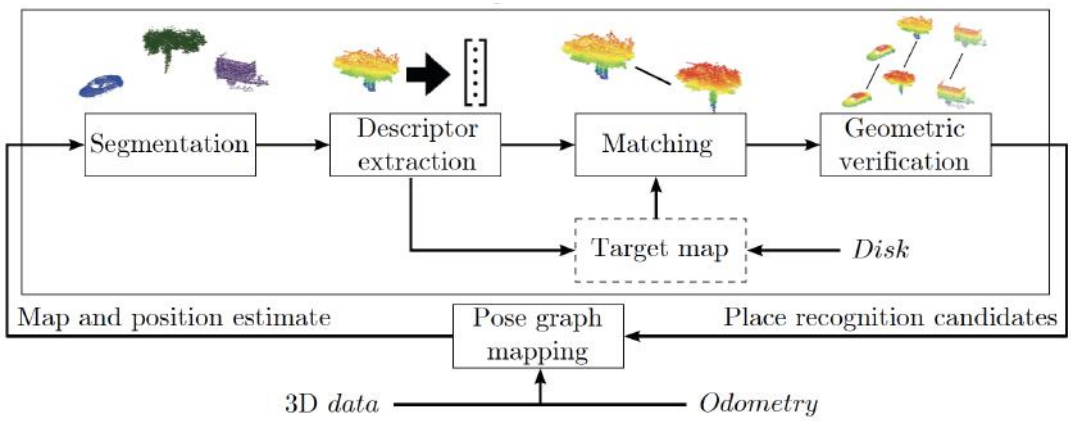


Figure 2.5. The architecture of the SegMatch algorithm. Image from [50].

1. **Segmentation**: SegMatch first defines a cylindrical region, applying a voxel grid filter to remove noise, and then also removing the ground plane. A Euclidean

clustering algorithm [52] is then used to extract a set of point clusters C_i, each with an associated centroid c_i.

2. **Feature extraction**: features are then extracted from each segment, compressing the raw data and building *signatures* that can be used for recognition and classification [50]. Each cluster C_i may have many descriptors computed for it, which are stored in a feature vector $f_i = \begin{bmatrix} f_i^1 & f_i^2 & \dots & f_i^m \end{bmatrix}$.

   Dubé et al. identify two promising descriptors for mapping in unstructured 3D environments:

   - f^1 *Eigenvalue-based*: a set of 7 scalar descriptors based on the eigenvalues of the point cloud cluster are computed as described in Weinmann et al. [53], encapsulating traits such as planarity, scattering and anisotropy.
   - f^2 *Ensemble of shape histograms*: the cluster is encoded in ten 64-bin histograms of *shape functions*, which describe various geometric characteristics of the cluster as per Wohlkinger et al. [54].

3. **Segment matching**: a learning-based approach is used to identify whether two segments correspond to the same full or partial object [50]. A random forest classifier is used to determine correspondence between clusters C_i and C_j based on their feature vectors f_i and f_j.

4. **Geometric verification**: after potential matches between segments have been computed, they are verified using a random sample consensus (RANSAC) method [55]. If a geometrically-consistent cluster of segments is identified (containing at least some chosen minimum number of segments), the transform between the two point clouds is computed based on the set of matched features [50].

In a subsequent paper, Dubé et al. [56] present the first multi-robot system able to achieve collaborative localisation and mapping using 3D lidar data, leveraging SegMatch to do so. Previously, the computational load of performing global lidar data association

in real-time had proved severely limiting. The SegMatch algorithm provides sufficient compression of the point clouds to enable effective information sharing between robots on their limited communication bandwidth.

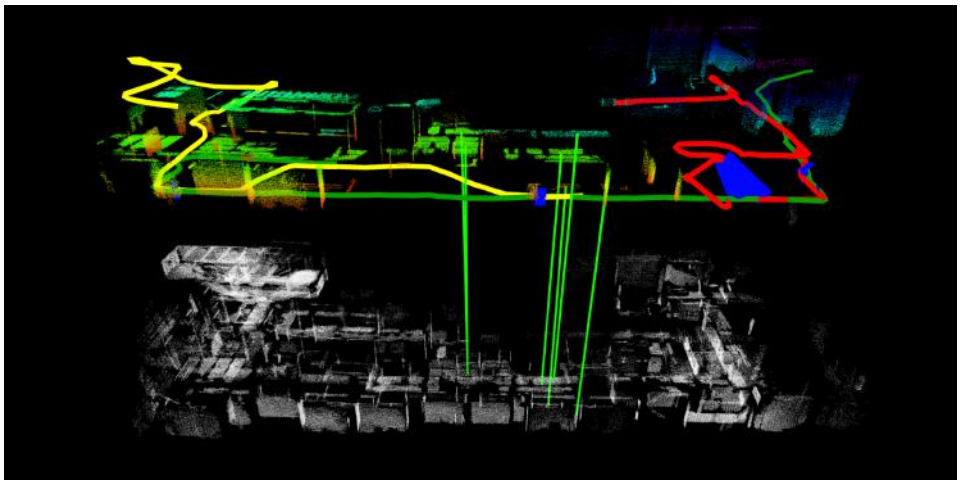


Figure 2.6. A multi-robot SLAM experiment from Dubé et al. [56] using data from a search and rescue operation in the Gustav Knepper Power Station. The trajectories of three robots are shown in green, yellow and red. The target map is shown below in white, with vertical green lines representing inter-robot place recognition through segment matching.

However, the multi-robot system from Dubé et al. is demonstrated in feature-rich environments with many distinguishable objects – experiments are conducted on the KITTI dataset [57], collected by autonomous cars in urban areas, and data from a search and rescue operation in the Gustav Knepper Power Station (Figure 2.6). While significantly different from each other, both datasets encapsulate man-made environments with many distinguishable geometric features. It is therefore unclear how well SegMatch or similar descriptor-based techniques would perform in a more diverse array of subterranean environments.

2.4 Loop Closure in SLAM

Loop closures are an essential component of SLAM systems, particularly in applications with long trajectories and large environments. Without loop closures, sensor drift in-

creases without bound over time. Correctly identified loop closures provide the potential for significant improvements in mapping and localisation accuracy.

This section provides an overview of common methods for detection of loop closures, as well as techniques for improving robustness of pose graph optimisation to spurious loop closures.

2.4.1 Loop Closure Methods

Lidar Loop Closure

Loop closure with lidar data typically operates similarly to lidar odometry as described in Section 2.3.1. A process running in parallel to the main SLAM algorithm compares the most recent lidar scan with earlier scans, using ICP or some variant. If a match between the two point clouds is found, the computed transform between them can be added as a factor in the pose graph.

However, the density of point cloud data makes ICP computationally costly, particularly when the most recent scan must be compared against *many* older scans. On a typical lidar SLAM system, it quickly becomes infeasible to search for loop closures against every stored scan. Methods for overcoming this issue include restricting the search radius in real space, or using compressed point cloud representations [7] such as described in Section 2.3.4.

Visual Loop Closure

Visual loop closure is less straightforward, as the appearance of a landmark typically depends on viewing angle and lighting conditions. A robust approach to visual loop closing must rely on salient features that are identifiable across a wide baseline of viewing angles [6].

Newman et al. 2005 [6] associate visual features with robust *descriptors*, allowing searches for similar visual features during the loop closure detection process. More recent approaches to visual data association are largely based on *bag-of-words* approaches [58].

These associate images with descriptors from a predefined *visual dictionary*, which are stored in a tree structure to allow for fast searches for similar images [59, 60]. The DBoW2 C++ library [61] implements bag-of-words image classification, and is widely used in SLAM systems.

Inter-Robot Loop Closure

In multi-robot systems, loop closures between different robots can occur through direct observation of one robot by another, or through mutual observation of a landmark. This is discussed further in Section 2.5.1 in the context of information sharing between robots in multi-robot SLAM systems.

2.4.2 Consistency Checks and Outlier Rejection

In any graph-based SLAM system, erroneous loop closures can significantly degrade the quality of mapping and localisation, producing factors that result in convergence to incorrect solutions. This is a significant concern in subterranean environments, which often lack distinct features and contain many similar-looking junctions and corridors, making false positive loop closures much more likely.

Consistency with Odometry

A first step toward robust loop closure detection is checking that the loop closure is consistent with odometry [62]. Whenever a new loop closure between poses x_i and x_j is detected, the new factor creates a cycle in the pose graph (shown in orange in Figure 2.7) consisting of all odometry edges except the single loop closure edge. In a perfectly consistent pose graph, the odometry transform T_{ij} between poses x_i and x_j should compose with the loop closure transform T_{ji} to produce the identity [63]. In reality, there will be some error:

$$T_{ij}^{\text{err}} := T_{ij}^{\text{odom}} \cdot T_{ji}^{\text{LC}} \tag{2.10}$$

The odometry consistency check simply determines whether the average translational and rotational errors – distributed over all edges in the cycle – are below some threshold values, and rejects them otherwise.

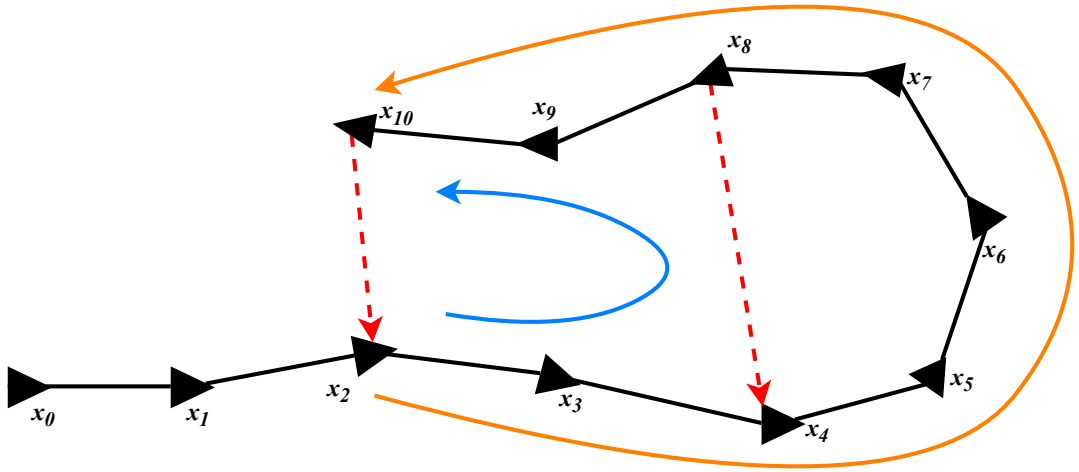


Figure 2.7. Loop closure consistency checking. Black edges are odometry factors, red edges are loop closures. The orange cycle is used for odometry consistency checking, the blue for pairwise consistency.

Pairwise Consistency

Even the most robust loop closure detection algorithms cannot eliminate false positives entirely, particularly in environments with repetitive structures. This is particularly the case as the robot trajectory becomes longer, since the likelihood of false positive loop closures passing the odometry check increases.

A novel approach for dealing with this issue is the Pairwise Consistency Maximisation (PCM) algorithm proposed by Mangelson et al. [62]. Consistency between two loop closures is defined with respect to the cycle in the pose graph created by odometry edges and the two loop closure edges (the blue cycle in Figure 2.7). Similarly to Equation 2.10, the average translational and rotational errors around the loop are computed, and the two loop closures are considered consistent if the values are below some chosen thresholds.

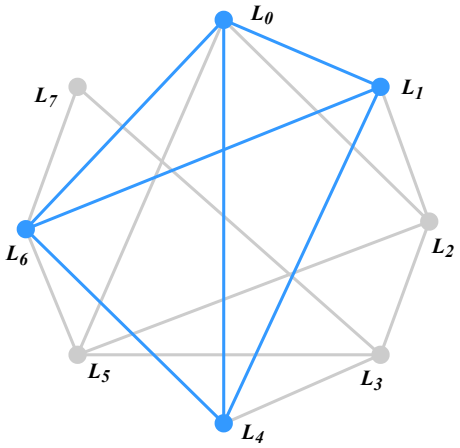


Figure 2.8. A consistency graph for a set of eight loop closures. Each vertex represents a detected loop closure, edges represent consistency between their two endpoints. The maximum clique is a 4-clique, indicated in blue.

The idea of PCM is to determine the largest set of detected loop closures which are *all* pairwise consistent with one another. This is equivalent to the maximum clique problem from graph theory. All detected loop closures are represented as vertices in an undirected graph $G = (V, E)$, where vertices are connected by an edge if the corresponding loop closures are consistent – Mangelson et al. refer to this as a *consistency graph* [62]. The maximum clique problem seeks to find the largest set of vertices $S \subseteq V$ such that for each pair of vertices $u, v \in S$, the corresponding loop closures are consistent with each other. All other loop closures are then regarded as outliers and excluded from the pose graph during optimisation. An example of the maximum clique over a consistency graph is shown in Figure 2.8.

The maximum clique problem is NP-hard [64], and also hard to estimate solutions for. Mangelson et al. leverage a method proposed by Pattabiraman et al. [65] which is optimised for sparse graphs, and due to its simplicity and parallelisability is able to solve the maximum clique problem for PCM in real-time.

PCM is a more powerful form of outlier rejection than the odometry check, as it considers *all* detected loop closures when deciding which to accept as inliers. As more loop closures are detected, provided the odometry backbone of the system is sufficiently

robust [62], PCM is able to revise its estimation of which loop closures are genuine based on all available information.

2.5 Multi-Robot Systems

Compared with single robots, multi-robot systems present the potential for significantly increased capability for subterranean exploration. A team of collaborative robots can explore and map an area much faster and more efficiently than a single robot, leveraging data fusion to improve mapping accuracy and fault tolerance [66]. In regions of restricted communication, robots can relay communications between each other as they move, whereas a single robot must rely on dropping static communication relay nodes. Multi-robot systems also provide redundancy where a single robot has none, a vital consideration in the unpredictable and challenging environments found underground.

This section outlines the primary data association methods for multi-robot SLAM, and provides an overview of collaboration architectures for multi-robot systems.

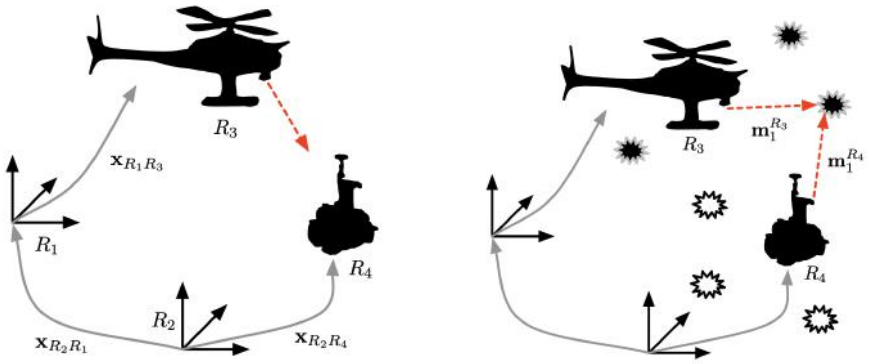
2.5.1 Inter-Robot Loop Closure

The development of an effective multi-robot SLAM system relies on sensible fusion of the data collected by different robots. In the case of graph-based SLAM, the data of interest is stored in the pose graph.

As each robot explores the environment, it constructs a pose graph representing the traversed path. Construction of a unified map requires a mechanism for fusing multiple pose graphs into a single, globally consistent one. This requires associations to be found between the poses of different robots at known points in time.

Vidal-Calleja et al. [67] describe two primary methods for inter-robot data association. First, one robot may observe another directly, in what is referred to as a *rendezvous event* – this gives a new factor between the two robot poses in the respective pose graphs. Second, two robots may make mutual observations of the same landmark, referred to as a

map matching event, which connects the two pose graphs through the shared landmark. [3] These two events are shown schematically in Figure 2.9.



(a) (b)

Figure 2.9. Two different types of inter-robot loop closures: (a) a rendezvous event, where one robot directly observes another, and (b) a map matching event, where two robots observe a common landmark. The relative initial poses are assumed to be known. Images from [67].

Each inter-robot loop closure event results in new factors connecting their respective nodes in the multi-robot pose graph, thereby improving the accuracy of the global map. Techniques such as PCM (Section 2.4.2) may be used to improve robustness to outliers, much like in the single robot case – in fact, PCM was originally developed for multi-robot systems [62].

2.5.2 Collaboration Architectures

Several different system architectures have been proposed to achieve the information sharing described in the previous section, each with their own advantages and drawbacks. These fall into two primary categories, *centralised* and *decentralised* architectures, shown schematically in Figure 2.10. There are many different taxonomies for categorising multi-

[3] Vidal-Calleja et al. also describe a third type of inter-robot loop closure which arises from *absolute* position measurements, such as a GPS fix or landmark recognition with a prior map. However, these do not occur in the GPS-occluded and unmapped subterranean environments that are the focus of this thesis.

robot systems, and network systems in general – in this thesis, we adopt those described by Cao et al. [68] and elaborated upon by Navarro et al. [69].

Within these architectures, further questions arise regarding the specific protocols by which information is shared – in particular, how each robot knows what information to share with each other agent in the system. Optimisation libraries typically achieve best performance when pose graph updates are received incrementally [20, 21], and communication bandwidth in subterranean areas places restrictions on the amount of data that can be transmitted. These considerations and several others affect the choice of architecture and information-sharing protocol, including communication bandwidth, computational power, environment size and required mapping accuracy.

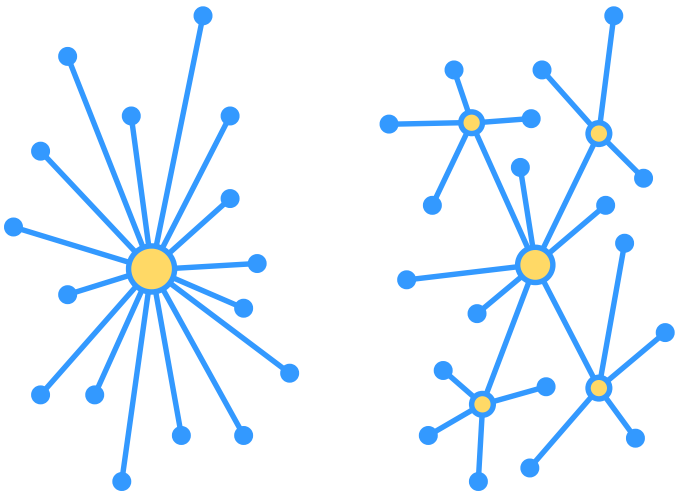


Figure 2.10. Schematic representations of (a) centralised and (b) decentralised communication architectures.

Centralised Architecture

In a centralised system for subterranean exploration, robots do not share information directly. Rather, each robot communicates only with a base station, as shown in Figure 2.10 (a). The base station may be either a stationary computer, or itself on-board a

robot. All optimisation and map fusion is performed on the base station, which shares information as needed with every robot in the system.

The centralised architecture is the most straightforward to implement, as the communication protocols involved are very simple. Each robot has only a single communication channel with the base station, through which it shares its pose graph and lidar scan data. The base station performs pose graph fusion to construct a global map, which can be incrementally updated as it receives data from the robots.

However, this simplicity comes with several drawbacks. The architecture relies on each robot being able to communicate its map back to the base station, as there is no capability for direct communication. It also places a heavy computational load on the base station, which must exchange data with every robot and perform the global pose graph optimisation.

Decentralised Architecture

In a decentralised system, robots are not restricted to communication solely with the base station, and may share information directly with each other. This does not preclude the use of a base station to create a global map, it simply means that the base station may never receive *all* of the information used to create it.

There are two prominent types of decentralised architecture. In *hierarchical* architectures, some robots may act as local base stations, aggregating information from nearby robots and then relaying it to a central base station [68]. Figure 2.10 (b) depicts a hierarchical decentralised system. *Distributed* systems, on the other hand, have each agent in an equal role – in a distributed multi-robot SLAM system, every robot would construct a multi-robot pose graph using information acquired from other robots.

Decentralised architectures are typically considered advantageous over centralised ones due to improved scalability, reliability, and fault tolerance [68]. However, decentralised systems are much less straightforward to implement, requiring much more complex protocols to describe how information should be shared between robots. While

some examples exist of algorithms and protocols for decentralised SLAM [70–74], none have seen widespread adoption or been comprehensively tested in the field.

2.6 Performance Evaluation of SLAM Systems

The development process for any SLAM system must include quantitative performance evaluation, such that the accuracy of the system during its intended usage can be concretely described and compared with other systems. There are a variety of different approaches which can be taken to performance evaluation, the suitability of which may depend on the system itself as well as the environment in which it is designed.

The SLAM problem represents a coupled formulation of the problems of localisation and mapping. The performance metrics by which SLAM systems are typically evaluated are therefore localisation error and mapping error, which are closely coupled by nature of the SLAM problem. A SLAM system that is able to produce low localisation drift over long distances will produce an accurate map of the environment, and likewise an inaccurate map will necessarily imply poor localisation. As such, either metric can be used to holistically evaluate the performance of a SLAM system.

2.6.1 Localisation Error

Localisation error is straightforward to compute and easily interpretable, making it a popular metric for evaluation of many SLAM systems. The localisation error at a particular time is simply the Euclidean distance between the position of the robot as calculated by the SLAM system, and the corresponding ground truth position at that time. Computing localisation error along a trajectory requires ground truth information for that trajectory.

For systems operating outdoors in suitable environments, accurate positioning information can be obtained via GPS. In robotics laboratories, state-of-the-art motion capture systems such as those developed by Vicon [75] and OptiTrack [76] are commonly used to provide localisation ground truth. The camera suites used in these systems require

significant effort to configure and calibrate, but once in place can provide extremely accurate real-time tracking of robots moving within the allocated space.

Neither of these localisation-based approaches to performance evaluation extends well to field testing in subterranean environments. GPS-based localisation is unavailable underground, where the enclosed environment prevents any receivers from obtaining a fix from the required satellites. Motion capture cameras, on the other hand, are costly to purchase, mount and calibrate, making them infeasible for short-term use in non-laboratory environments. Motion capture systems are also ill-suited to large, complex environments, as it becomes difficult to maintain the required lines of sight throughout the environment with a reasonable number of cameras. [4]

2.6.2 Map-Based Evaluation

For some SLAM systems, the performance of the system can be assessed through the quality of the map that it produces. In this case, system performance is evaluated by comparing the SLAM map to a reference map using some chosen error metric.

Unlike in the case of localisation error, there is no universally favoured metric for quantifying mapping error in SLAM systems [77]. Furthermore, the choice of error metric for map comparisons depends on both the available data and the SLAM system itself. For point cloud maps, one technique for evaluation is by matching 3D scans against a reference map generated from CAD data [78]. This type of technique relies on the availability of such a map, which may not exist for certain types of subterranean environments. Some SLAM techniques store only the locations of landmarks, rather than a full environment map – in these cases, localisation error can be computed for landmarks with known locations, though this will typically produce far fewer data points than the localisation-based evaluation approaches.

While map based approaches do have some merits, their reliance on the existence of a reference map limits their utility in subterranean environments. Variants of these

[4] It is worth noting that localisation error can be a powerful and practical metric for testing in *simulated* subterranean environments, as the reference trajectory is readily available in this case.

approaches could, however, be practical in subterranean environments, as shall be discussed in the remainder of this chapter.

2.7 Gaps in the Literature

Research into autonomous multi-robot exploration and mapping of subterranean environments has been quite limited, with the majority of SLAM literature focusing on feature-rich, highly geometric man-made environments. This thesis aims to address the following gaps in the literature for multi-robot SLAM:

1. **Efficient centralised fusion of pose graphs from multiple robots to construct an accurate collaborative lidar map**. Existing systems for collaborative lidar SLAM are quite limited, since in many applications it is preferable to use computationally lighter solutions such as VSLAM. This thesis investigates methods for efficiently fusing lidar-based SLAM data from multiple robots, such that an accurate collaborative map can be produced within reasonable computation time.

2. **Performance evaluation of SLAM systems in large-scale, GPS-denied environments**. Traditional methods of evaluation for SLAM systems make use of ground truth trajectories, typically obtained from GPS measurements or motion capture systems. Neither of these methods is feasible for typical robotics operations in subterranean environments. This thesis aims to develop a method for estimating ground truth trajectories using data that is readily available during subterranean environments, enabling more informative performance analysis of SLAM systems operating in these environments.

CHAPTER 3

DESIGN OF A HETEROGENEOUS MULTI-ROBOT SLAM SYSTEM

Most of the SLAM systems in the literature are designed for deployment in particular environments or scenarios, and make use of the minimum set of sensing modalities required by their design goals. Camera-based SLAM systems have been demonstrated successfully using pure vision [79, 80], visual inertial odometry (VIO) [81, 82] and thermal inertial odometry (TIO) [83], each making use of different methods of feature extraction and ego-motion estimation. In the last five years, systems built around 2D or 3D lidar have also shown increasing promise in a variety of applications [31, 84–87]. Far less common are systems designed to incorporate multiple of these sensing modalities, though there are cases in which sensor fusion has provided demonstrable benefits [33, 34].

This focused design methodology has many advantages, particularly in simplicity of implementation and the computational efficiency that comes with running as few processes as possible. However, narrower focus often results in a lack of adaptability, creating SLAM systems that perform well only in the particular types of environments for which they are designed and optimised.

The work in this thesis is built on top of Large-Scale Autonomous Mapping and Positioning (LAMP), a lidar-based SLAM system developed for the DARPA Subterranean Challenge by the CoSTAR team. As a highly experimental SLAM system, LAMP incorporates a broad set of sensing modalities and technologies, with the aim of achieving good performance in diversely challenging subterranean environments. The specific challenges of these environments are not necessarily known at design time, and as such the system must be designed in a way that makes minimal assumptions about the topology of the environment and the features present within it.

LAMP originated as a fork of Berkeley Localisation and Mapping (BLAM) [88], an open source ROS C++ package for real-time SLAM using 3D lidar. Over the course of its first year in development, LAMP diverged significantly from BLAM, growing by thousands of lines of code in features, extensions and optimisations. In its current form, LAMP supports a diverse array of sensor inputs and configurations, including software to run on-board robots as well as on a base station. At its core, LAMP serves as a back-end for graph-based SLAM, accepting inputs from a variety of different front-end components. LAMP aggregates these different inputs to construct a pose graph, from which an accurate point cloud map of the environment can be generated. It includes several calibration and data reporting procedures, in accordance with the specifications of the Subterranean Challenge.

The rapid development of the LAMP system over the course of the first year of the Subterranean Challenge resulted in software that was unnecessarily complex in both architecture and implementation. As such, debugging, modifying and extending the code with new features became increasingly difficult. This chapter describes the process of redesigning the LAMP system from the ground up, updating the architecture in order to create a modular and extensible platform upon which future work can build.

3.1 Overview

Design of a complex robotic subsystem with many constraints and interfaces presents a variety of challenges in software design and integration. Often the most obvious designs for such systems are ones that achieve the immediate goals, but are not necessarily well-suited to extension and modification for future work.

This chapter describes the design process used to arrive at the final updated system architecture for LAMP. It begins with an overview of the system components and design requirements, which form the constraints for the design problem. This is followed by a description of the conceptual design and implementation of the new system, including justification of key design choices and implementation details. The chapter concludes with a high-level overview of the resulting system and discussion of how the new design accommodates for planned future work.

3.2 Components and Interfaces

This section describes the components and interfaces of the original LAMP system that are preserved in the new architecture.

3.2.1 Odometry Front-End

Subterranean environments present many challenges for robotic perception, and as such any single odometry method is unlikely to be robust. Wheel odometry fails on slippery or uneven terrain, visual odometry can be unreliable in feature-poor or unevenly illuminated environments, and lidar odometry cannot detect motion in featureless, translationally invariant corridors.

LAMP simultaneously accepts data from multiple sources of odometry, fusing information from all of them into a unified odometry estimate. Since each odometry method has different failure modes, failures of each method can be separately recognised and

accounted for to arrive at a unified odometry estimate that is more robust than any of them individually.

Lidar Inertial Odometry

LIO serves as the primary source of odometry for LAMP, using a two-step scan matching process similar to that of algorithms such as LOAM [31]. The lidar front-end obtains an odometry estimate by first performing scan-to-scan ICP matching at high frequency, initialised with a motion prior from the IMU. A secondary localisation produces a refined odometry estimate by performing scan-to-submap matching, constructing a full point cloud map of the environment in the process. This reduces the amount of drift that accumulates compared with pure scan-to-scan odometry, resulting in a more accurate point cloud map [31, 32]. It should be noted that the point cloud map referred to here is *only* generated to improve odometry – the map output from LAMP is created from pose graph optimisation, incorporating the other odometry sources as well as loop closures. Although quite computationally intensive, lidar odometry performs well in a diverse set of challenging environments, without necessarily relying on the prior knowledge of the environment required for appropriate selection of features.

Wheel Inertial Odometry

Wheel encoders are also used to provide another estimate of robot motion. Wheel encoders alone can only estimate motion along a surface, and thus cannot be used to determine full 3D motion. Full 3D motion estimation can be achieved with the addition of an IMU, but is not accurate on rough terrain since significant errors accumulate during inclination changes [89]. Wheel odometry is therefore not preferred as a sole odometry method, and is instead used largely to detect failures in the other odometry methods used by LAMP.

Visual Inertial Odometry

LAMP makes use of VIO provided by the Intel RealSense Tracking Camera T265 software suite [90], to provide an additional layer of robustness to the system odometry in

cases where LIO or WIO perform poorly. Visual inertial odometry is a computationally lightweight odometry solution that performs well in many different environments.

3.2.2 Pose Graph Optimiser

LAMP is a graph-based SLAM system, constructing a pose graph to encode the measurement constraints for its localisation and mapping estimate as the robot explores its environment. New nodes and edges are added periodically based on distance traversed as computed from odometry. When loop closures are added to the pose graph, an optimiser module computes the pose estimate at each node through a nonlinear optimisation process.

The optimiser used by LAMP is the Kimera-RPGO (Robust Pose Graph Optimisation) library [91]. Kimera-RPGO is built on top of GTSAM [18], an open-source library that provides an extensive framework for pose graph optimisation. The nonlinear optimisation problem is solved using the Levenberg–Marquardt algorithm.

3.2.3 Loop Closure Modalities

Recognising previously visited locations becomes more difficult in repetitive, feature-poor environments – this is true for robots for much the same reasons as for humans. LAMP supports multiple independent loop closure modalities, each of which may be effective in some types of subterranean environments.

Lidar

Point cloud scans provide a robust method for loop closure detection that is not necessarily reliant on feature extraction, although descriptors can be used if appropriate. In order to check for loop closures between two scans, ICP is performed, initialised using the transform between them from the current pose graph. If the final alignment has an error below the set threshold, a loop closure factor is added between the corresponding pose graph nodes using the transform computed by ICP.

Visual

Identification of visual features can provide a powerful and computationally lightweight method for recognising previously visited locations. LAMP accepts inputs from a visual loop closure detection module, which makes use of a bag-of-words approach (Section 2.4.1) to associate locations visited by the robots.

Unlike lidar-based loop closure, visual loop closures of this type do not natively compute the transform between the two locations as output. While the transform can be extracted from the visual data, LAMP simply uses the visual loop closure pipeline to *detect* loop closures. Once a visual loop closure is detected between two locations, ICP is applied to the two corresponding point clouds to compute the transform, which can then be incorporated into the pose graph as a loop closure factor.

Costmap

LAMP also supports an experimental method of loop closure detection through *costmaps*, 2D representations of the robot's surroundings. Costmaps are extracted from point cloud scans, and used for traversability analysis and local path planning. Each pose graph node has an associated costmap frame that is stored by the system. These costmaps are compared through 2D descriptors, with loop closures reported if there is sufficient confidence in the match. Similar to visual loop closures, costmap matching does not provide an accurate 3D transform, so after detection of a costmap loop closure the transform is extracted using ICP on the corresponding point clouds.

3.2.4 Landmark Observation Front-Ends

The perceptual challenges of subterranean environments make it difficult to map them accurately, with localisation drift becoming significant as the robots traverse farther.

Observations of landmarks can provide valuable information to combat this drift. Repeated observations of fixed landmarks form loops in the pose graph, reducing drift in the vicinity of the landmarks by anchoring the mutual observation poses to a single

value. Observations of landmarks for which prior information is available (position and/or orientation) can also provide a similar anchoring effect.

Reference Frame Fiducials

Mobile robotic systems often require some form of calibration to allow the robot to localise itself with respect to an externally defined coordinate frame. This can be done through a global localisation method such as GPS, or through calibration to some landmark with a known location.

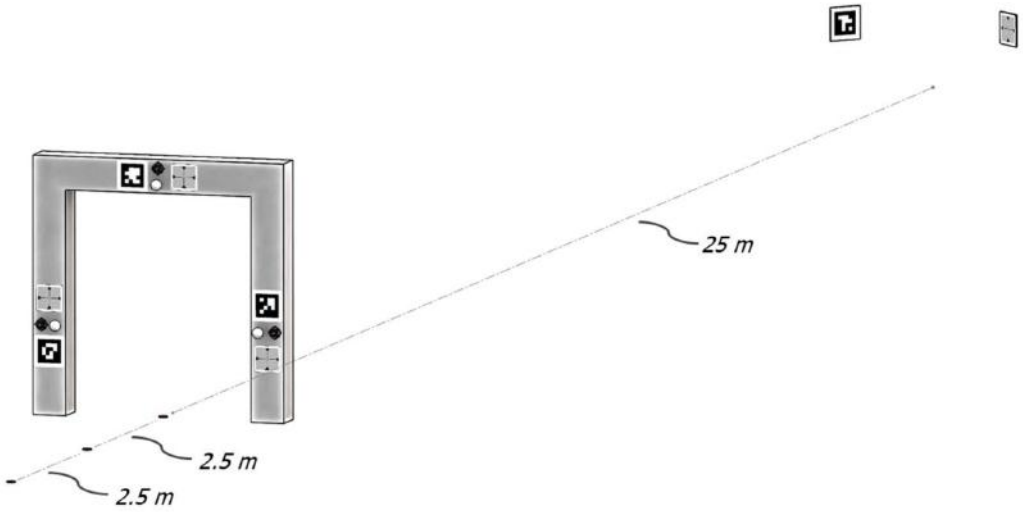


Figure 3.1. The fiducial calibration gate used to define the world reference frame in the DARPA Subterranean Challenge [92].

In the DARPA Subterranean Challenge, the world coordinate frame is defined by a fiducial gate as shown in Figure 3.1. This gate has several different fiducial markers mounted on it at fixed locations, which are published before the competition. Robots can use any combination of these markers to calibrate their position in the global frame.

LAMP supports detection of two of these markers for calibration. Retro-reflective targets are provided for calibration with lidar, designed to stand out from their surroundings due to their high reflectivity. 2D barcode tags known as AprilTags are also used for camera-based calibration, with the format having been designed specifically to allow for accurate visual localisation [93].

Each of these fiducial marker types are registered by LAMP through the respective input data streams, and added as prior factors to the pose graph using their known ground truth locations. This information establishes the robot's initial position with respect to the global coordinate frame defined by the fiducial gate.

Artifacts

Points are awarded in the DARPA Subterranean Challenge for detection and accurate localisation of artifacts in the environment. These artifacts take a variety of forms as shown in Figure 3.2, each presenting different challenges for detection. Some artifacts, such as the backpack and fire extinguisher, are readily detected purely with visual cameras. On the other hand, the survivor and gas vent have heat signatures which lend them to detection by thermal cameras; the mobile phone emits a WiFi signature; and the gas leak can be tracked using a CO_2 sensor.

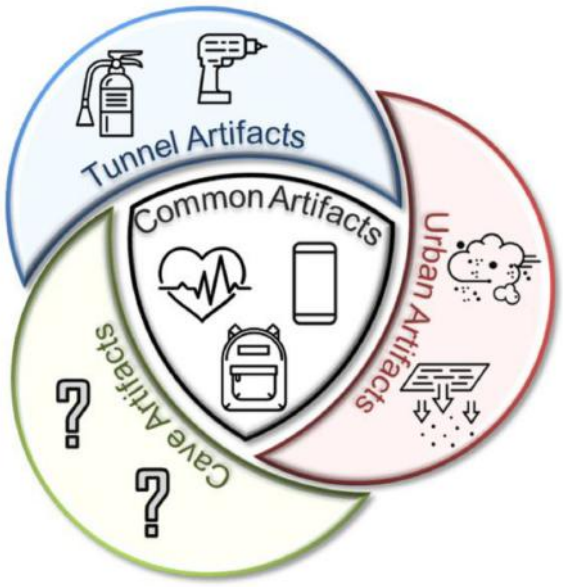


Figure 3.2. The artifacts used for scoring each circuit in the DARPA Subterranean Challenge [92]. The artifacts are: (Common) a survivor, a mobile phone and a backpack; (Tunnel) a fire extinguisher and a drill; and (Urban) a CO_2 gas leak and a heated vent.

Detection of all artifact types is handled by an artifact detection front-end, which accepts inputs from all of the required sensors. This front-end includes an artifact reconciliation process which identifies when multiple observations correspond to the same physical artifact. The positions of reconciled artifacts relative to the robot are passed to LAMP through a single interface, and subsequently added to the pose graph.

Ultra-Wideband Beacons

In order to aid with localisation, each robot carries an ultra-wideband (UWB) radio. At strategically chosen locations, such as large junctions, the robots drop UWB beacons, each of which continuously emits a signal with a unique signature. The receiver on-board the robot can use this signature to detect the range to each beacon, accurate to within a few centimetres [94].

Similarly to artifacts, UWB data is processed in a front-end module which passes data to LAMP as factors for the pose graph. Thus far, all pose graph factors discussed in this thesis have been constraints on the robot pose, specifying a transform with position and rotation information. [1] The addition of UWB data requires a *range-based* factor, which specifies a constraint on the range of the detected beacon from the robot but provides no information on the direction. A single range measurement is never enough to fully constrain the position of the robot, particularly while it is in motion, but through multiple measurements over time the robot can be localised with respect to a UWB beacon in a conceptually similar process to multilateration. The range factor is provided natively by GTSAM, allowing UWB data to be incorporated into the pose graph optimisation problem.

The use of UWB beacons provides yet another mechanism for loop closures and reduction of localisation drift, increasing the accuracy of the whole map, particularly in situations where odometry drift is large.

[1] In some cases, such as detection of the laser fiducial plate, only the relative position is computed and not the rotation. In these cases, the factor is stored with arbitrary rotation components, each with infinite covariance.

Ground Truth Observations

In scenarios where ground truth observations of a robot are available, they can be used to provide substantial improvements to localisation and mapping accuracy. A ground truth data collection tool that can be conveniently used in the field is the total station, an electronic surveying instrument commonly used by civil engineers. In the DARPA Subterranean Challenge context, total stations placed in the staging area can be calibrated to the world coordinate frame using the reflective tags on the fiducial gate. A prism can then be mounted on the robot for the total station to observe, allowing the position of the robot to be measured extremely precisely from anywhere within line-of-sight of the total station. LAMP provides an interface through which such measurements can be incorporated into the pose graph in order to improve localisation accuracy.

Total stations can also be used to obtain ground truth locations of artifacts placed in the environment. Chapter 4 of this thesis develops a method for applying total station measurements to approximate the full trajectory of the robot, enabling improved analysis of the localisation and mapping performance of the system.

3.3 Design Requirements

The primary aim in redesigning the LAMP architecture is to improve the performance and usability of the system, not only at the time of development but over the subsequent years of the development lifetime. This requires consideration not only of the present capabilities of the system, but of the broader possibilities for capabilities that might be added in the future, including ideas which may not even have been proposed at design time.

It is worth noting that the work described in this chapter is a redesign and re-implementation of the *system architecture*, not of of each of its components. As a software package, LAMP has dependencies upon dozens of other packages, including ones for the components described in Section 3.2 and many others. These other packages implement such processes as point cloud filtering and mapping using an octree representation, merg-

ing of pose graphs, and visualisation of mapping output. While some of these packages may require minor modifications to their interfaces for the upgraded architecture, their internal implementations shall remain largely unchanged.

3.3.1 Software Design Goals

At its core, LAMP is a software package that is responsible for localisation and mapping on a robotic platform. The design goals described in this section are therefore informed heavily by established software design principles, although some do have additional motivations or considerations that are particular to robotics.

Modularity

Modular code is a keystone of effective software design, and is especially important in a system such as LAMP with dozens of conceptual components and complex interactions. In this context, modularity refers to separation of components or subsystems into discrete software modules which *encapsulate* their internal behaviours. This is closely related to the *single responsibility principle* from object-oriented design, which states that every module should have sole responsibility over a single part of the overall functionality [95].

These principles encourage software in which all code related to a single behaviour is located in one place, and interfaces between modules are minimal. This has a variety of benefits [96], including:

- **Reduced code duplication**: any functionality that is used multiple times is implemented as a reusable function or module. This also reduces development time by removing the need for re-implementation of similar code for slightly different purposes.
- **Substitutability**: since all behaviours are implemented in modules with simple interfaces, it is straightforward to replace a module with an alternate implementation.

- **Ease of debugging**: errors are typically localised to a single subroutine or function call, rather than arising from complex interactions between components.
- **Parallel collaboration**: since modules are decoupled from each other and depend only on their interfaces, multiple modules can easily be developed and tested in parallel.

Extensibility

One of the primary design challenges for LAMP is in the requirement to support interfaces with a large number of other processes. As outlined in Section 3.2, the system fuses multiple odometry inputs, detects loop closures using a range of different techniques, and incorporates landmark observations from multiple sensor modalities. As a system under continuous development with new requirements and challenges arising over time, it is important that the system architecture makes it easily extensible for future development.

An example of the importance of extensibility for LAMP is in the interface with the artifact detection modules. In the first year of the competition, the five available artifacts (the Common and Tunnel artifacts in Figure 3.2) could be detected effectively using visual cameras, thermal cameras or WiFi. With the subsequent addition of the Urban circuit artifacts, in particular the gas leak, the artifact detection pipeline has a new requirement to handle inputs from a CO_2 detector. The system was therefore designed not purely for the existing artifact detection sensors, but with a general framework for artifact detection that could handle inputs from any sensor through a shared interface. Under this design, artifacts detected by the new sensor are handled differently by the artifact detection front-end, but received by LAMP through an interface that is agnostic to the sensor from which the data originated, meaning that no changes to LAMP are required for the addition of the new sensor.

Designing the entire system for extensibility emphasises the use of shared interfaces for similar components. Modular design also promotes extensibility of baseline functionality, since any behaviour that needs to be updated is implemented within a single module, and only loosely coupled to the rest of the system.

Efficiency

As a component in a robotic system, LAMP produces outputs that are consumed in real-time by other subsystems. The robot pose estimated by LAMP is used by the state estimation subsystem to maintain an accurate estimate of robot motion. Likewise, the autonomy system uses the pose graph to plan exploration of the environment. These dependencies require LAMP to publish messages at a consistent frequency, without allowing latencies to accumulate in the output. This requires *efficient* software design, where efficient in this case means that the core LAMP processes are lightweight and do not block data from being published to other subsystems.

Efficient design for LAMP is therefore a problem of resource allocation and parallelisation. Processes that occupy significant computation time must be separated into their own ROS nodes or processes, such that they do not slow down or block the main node from publishing output at its nominal frequency.

3.3.2 Issues to Address

In addition to general software design principles, the process of redesigning LAMP aims to address several specific issues with the original version. These aims are:

- **Simpler interfaces to the pose graph**: the original LAMP system uses a single monolithic module to handle every interaction with the pose graph. The result of this is extremely convoluted and unintuitive code, in which the logic for modifying the pose graph is extremely complex due to interactions between callbacks from different front-ends (odometry, artifacts, UWB, etc.).

- **One clear source of truth**: early versions of LAMP struggled with delays appearing in the output due to the computational time occupied by loop closure search and pose graph optimisation. This was dealt with by running two parallel instances of LAMP on the robots, one with loop closures on (and therefore the potential for delays) and another without. The two graphs were periodically merged to ensure consistency. However, the existence of multiple pose graphs gave rise to many

software bugs, wherein a module would attempt to access a pose graph node or edge that existed in one graph but not the other. Reduction to a single internal pose graph per robot would significantly simplify interactions between modules within LAMP.

- **Separation of robot and base station**: LAMP provides software to run onboard the robots, as well as on the base station. The original LAMP system has the code for these two components implemented inline, separated only by conditional statements within function calls. This system would benefit significantly from clear separation of the code that is unique to one component, and consolidation of the common elements.

In general, the upgraded LAMP system will also aim to make much stronger use of C++ object-oriented design to improve the usability of the code. Better use of class inheritance in particular will allow for intuitive sharing of functionality between derived classes without duplication of code.

3.4 System Design and Implementation

This section presents the updated design for the LAMP architecture, describing in detail the reasoning behind the major design choices. The design incorporates all elements of the LAMP system described in Section 3.2, guided by the design principles from Section 3.3.

3.4.1 Conventions and Terminology

Pose Graph Keys

The conventions used in LAMP for handling the pose graph and point clouds are developed from BLAM [88], the localisation and mapping library from which LAMP was originally built. These conventions are in part motivated by compatibility with the GTSAM library.

Each robot operating with LAMP is assigned a unique prefix character (e.g. `a`, `b`, ...), which is used to label all of its odometry nodes – that is, pose graph nodes that represent poses of the robot, rather than landmarks. Odometry nodes for each robot are numbered sequentially starting from zero, increasing as the robot explores (`a0`, `a1`, `a2`, ...). These prefix-number combinations are referred to as *keys* in GTSAM.

In LAMP, each odometry node key is associated with a point cloud scan at its time of creation – these scans are referred to as *keyed scans*, since they can be looked up based on their pose graph key. These keyed scans are the *only* scans that contribute directly to the final point cloud map produced by LAMP.

In addition to its odometry prefix, each robot has a unique artifact prefix (e.g. `l`, `m`, `n`, ...), which is used for pose graph nodes corresponding to landmark observations. These are again numbered sequentially in order of observation, so the landmarks observed by the robot with key `b` might be labelled `m0`, `m1`, `m2`, ...

Through these conventions, the origin of each node in the pose graph is clear from its unique GTSAM key, and any information associated with it may be indexed by this key.

UML Diagrams

In this chapter, some C++ inheritance relationships are described through Unified Modelling Language (UML) diagrams. In order to maintain readability and relevance, not all class methods and variables are shown in the diagrams. For base classes, the main methods and member variables are shown. For derived classes, only the main member variables and ROS publishers/subscribers are shown.

3.4.2 Commonalities: Robot and Base Station

LAMP as a system supports centralised multi-robot SLAM through a base station, and provides separate software components for the robots and the base station. In principle, the base station software can run on-board a robot as well, but throughout this thesis the base station shall be considered as a stationary computer located at some staging area at the entrance to the exploration environment.

The two components of LAMP, although substantially different, include many commonalities in functionality and implementation. In the updated architecture, they are therefore derived from a common base class, as shown in Figure 3.3. All functionality that is common to the robots and base station is therefore implemented in the base class, through which it is accessible to both derived classes.

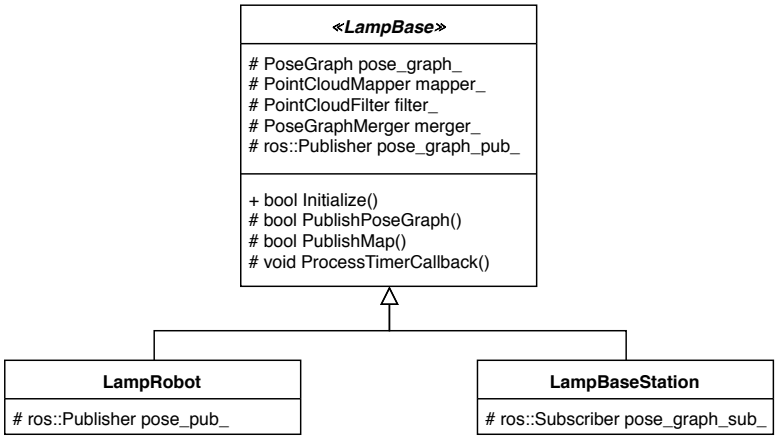


Figure 3.3. UML diagram showing the core classes for LAMP, and their main class variables and methods.

The robot and base station LAMP components both serve the fundamental purpose of maintaining and updating an internal pose graph, from which the map of the environment is constructed. In each case, this pose graph is built through incremental updates, which are received through ROS subscribers. When new data is received, it is stored in a temporary buffer. Each LAMP node has a regular timed callback in which all buffers containing new data are processed and cleared. This processing results in new factors to add to the pose graph, which is subsequently published for pose graph optimisation.

Each instance of LAMP interacts with a corresponding instance of the pose graph optimiser, which is identical between robots and the base station. Once the optimiser has found a solution for the pose graph nodes, it publishes these back to LAMP, which then merges the new data into its internal pose graph to obtain the most up-to-date representation.

3.4.3 LAMP: Robot Component

The robot component of LAMP implements the core functionality of a SLAM system, incorporating data from a variety of sources into a single pose graph representation. This section provides an overview of the design for robot LAMP, with reference to the system diagram in Figure 3.4. All components shown in blue are contained within the main ROS node of the robot LAMP system. The primary interfaces implemented by this node are with sensor front-end modules, a pose graph optimiser node, and a loop closure detection node.

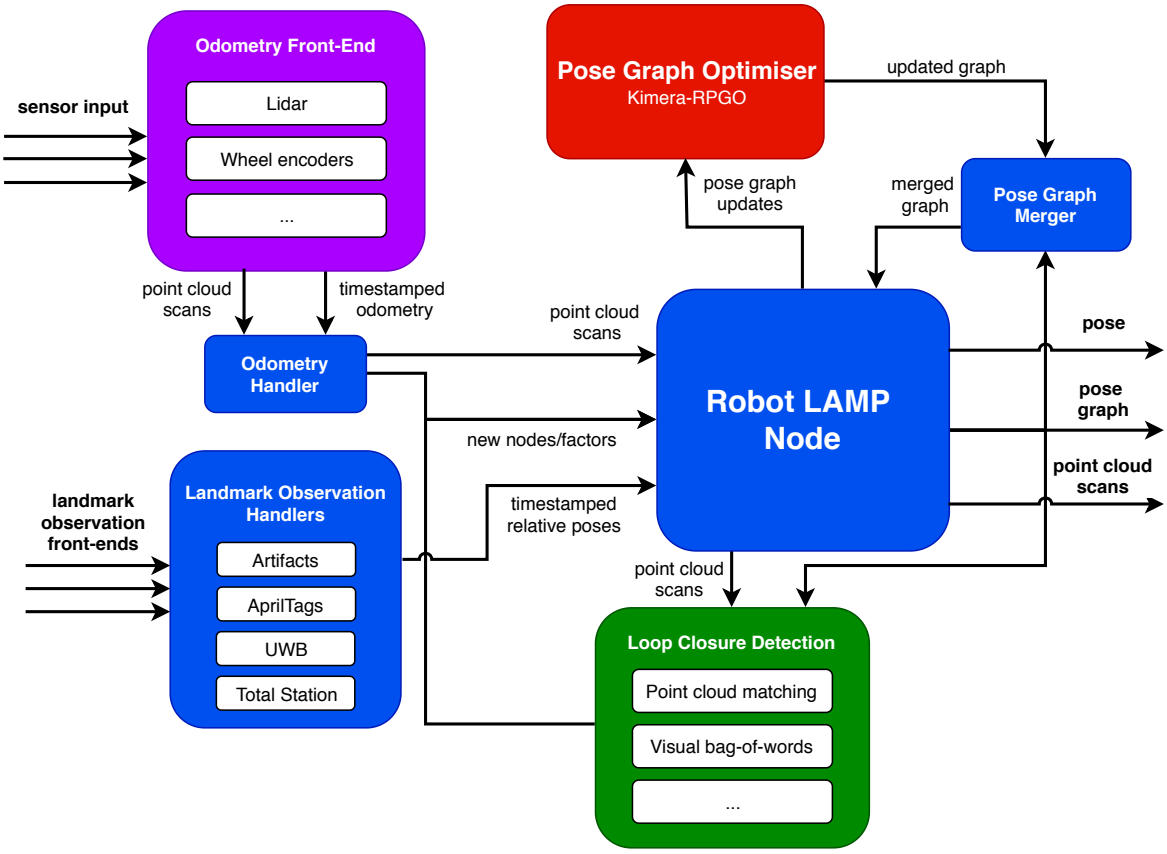


Figure 3.4. System diagram for the core components of robot LAMP.

Data Handlers

All data received by LAMP is passed through modules called *data handlers*, which share a common interface with LAMP as defined in a base class.

The concept of a data handler provides a unified method for LAMP to add new information to the pose graph, by periodically querying each handler for new information which is returned in a common data format. The data handlers themselves therefore simply serve the purpose of receiving data from front-end modules, and converting them into the standardised format required by LAMP.

Figure 3.5 shows the inheritance structure of the data handler classes – each implements the interface with LAMP defined in `LampDataHandlerBase`. This layer of abstraction reduces the complexity of LAMP itself, since it requires no direct knowledge of how raw data is processed in the front-end. LAMP simply creates an instance of each data handler class, and periodically invokes its `GetData` method to retrieve new nodes and edges for the pose graph. When this method is called, a flag may also be set by the handler which indicates that a loop has been added to the graph, which is used by LAMP to determine when pose graph optimisation is required to update the state estimate.

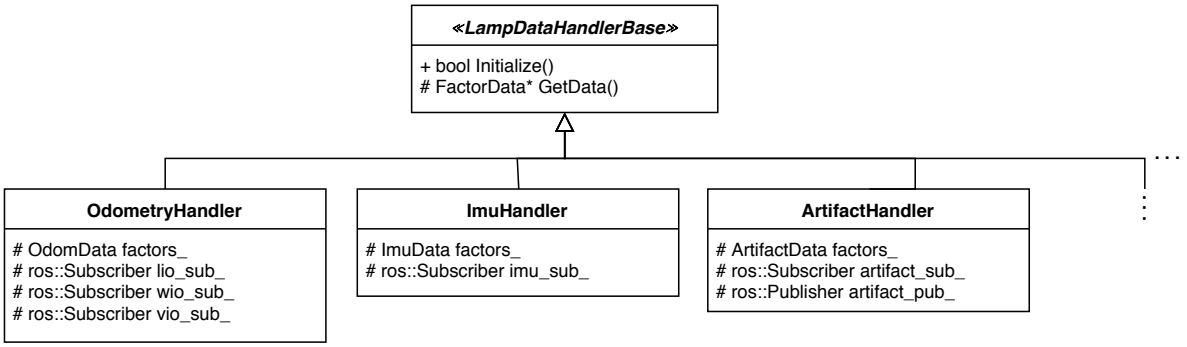


Figure 3.5. UML diagram of the data handler classes. Similar derived classes exist for each front-end interface, as well as for handling pose graphs received at the base station from the robots.

Pose Graph Optimiser

Nonlinear optimisation over pose graphs can be a computationally intensive process, particularly as the robot explores farther and the size and complexity of the graph increases. For this reason, the optimiser is implemented in a separate ROS node from LAMP as shown in Figure 3.4. This means that the nonlinear optimisation runs in a separate process, preventing it from blocking the real-time operation of the LAMP node.

The LAMP node and the optimiser node exchange data through two interfaces. When LAMP adds new loop closures to its pose graph, it sends the graph to the optimiser. Once the optimiser has computed an estimate of the values at each node, it sends these values back to LAMP. This new data from the optimiser is used to update the internal pose graph through a merging process that incorporates the new information.

Loop Closure Detection

The current iteration of LAMP supports loop closure detection based on lidar and visual data. Since many of the processes required by these two detection methods are common, they derive from a common base class `LoopClosureBase` as per Figure 3.6. The base class implements generic functionality such as subscribing to the pose graph from LAMP, triggering a search for loop closures, and publishing the detected loop closure pose graph edges. Each derived class implements its own logic for searching the pose graph for loop closures, as well as subscribers to additional data required from LAMP.

Searching for lidar-based loop closures is computationally demanding, since each iteration of the search performs ICP scan matching on multiple pairs of point clouds consisting of thousands of points each. Similarly to the pose graph optimiser, loop closure detectors are therefore implemented as distinct ROS nodes in order to avoid blocking the main LAMP process, as shown in Figure 3.4. These heavy computational requirements also mean that the module must be selective in its loop closure search – checking for loop closures between *every* pair of pose graph nodes would be impossible to run in real time. This search algorithm is explored in greater depth in Chapter 5.

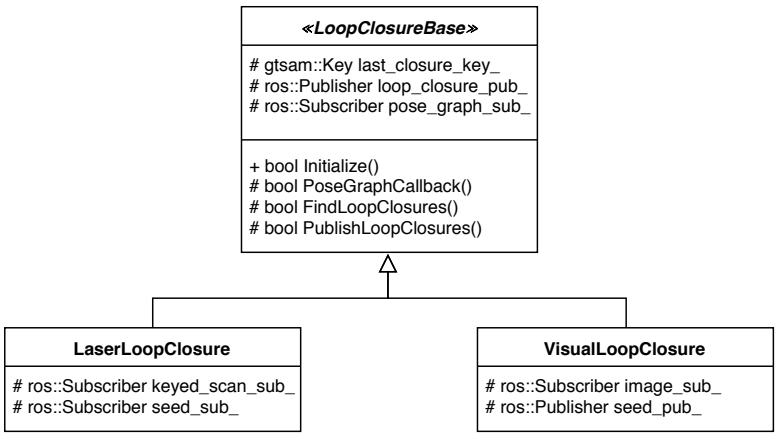


Figure 3.6. UML diagram of the loop closure detection classes.

Visual loop closure detection through bag-of-words approaches is a more lightweight process. However, as implemented in LAMP, the visual loop closure module only *detects* loop closures between two pose graph nodes – it does not compute the relative transform between them. The visual loop closure node therefore publishes a suggestion (a *seed*) of two pose graph nodes which may represent a loop closure. This seed is received by the laser loop closure node, which uses ICP on the corresponding point cloud scans to compute the 6D transform.

The result of this design is that the majority of loop closure code is implemented in the base class. Each derived class implements only modality-specific internal logic for loop closure detection, and only the lidar loop closure module publishes new edges for LAMP.

System Output

The robot LAMP node publishes three primary types of output – the pose graph, keyed scans, and the robot pose.

The current pose is computed by retrieving the most recent odometry node pose from the pose graph, and composing this with a transform calculated from the subsequent odometry measurements. This is published at a high frequency of approximately 10 Hz using a timed ROS callback.

The pose graph is published whenever new nodes or factors are added to it, which can be at up to a maximum frequency of approximately 1 Hz in typical operations. There are two separate ROS publishers on the robot for the pose graph – an *incremental* publisher and a *full* publisher. The incremental publisher only publishes *new* nodes and factors, which are transmitted by radio link to the base station where the full pose graph is reconstructed. The base station performs its own optimisation, and thus does not rely on updates to the state estimate from the optimiser on the robot. This ensures that the communications load on the system does not increase during operation. The full publisher publishes the entire graph, including information from the optimiser, for other subsystems on-board the robot to use. Each keyed scan is only published once, at the same time as its corresponding node is published for the first time.

The information published by LAMP is used by the autonomy subsystem on the robot for real-time path planning. Reliable publishing of the pose and pose graph is essential to ensure that the autonomy subsystem is operating with up-to-date information, allowing it to produce the desired behaviour.

3.4.4 LAMP: Base Station Component

The primary role of the base station LAMP software is to aggregate information from all robots into a single, unified SLAM estimate. The architecture of base station LAMP, shown in Figure 3.7, bears many similarities to the robot LAMP system discussed in Section 3.4.3.

Data Handlers

The data handlers concept introduced in Section 3.4.3 is applied in the base station software to manage incoming data. The base station receives two primary types of data for its pose graph – incrementally published pose graphs from the robots, as well as input from the operator. These interfaces are each implemented in a derived class as shown in Figure 3.8, where `LampDataHandlerBase` is the same base class used for the data handlers on the robot.

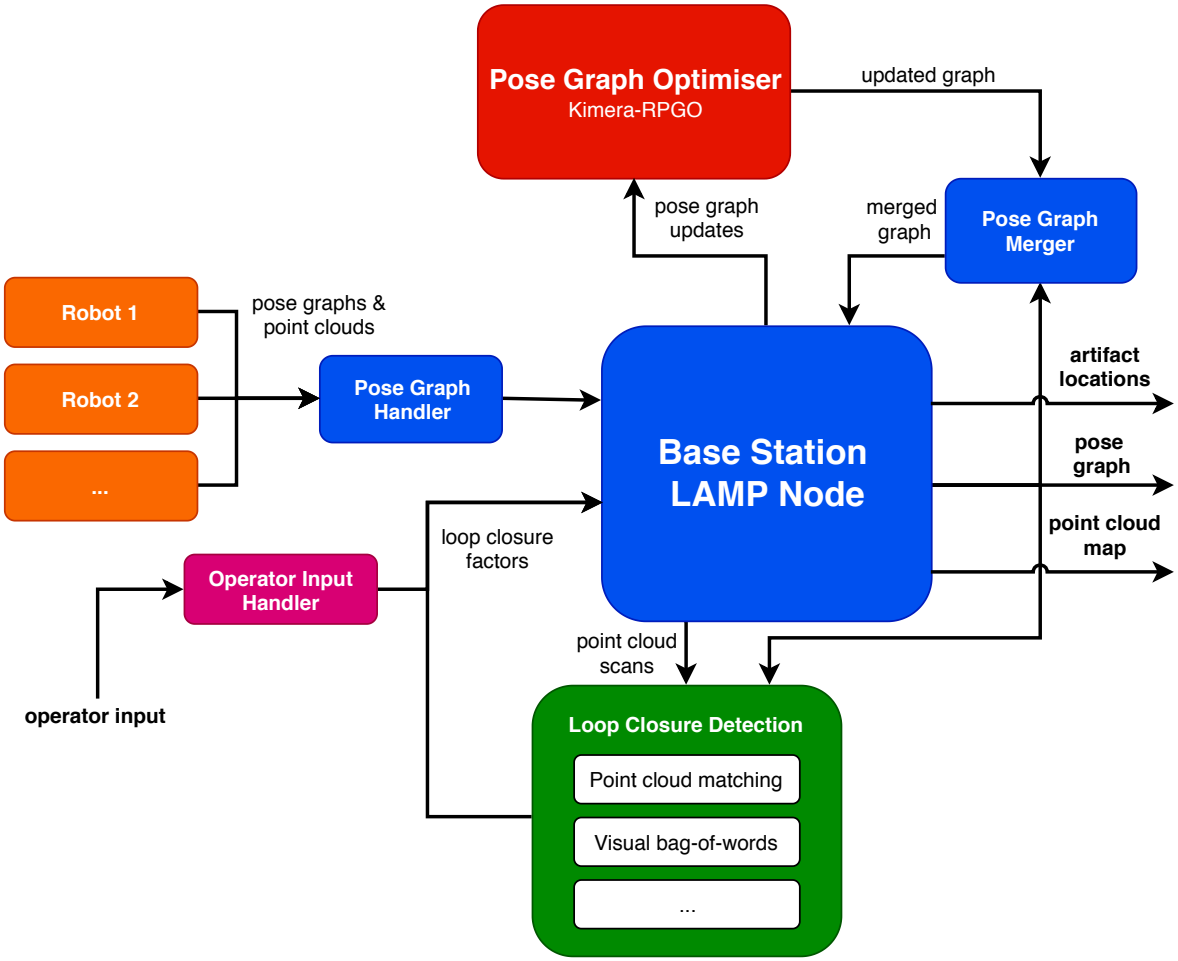


Figure 3.7. System diagram showing the core components of base station LAMP.

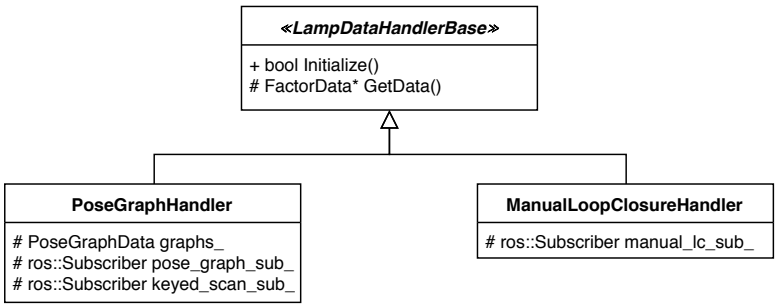


Figure 3.8. UML diagram of the data handler classes for the base station. The base class is the same as that for the robot data handlers in Figure 3.5.

The pose graph handler subscribes to the incremental pose graphs from each robot, and adds them to a single pose graph on the base station. It also subscribes to the

corresponding keyed scans – the point clouds associated with each node – from which it can construct a point cloud map of the environment. Since each robot's pose graph nodes are identified by a unique prefix character, it is clear on the base station which robot each node belongs to. When the received data includes loop edges – either single- or inter-robot – the pose graph optimisation flag is set, triggering an update of the state estimate.

The operator can also provide input directly to the base station pose graph in the form of *manual loop closures*, which are used to correct the pose graph in cases where automatic loop closure detection has failed. These are specified by a pair of pose graph nodes which the operator identifies as corresponding to approximately the same location, but may not appear so in the map due to localisation drift. These are processed through the manual loop closure handler, which converts the incoming ROS message into the common `FactorData` format for LAMP.

Pose Graph Optimiser

The base station runs its own instance of Kimera-RPGO, the same optimiser as used on each robot. This instance processes a pose graph containing information from multiple robots, but the optimisation process is unchanged by this, and the interfaces with the LAMP node are as described in Section 3.4.3.

Loop Closure Detection

The loop closure detection architecture on the base station is largely the same as that on the robot, with several key differences.

The computer at the base station is not constrained in size and weight in the same way as hardware for mobile robots. As such, a much more powerful processor may be used, which allows for much more computationally intensive loop closure search algorithms to run in parallel to LAMP. Additionally, the base station pose graph encapsulates information from *all* robots, meaning that its loop closure detection module can search

for *inter-robot* loop closures, whereas the robot loop closure module can only search for single-robot loop closures.

These additional considerations are addressed by having the base station loop closure modules run a different search algorithm than the corresponding modules on the robots. In this way, the system leverages its additional computational power to find more inter-robot loop closures, thereby improving the accuracy of the unified multi-robot SLAM estimate and improving artifact localisation accuracy. The design and implementation of the loop closure framework for the base station is the subject of Chapter 5.

System Output

The base station LAMP node publishes three primary outputs – the pose graph, the artifact locations, and the point cloud map.

The pose graph and point cloud map are both published for visualisation purposes – the operator is able to view them in rviz, a ROS visualisation tool, in order to monitor the progress of each robot in its exploration. The multi-robot pose graph is additionally used by the autonomy subsystem on the base station, in order to coordinate high-level exploration behaviours of all operating robots.

The base station also publishes estimated locations of all detected artifacts, which update every time a loop closure is processed by the optimiser. In the context of the DARPA Subterranean Challenge, this information is presented to the operator, who can decide to report artifacts to a DARPA server for scoring. In more general applications, the artifacts can represent any features of interest that the robots are tasked with locating. This localisation output – along with the environment map – are therefore the information of highest interest produced by LAMP on the base station.

3.5 Conclusion

This chapter has presented a system architecture design for a modular and extensible multi-robot SLAM framework. This design is implemented as an upgrade to Large-Scale

Autonomous Mapping and Positioning (LAMP), the SLAM system developed for the DARPA Subterranean Challenge by the CoSTAR team.

The system is characterised by its incorporation of multiple sensing modalities, in order to deal with the many challenges of subterranean exploration. The odometry subsystem fuses inputs from lidar, visual and wheel odometry front-ends, each of which incorporates IMU data in real-time. The system is able to seamlessly fuse or transition between different odometry methods, creating resilience in conditions where one or more methods may have failed. Likewise, the landmark detection subsystem accepts inputs from visual cameras, thermal cameras and WiFi for artifact detection, and UWB radio and AprilTags for improved localisation. Each of these front-end inputs, although substantially different in the data they provide and the ways in which that data must be handled, ultimately supply information to LAMP through common interfaces in the new architecture. This simplifies the system immensely, while also providing natural entry points for new sensors or techniques to be incorporated into the system as new requirements arise.

The aim of this work in redesigning the system architecture for LAMP was to provide a modular, intuitive platform for future work on the system. This is achieved through the emphasis on logical separation of components, and interaction through unified interfaces, resulting in the architecture presented in Figures 3.4 and 3.7. The new design represents a significant improvement on the prior system, in which many unrelated systems were tightly coupled and therefore inextensible. As such, it provides a strong foundation for the continued development of the system as its capabilities are expanded to cope with more of the challenges of subterranean exploration.

The work in the remainder of this thesis is built on top of the redesigned LAMP system presented in this chapter.

CHAPTER 4

PERFORMANCE EVALUATION WITH SPARSE GROUND TRUTH DATA

Throughout the development process for any experimental robotic system, access to meaningful metrics for performance evaluation can be extremely beneficial. Without a concrete method for evaluating the performance of the system, it is difficult to determine how it compares to similar systems in the literature, or whether it fulfils its design goals. Moreover, the usefulness of any robotic system depends upon its ability to produce results with known accuracy – without any quantitative measure of performance, the system cannot be specified as suitable to any degree of accuracy for any given task.

In the subterranean context, the ground truth data required to perform traditional mapping evaluation is often extremely difficult to acquire within the practical constraints imposed by the environment. Rapid development of a SLAM system for subterranean exploration therefore requires alternate methods for performance evaluation and benchmarking. This chapter develops a novel technique for performance evaluation of graph-based SLAM systems, using the limited ground truth data that can be acquired during subterranean testing to obtain full trajectory estimates that enable deeper analysis.

4.1 Overview

This chapter proposes a new method for evaluation of SLAM systems, attempting to overcome some of the limitations of existing methods in the subterranean context. The following section provides an overview of the proposed method. Subsequent sections describe the required process for obtaining ground truth data, and the software pipeline to perform the analysis. Finally, the technique is applied to several datasets in order to evaluate the quality and validity of analysis that it can produce.

4.2 A New Method for Performance Evaluation

As noted in Section 2.6, existing techniques for performance evaluation of SLAM systems face many practical limitations in subterranean operations. This section presents an alternate approach to the problem, which aims to provide quantitative evaluation of the performance of a SLAM system through processes that are feasible in the context of subterranean field testing. In particular, the method must not rely on pre-existing reference maps, or on equipment that is costly or time-consuming to set configure.

The proposed method makes use of sparse ground truth information that can be obtained with relatively little setup through use of a total station or similar surveying equipment. This information can be ground truth locations of landmarks that are observable by the robot, or timestamped measurements of the robot's ground truth location taken during exploration. Each measurement is incorporated into the final pose graph as a prior factor with extremely low covariance, thereby "anchoring" the pose graph to these ground truth measurements. The resulting pose graph, while not strictly consisting of ground truth measurements at every node, provides an approximation to the ground truth trajectory by eliminating drift at every anchor point. Figure 4.1 depicts the use of ground truth locations of landmarks to correct the pose graph. The nodes in the original graph can then be compared to the corresponding nodes in the corrected graph to determine localisation error at each node.

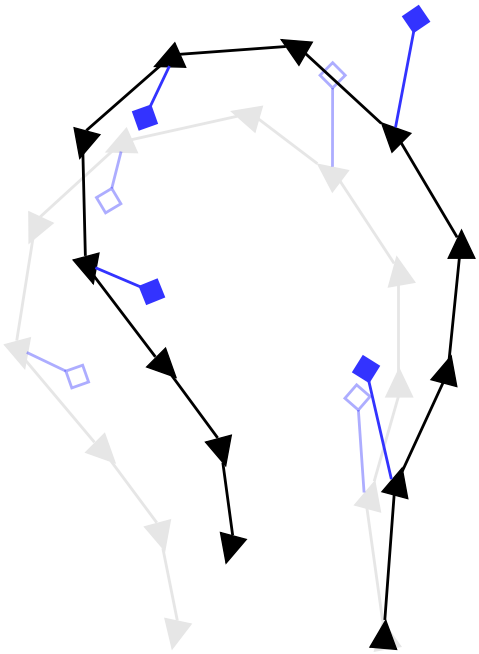


Figure 4.1. Adjusting the pose graph using ground truth locations of landmarks. The greyed out pose graph depicts the state estimate without ground truth information. The darker graph depicts the adjusted state, with the filled blue diamonds indicating ground truth prior factors on the locations of the landmarks.

The following sections explain in detail the process for configuring the ground truth measurements, as well as the implementation of the analysis pipeline in software.

4.3 Ground Truth Measurement

In order to perform the trajectory analysis, ground truth locations of landmarks must be known *in the world reference frame used by the SLAM system*. In the context of the DARPA Subterranean Challenge, this reference frame is defined by the fiducial gate (Figure 3.1), but in general it can be defined through any reference points or features in the environment.

This section will discuss ground truth calibration with respect to a fiducial gate, although the techniques used are applicable to other configurations. The method requires the use of a total station or similar surveying equipment to obtain ground truth

data. Total stations are affordable and relatively straightforward to operate, and as such are ubiquitous in areas such as mining, construction and surveying. They thus represent a practical method through which ground truth data can be obtained for robotics applications.

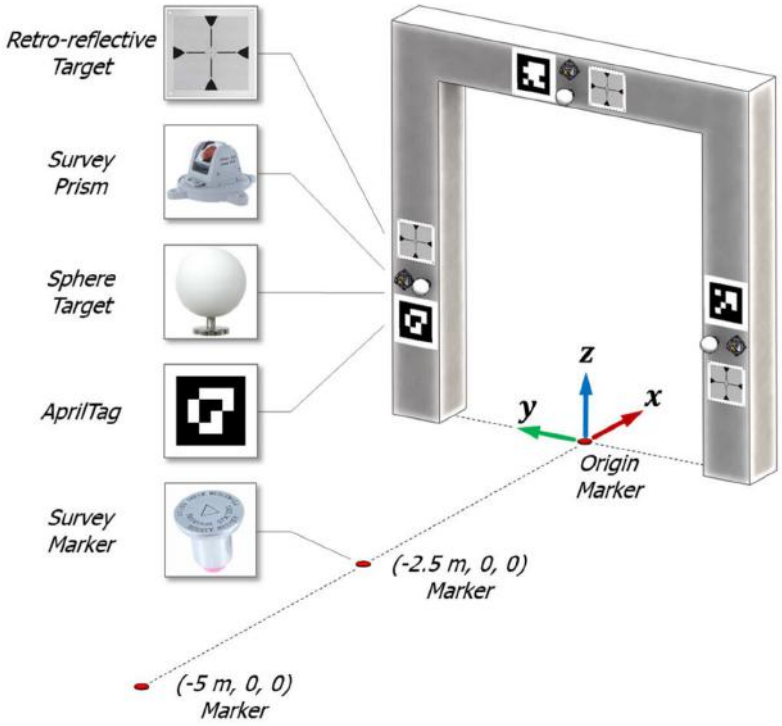


Figure 4.2. Markers on the fiducial gate for the DARPA Subterranean Challenge [92].

4.3.1 Calibration Process

In order to obtain ground truth measurements, the position and orientation of the total station must be determined in the coordinate system defined by the fiducial gate. This is done by taking measurements of the locations of the three survey prisms on the gate shown in Figure 4.2, which each have known locations on the gate and provide sufficient information to constrain the system. With this information, the total station can perform a *resection* operation, which solves an optimisation problem to estimate its position and orientation with respect to the gate.

After resectioning, the total station can measure the locations of any artifacts within its view, and compute their positions in the world frame. Additional reflective tags can also be placed in the environment with their locations measured, such that they can be used for future resectioning of the total station within the same environment. This also allows the total station to be relocated in order to survey artifacts that are not all within line of sight from a single point. Depending on the topology of the environment, several iterations of marker placement and resectioning may be required in order to obtain ground truth measurements for all artifacts.

4.3.2 Calibration from Reference Map

In some subterranean environments, a floor plan or 3D environment model may be available, which allows for a more efficient ground truth determination process. In this case, locations of artifacts are immediately known in the coordinates of the provided map, and the total station is only required to compute the transform from the coordinate system of the map to the world frame defined by the fiducial gate.

This transform can be found by placing several tags at known locations on the map, and measuring their positions from the total station. From these measurements, the total station computes its pose in the map frame, from which it can also compute the transform to the gate frame.

4.4 Data Analysis Pipeline

Once the ground truth positions of landmarks in an environment are known in the world frame, they can be used to construct a ground truth estimate over a full pose graph obtained in that environment. While this process strictly only requires a single landmark to be present in the pose graph, its accuracy improves as more landmarks are observed, and if they are spatially distributed across the trajectory.

The ground truth pose graph can be obtained simply by loading the stored pose graph into the SLAM system, adding a prior factor on each known landmark with

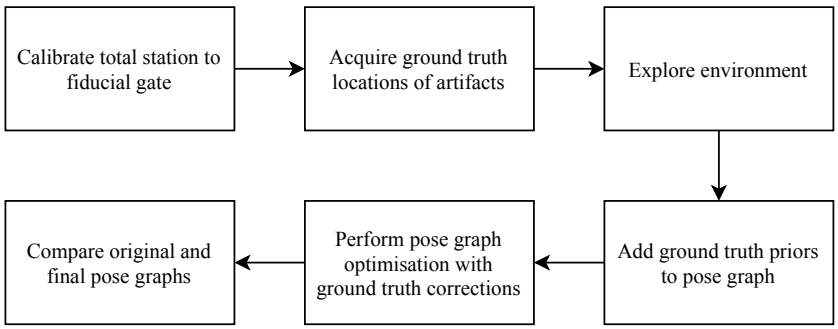


Figure 4.3. Pipeline for ground truth analysis using sparse ground truth data.

its corresponding ground truth location, and performing pose graph optimisation. The prior factors specify the position of each landmark with relatively low covariance, since the locations obtained by surveying are much more accurate than those estimated through SLAM. As a result, these measurements hold much higher weight in the pose graph optimisation process than odometry factors, anchoring the graph to the ground truth locations and producing a more accurate overall trajectory. The full pipeline is summarised in Figure 4.3.

4.5 Results and Analysis

This section presents a series of experiments performed with the new performance analysis technique, in order to establish its feasibility and usefulness, as well as its limitations and other subtleties.

4.5.1 Simulated Data Test

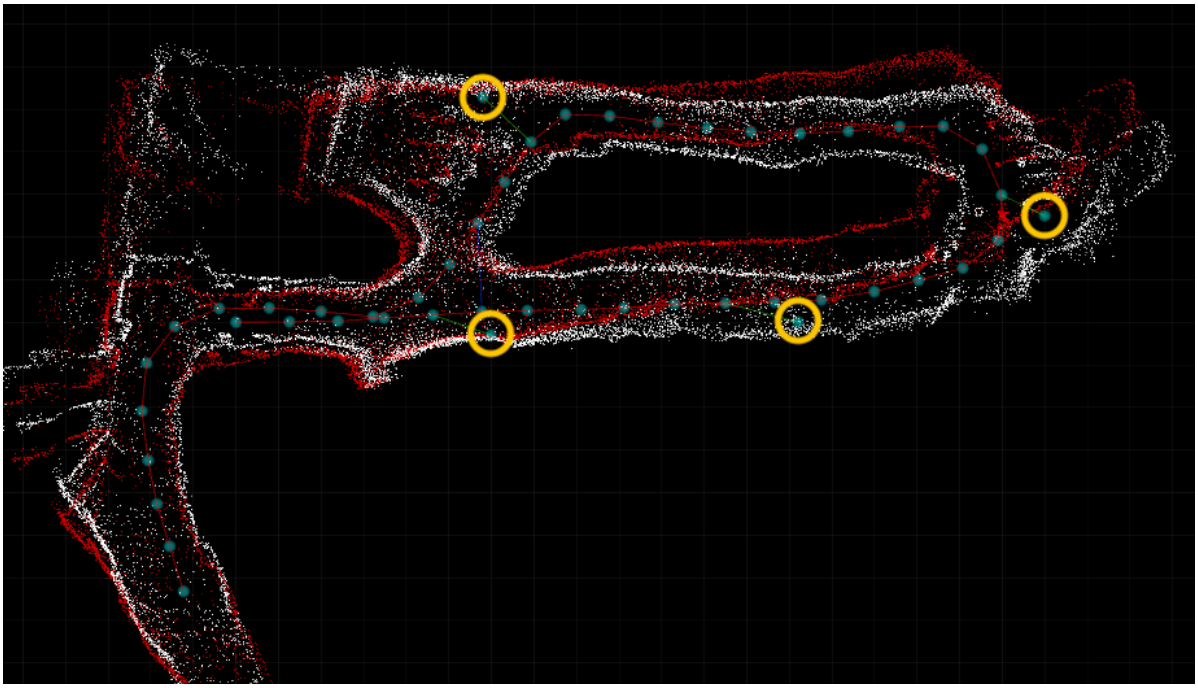
As an initial demonstration that the technique is able to produce reasonable results, this section applies it to a dataset collected at Eagle Mine, a historic gold mine located in Julian, California. Eagle Mine is a small-scale example of the type of subterranean environments in which the technique might be valuable. The trajectory examined in this section is approximately 50 m in length, consisting of a single loop.

At the time when this dataset was collected, the environment was not surveyed to obtain the required ground truth data as described in Section 4.3. This analysis therefore makes use of simulated landmark observations and ground truth locations that were inserted into the dataset after the fact. The aim of this analysis is simply to demonstrate that a reasonable and useful output can be produced by applying the techniques developed in the previous sections.

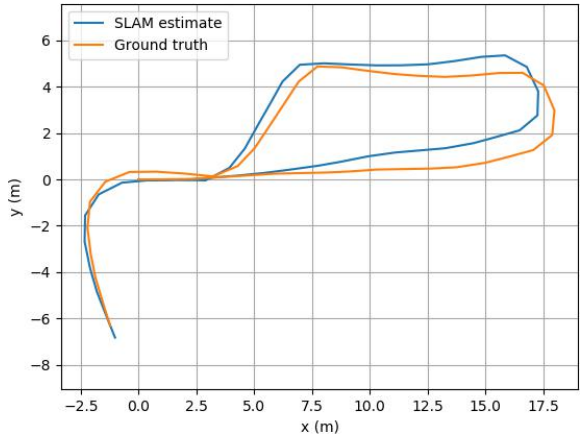
Figure 4.4 (a) shows the result of applying a ground truth correction based on four simulated surveyed landmarks. The original point cloud map is skewed relative to the ground truth map, as might be expected in cases of poor calibration or significant localisation drift. However, the corrected map clearly maintains the overall shape and structure of the original map, without significant evidence of bending or distortion.

Figures 4.4 (b) and 4.4 (c) show examples of the trajectory data that can be extracted after incorporating ground truth data into the pose graph. Figure 4.4 (b) shows a 2D projection of the full measured trajectory and the approximated ground truth, using each pose graph node as a data point. This full-trajectory plot provides a clear visual impression of the localisation drift occurring continuously as the robot moves, despite only making use of four ground truth data points. Figure 4.4 (c) shows the error between the two trajectories at each pair of corresponding pose graph nodes, projected onto each of the coordinate axes. This is overlaid onto an area plot of the computed ground truth elevation (z-coordinate) along the trajectory, as an example of a metric that could be correlated with localisation error. Other metrics such as yaw rate, pitch rate, or velocity could be similarly displayed in order to investigate potential causes of localisation error in the system.

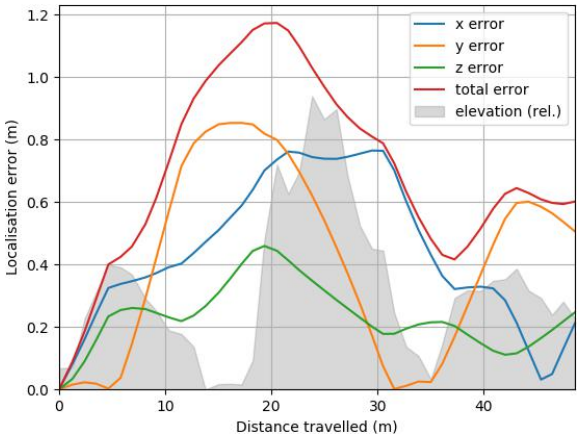
These plots demonstrate that this technique provides access to similar information and analysis as can be obtained through a full ground truth trajectory. By transforming sparse ground truth measurements into an approximation of a full ground truth trajectory, the SLAM system can be subjected to much richer performance analysis than would otherwise be possible.



(a)



(b)



(c)

Figure 4.4. Ground truth analysis using data from Eagle Mine, CA: (a) correction of the point cloud map: the red point cloud is the map obtained with no prior information, and the white point cloud is the result of adding ground truth priors to the four circled pose graph nodes; (b) the trajectories obtained from each pose graph; (c) localisation error in each coordinate axis over the trajectory.

4.5.2 Field Test in Larger Environment

Having demonstrated a proof of concept for obtaining a full ground truth trajectory from sparse measurements, this section applies the technique in a larger environment, using real ground truth data from surveyed landmarks.

The environment considered in this section is the basement level of a building at the NASA Jet Propulsion Laboratory, with an exploration area spanning approximately 80×40 m. This environment was chosen due to its similarity in scale and topological structure to certain types of urban underground environments in which robots might be deployed for exploration. Seven artifacts were placed in the environment to act as landmarks, with their ground truth locations computed from a floor plan of the building.

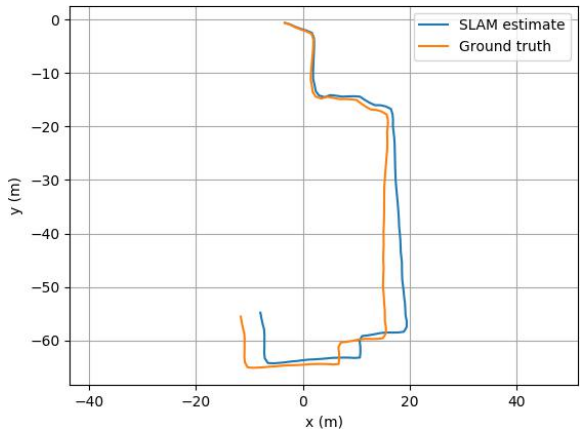
The mapped portion of the environment is shown in Figure 4.5 (a), over a trajectory slightly over 100 m in length. Only three of the seven artifacts were observed by the robot along this trajectory, as indicated in the figure. The fiducial calibration gate was located in the staging area in the lower-left corner of the map. The comparison of the original map (red) and the corrected map (white) immediately reveals that the localisation error is largely due to errors in the orientation estimate of the robot.

Initially, there is a a very small orientation error arising from calibration with the fiducial gate – this error is less than 2°, resulting in localisation error of less than 1 m in the first 20 m of the trajectory (Figure 4.5 (c)). As the robot explores farther, however, not only does this small angular offset cause increasing localisation error, the angular offset itself grows to more than 5°. This is due to additional errors accumulating from the robot's lidar odometry as it moves quickly or turns sharply – robot motion causes distortion of the point cloud scans, which are acquired by the Velodyne lidar at approximately 10 Hz, resulting in less accurate scan matching and inducing systematic errors in the odometry. Additionally, ICP scan matching performs poorly with large rotational offsets [97], such as those encountered during sharp turns, which likely contributes further to the errors.

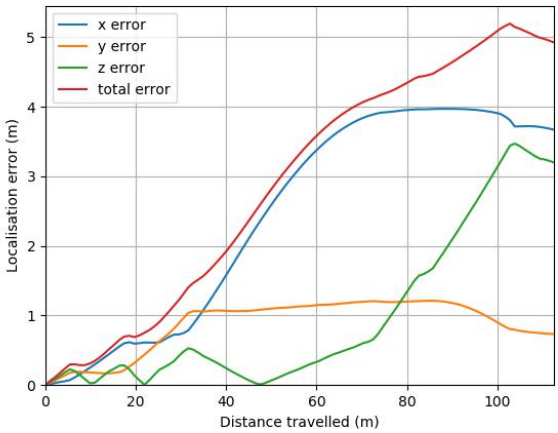
In this dataset, the localisation error exceeded 5 m towards the end of the 100 m trajectory, or ~5% of the distance traversed. However, this section demonstrates that the data in Figure 4.5 serves as a valuable and visually intuitive tool for diagnosing these errors.

The validity of this analysis rests on the fact that the corrected map (shown in white in Figure 4.5 (a)) is much more accurate than the original map. While this is impossible

(a)



(b)



(c)

Figure 4.5. Ground truth analysis in a man-made environment at NASA Jet Propulsion Laboratory: (a) correction of the point cloud map: the red point cloud is the map obtained with no prior information, and the white point cloud is the result of adding ground truth priors to the four circled pose graph nodes; (b) the trajectories obtained from each pose graph; (c) localisation error in each coordinate axis over the trajectory.

to confirm without additional ground truth data, there are several reasons to believe that it is a good assumption:

1. The overall shape of the map appears correct, both in terms of overall structure and consistency with the floor plan. The environment is highly structured, and the map clearly maintains this structure, as seen in the straight corridors and perpendicular walls.

2. After adding the ground truth priors, the artifacts still appear at the appropriate locations on the corrected map. This confirms that the process of incorporating new prior information into the map estimate was successful.

3. The errors computed from this analysis, in particular the localisation drift of ~5 m over a trajectory longer than 100 m, are within the expected bounds for the system at the time when the data was collected.

However, there are also visible imperfections in the process. For example, in the corrected map, not all corridors are *perfectly* straight, and not all junctions are *exactly* 90° – this is visible particularly in the bottom-left quarter of Figure 4.5 (a). This emphasises the fact that the technique remains limited by the sparsity of the ground truth data – in this case, only three landmarks were used to obtain the corrected map, and thus any localisation drift occurring *between* successive landmarks cannot be entirely corrected for. It is likely that within the constraints of the available ground truth data, it would not be possible to improve significantly on this result, and rather that the solution would be to survey more landmarks in the environment at the time of operation.

4.5.3 Challenging Dataset

Despite the promising results presented in previous sections, certain types of errors in the SLAM estimates can cause the technique to produce inconsistent results. This section explores the contributing factors to one failure mode of the technique, using a dataset collected in the same environment as that in Section 4.5.2.

Figure 4.6 shows the result of applying the ground truth correction on the dataset, once again using three landmark observations. Compared with the previous dataset, the

key differences here are that this dataset contains a longer trajectory, and includes loop closures in the pose graph where the robot returns to previously visited areas.

Figure 4.6. The ground truth correction process producing inconsistent results on a dataset collected at the NASA Jet Propulsion Laboratory. The original point cloud map is shown in red, the corrected map in white.

The uncorrected map, shown in red in the figure, appears for the most part to be a reasonable result. However, in traversing the long and featureless corridor indicated on the figure, the robot incurred significant lidar slip (introduced in Section 2.3.1) – that is, due to the lack of features in the corridor, it was unable to accurately register its motion through lidar. Consequently, it underestimated the length of the corridor by approximately 5.5 m. This resulted in inconsistencies in the map near the robot's starting point, where two copies of the same area are visible on the map with an offset of 5.5 m. Upon the robot's return to the starting corridor, LAMP also detected a loop closure, which is incorporated in the pose graph – however, this loop closure proved insufficient to correct for the large lidar slip errors.

When the ground truth correction is applied to this dataset, the inconsistencies in the initial map result in significant distortion in the corrected map, as shown in white in Figure 4.6. This is because although the landmarks are pinned to their ground

truth locations, the pose graph does not behave as a rigid structure in the subsequent optimisation routine – it bends in such a way as to minimise the overall error. In cases where errors in the pose graph are small and relatively uniform, as in Sections 4.5.1 and 4.5.2, the ground truth correction will smoothly transform the map towards the ground truth. However, in this case, the lidar slip creates highly *non-uniform* errors in the pose graph – the factors added in the long corridor represent measurements that are much less accurate than those in the rest of the graph. As a result, the optimisation process no longer preserves the overall shape of the graph, resulting in the distorted map shown in the figure.

Challenging cases such as this could be addressed through the ground truthing technique, or through the SLAM system itself. In the former case, as with the previous dataset, the solution would be the addition of more landmarks with ground truth data. In particular for this case, having a landmark before and after the challenging corridor would likely solve the problem – these two landmarks would constrain the ends of the corridor to their correct locations, allowing the intervening pose graph edges to stretch uniformly without bending. In general, however, it would be difficult to know a priori where such failures might occur in an unknown environment, so a general strategy of placing landmarks at all corners of the map, and near key junctions, would be a reasonable policy. On the SLAM system side, issues of this type could likely be surmounted by computing *more accurate covariances* in the pose graph. The current LAMP system did not accurately capture the high uncertainty in its lidar odometry as the robot drove down the long corridor. Had it accurately incorporated this high uncertainty in the pose graph, this corridor would have been "stretchier" when the ground truth correction was applied, potentially reducing the distortion of the map.

4.6 Conclusions

Performance evaluation is an important part of the development cycle for a SLAM system, and indeed for any robotic system. As well as providing a way to quantify the current

performance of the system, meaningful performance metrics can also provide a deep and intuitive understanding of the behaviour of an otherwise complex and opaque system, providing a boost to ongoing development work.

This chapter has presented a novel method for constructing an approximate ground truth trajectory from sparse ground truth measurements. This technique has the potential to be extremely valuable in subterranean operations, providing access to a full ground truth trajectory which would typically be impossible to obtain within the practical constraints of such environments. It allows for richer analysis of system performance than can be achieved with the sparse measurements alone, providing an easier route to diagnosing and addressing weaknesses in the system.

The method is not without faults, as demonstrated in Section 4.5.3. While valid ground truth trajectories can often be obtained through only a few surveyed landmarks, this is not the case when the errors in the pose graph are significantly non-uniform. In order to improve the reliability of the analysis, landmarks would need to be placed strategically around the environment to ensure sufficient constraints on the ground truth trajectory. Nonetheless, the results presented in this chapter serve as a proof of concept, demonstrating the feasibility of the technique to provide richer performance analysis for SLAM systems that would typically be available for subterranean operations.

CHAPTER 5

MULTI-ROBOT LOOP CLOSURE SEARCH AND DATA FUSION

Loop closure detection is an essential capability for SLAM systems operating in large, complex subterranean environments. Localisation drift inevitably grows worse as greater distances are traversed, resulting in poor localisation and inaccurate or inconsistent maps. These effects become even more pronounced in multi-robot systems, in which additional inconsistencies can arise between the maps created by different robots.

By incorporating loop closures into the SLAM estimation process, many of the errors that arise in multi-robot SLAM can be significantly mitigated. Loops within the trajectory of a single robot help to combat localisation drift – in nominal conditions, drift is removed completely at the location where the loop closure is detected, and reduced to a lesser extent around the entire loop. Loop closures *between* robots are much more computationally demanding to detect, but provide measurement constraints that are essential to achieving effective data fusion and producing consistent multi-robot maps [98, 99].

5.1 Overview

This chapter presents the introduction of a real-time inter-robot loop closure search algorithm to the LAMP system described in Chapter 3. The existing loop closure framework in LAMP, which includes single-robot loop closure search only, is introduced in Section 5.2. The subsequent sections present a generalisation of the existing algorithms to multi-robot systems, and evaluate the results produced by deploying this new algorithm in offline and field testing.

5.2 Single-Robot Local Search

Lidar-based loop closures are detected in LAMP by comparing the point cloud scans associated with two pose graph nodes (referred to as *keyed scans*, as discussed in Section 3.4.1). The comparison is performed through ICP scan matching (Section 2.4.1), which is initialised using the current estimate of the transform between the two nodes in order to improve the likelihood of convergence to the correct local minimum. The loop closure is accepted only if the ICP fitness score is below some threshold. PCM-based outlier rejection (Section 2.4.2) is used to reduce the likelihood of erroneous loop closures negatively impacting the SLAM estimate.

Within this framework, a straightforward loop closure search algorithm would simply compare each incoming keyed scan with all pre-existing keyed scans, reporting back any detected loop closures. However, performing iterative comparisons between point clouds consisting of thousands of points each is a computationally demanding process. Searching for loop closures with *every* previous node would require computation time that grows linearly without bound as the length of the trajectory increases. As such, this approach is not feasible for the real-time operation required by LAMP.

To address the limitations in available computational resources, LAMP uses several criteria to narrow down its search for loop closures.

5.2.1 Search Radius

The first criterion by which the loop closure search is narrowed is a *search radius* r_s. For each new pose graph node, LAMP checks for loop closures only with other nodes that are within distance r_s using the current state estimate. This is shown graphically in Figure 5.1 (a).

The idea behind the search radius, which is typically set to a value of 5-10 m [1], is that it should only ever eliminate nodes where a loop closure *would have been unlikely to begin with*. The following observations explain why this is typically the case for lidar SLAM in subterranean environments:

1. In constrained subterranean environments, lidar-based loop closures can typically only be detected between positions that are at most a few metres apart. This is because if they are much farther apart, the fields of view from the two locations will be substantially different, making it difficult to accurately associate them through scan matching.

2. If two pose graph nodes represent the same location, they cannot be arbitrarily far apart according to the current state estimate even if no loop closure has been detected. The estimated distance between the two nodes represents the amount of localisation drift that occurred in the intervening journey – for example, if the robot travelled 100 m, the nodes should be at most a few metres apart.

The implication of these two observations is that even *if* LAMP did search for loop closures between every pair of nodes, they would almost always only be found between nodes that were already in close proximity to each other. As such, the fixed search radius provides improved computational efficiency with little to no reduction in the number of loop closures that the system detects.

[1] These values are suitable for exploration of an environment where the robots can each traverse several hundred metres, based on the typical amount of localisation drift incurred by LAMP. In larger or more challenging environments, a larger search radius could be beneficial.

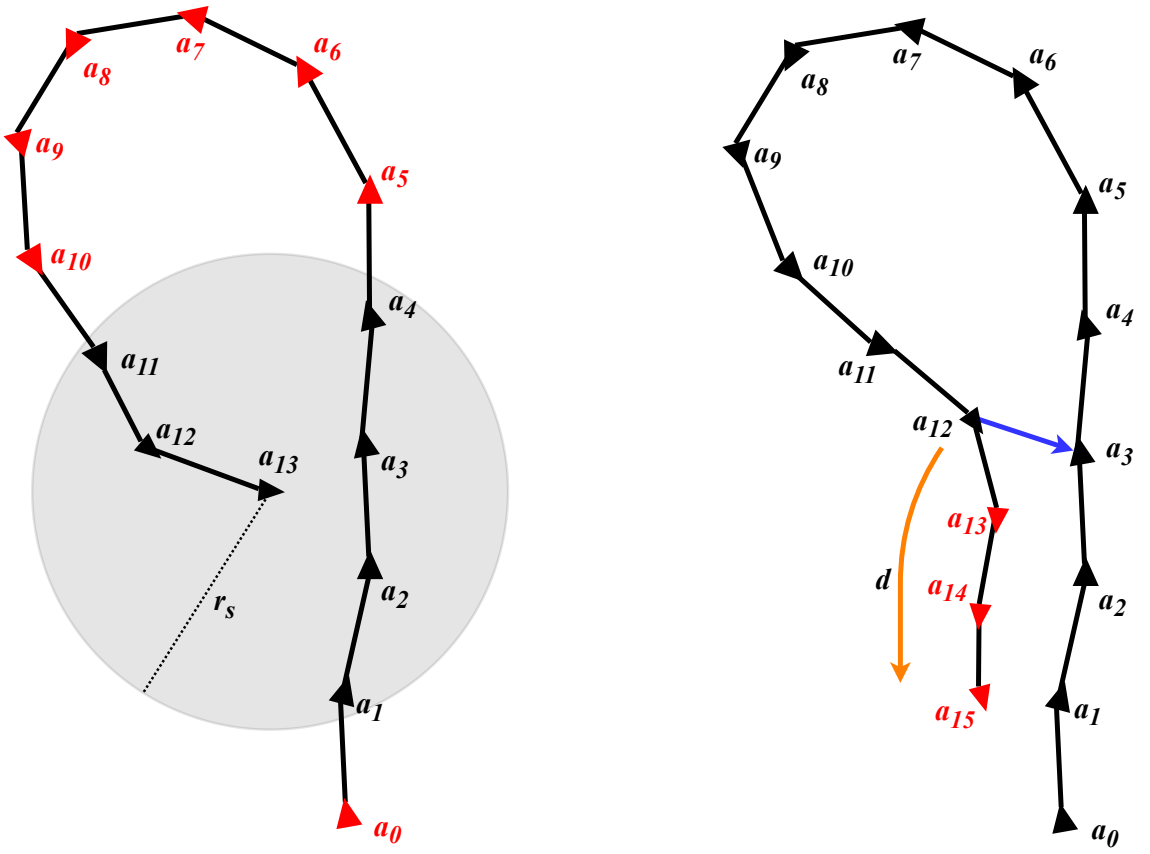


Figure 5.1. Loop closure search criteria in LAMP. (a) Nodes outside of the local search radius r_s, shown in red, are excluded from the search at the current node. (b) After finding a loop closure, new nodes found for the following distance d of ground traversed, shown in red, will be excluded from the loop closure search.

5.2.2 Distance Before Reclosing

The second criterion used to improve the efficiency of loop closure search is a *distance before reclosing* d. After a loop closure is detected, LAMP will no longer search for new loop closures as new nodes are added until the robot has travelled distance d from the most recent loop closure. This is shown graphically in Figure 5.1 (b).

This condition is enforced for two primary reasons. Firstly, it prevents the system from repeatedly detecting similar loop closures – if the robot recognises that the current

node is at a similar location to a previous node, it will most likely detect a loop closure at the next few nodes as well, since it will still be in close proximity to the previously visited location. While these loop closures will also be valid, they would not add significant new information to the optimisation problem. Secondly, the distance before reclosing – which is typically 10-20 m – helps to reduce computational load. If distance before reclosing is not used, and a robot travels through an area it has previously visited, then it will detect loop closures *every time* a new node is added to the graph (i.e. approximately every 1 m traversed). As a result, the pose graph will be almost constantly re-optimised, which occupies a lot of computation time with little benefit to the accuracy of the SLAM estimate. As such, enforcing a minimum distance that the robot must travel before searching for new loop closures helps to lighten the computational load with minimal effect on accuracy.

5.3 Inter-Robot Local Search

This section introduces inter-robot loop closures to the LAMP system, performed in a centralised manner on the base station which receives data from all robots.

The core implementation of loop closure search on the base station extends from the existing single-robot loop closure software, which can run on-board each robot or on the base station. The criteria described in the previous section are applied here as well to improve efficiency. The first criterion, the search radius r_s, extends naturally to multi-robot scenarios with little modification. However, the concept of distance before reclosing d is less straightforward, since all robots in the system travel independently of each other, and loop closures may be registered between *any* pair of them. However, it is important to still have such a concept, or else common scenarios in which multiple robots traverse the same corridor would quickly overload the base station computer with loop closure searches and optimisations occurring in rapid succession. The aim of this section is to develop a reasonable generalisation of this concept to inter-robot loop closures, which produces similar benefits without loss of good loop closures.

As an example, suppose three robots, a, b and c, are operating in a single LAMP system. If robots a and c traverse the same corridor (simultaneously or separately), it is undesirable for LAMP to detect inter-robot loop closures at *every* node of each robot, as these will contain redundant information and result in unnecessary computation. This scenario is shown schematically in Figure 5.2 – in practice, dozens or even hundreds of loop closures could be detected if two robots traversed a common path only 20-30 m long. On the other hand, suppose a loop closure is detected at some point between robots a and b. If robot c then passes through a location that either or both other robots visited, new loop closures should be added irrespective of the recent loop closure between robots a and b, since this would represent completely new and useful information. The new algorithm must deal with each of these cases in a reasonable manner.

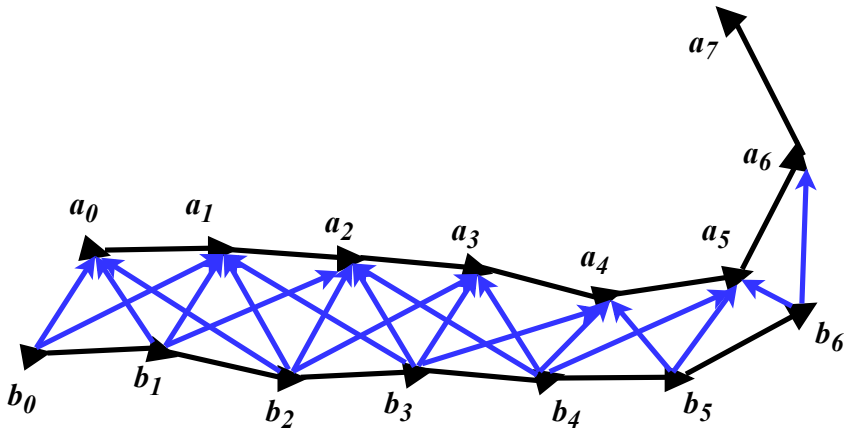


Figure 5.2. Detection of excess inter-robot loop closures when no controls on loop closure frequency are in place. This scenario represents a corridor that was first traversed by robot a, then by robot b – blue arrows represent loop closures.

This is achieved by maintaining an independent record of the most recent loop closure between *each unique pair* of robots in the system, stored in a *distance matrix* $\mathbf{P}$:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & & \\ \vdots & & \ddots & \vdots \\ p_{n1} & & \cdots & p_{nn} \end{bmatrix} \tag{5.1}$$

P is a $n \times n$ matrix, where n is the number of robots in the system, and each element p_{ij} stores the most recent pose graph key of robot i for which a loop was closed *with robot* j.

For example, if the first inter-robot loop closure is detected between pose graph nodes a_{10} and c_{25}, where a is the first robot and c is the third robot, then the element p_{13} will have a value of 10, and p_{31} will have a value of 25.

In general, if a loop closure is added between key n of robot i and key m of robot$j$, the corresponding matrix elements will be updated using the following rules:

$$p_{ij} = \max(p_{ij}, n) \tag{5.2}$$

$$p_{ji} = \max(p_{ji}, m) \tag{5.3}$$

Now, when a new node n is received for robot i, LAMP should only check for loop closures with robot j if the following condition is met:

$$n - p_{ij} > d \tag{5.4}$$

This logic acts a generalised version of the distance before reclosing logic from the single-robot case. Each pair of robots is considered separately in the loop closure search, ensuring that meaningful loop closures between any given pair of robots will not be lost. It is also worth noting that in the case of single-robot loop closure, the original logic is encoded in the diagonal elements of the distance matrix **P**. Thus, this system preserves all existing behaviour for the single-robot case, while extending it to also work with multiple robots.

5.4 Results and Analysis

This section presents results obtained using the implementation of inter-robot loop closures. In all cases, LAMP was configured with d in the range 5-10 m – that is, the

robot would have to travel at least this distance between consecutive loop closures with the same pair of robots. This setting is sufficient to avoid the excess loop closure detection scenarios shown in Figure 5.2, ensuring that the computational load of the loop closure search remains within reasonable bounds. The aim of this section is to demonstrate that inter-robot loop closures are able to produce improvements in the multi-robot maps, within the constraints of the new search algorithm.

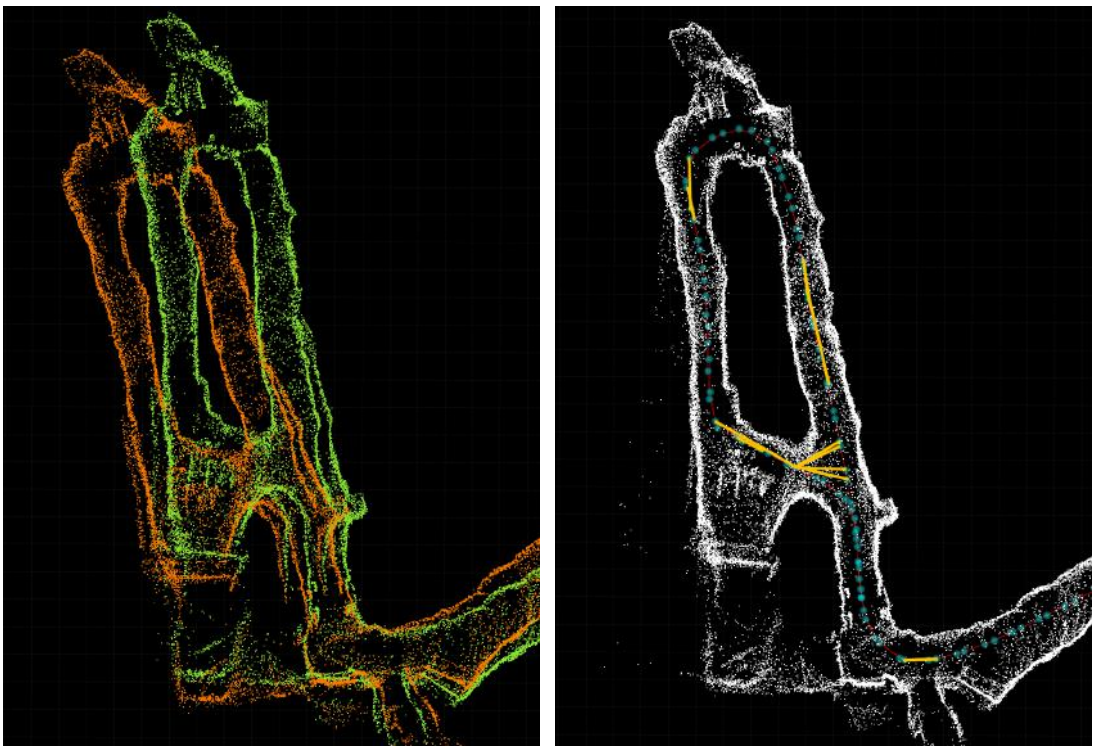
5.4.1 Miscalibration: Same Trajectory

As an initial demonstration, the loop closure mechanism was tested on a single-robot dataset played back twice, with an offset in initial conditions between the two instances. In effect, this simulates two robots traversing the same path with a mismatch in their initial calibration. The miscalibration used was an angular offset of 8°, which is larger than should typically be expected in an AprilTag-based calibration. As shown in Figure 5.3 (a), if loop closures are not used at all, this produces a map with significant inconsistencies, overlaying two maps which are skewed relative to one another.

With loop closures activated, the pose graphs from the two robots are effectively "stitched together", as shown in Figure 5.3 (b). The system is able to recover from the large angular offset and produce a consistent multi-robot map. In this case, both robots' initial calibrations were regarded as equally accurate, so the effective fused calibration is the average of the two. Had one robot obtained a calibration with a much lower covariance, greater trust would be placed in this one, allowing it to act as a correction for the other robot whose calibration was not as accurate. In either case, the overall effect of the multi-robot optimisation process is to fuse the *best available information* into a unified map estimate.

5.4.2 Miscalibration: Different Trajectories

This dataset, recorded outdoors at the NASA Jet Propulsion Laboratory, demonstrates the effect of inter-robot loop closures for two robots on *different* trajectories. Figure 5.4 shows the map obtained without loop closures, with both robots starting from a common

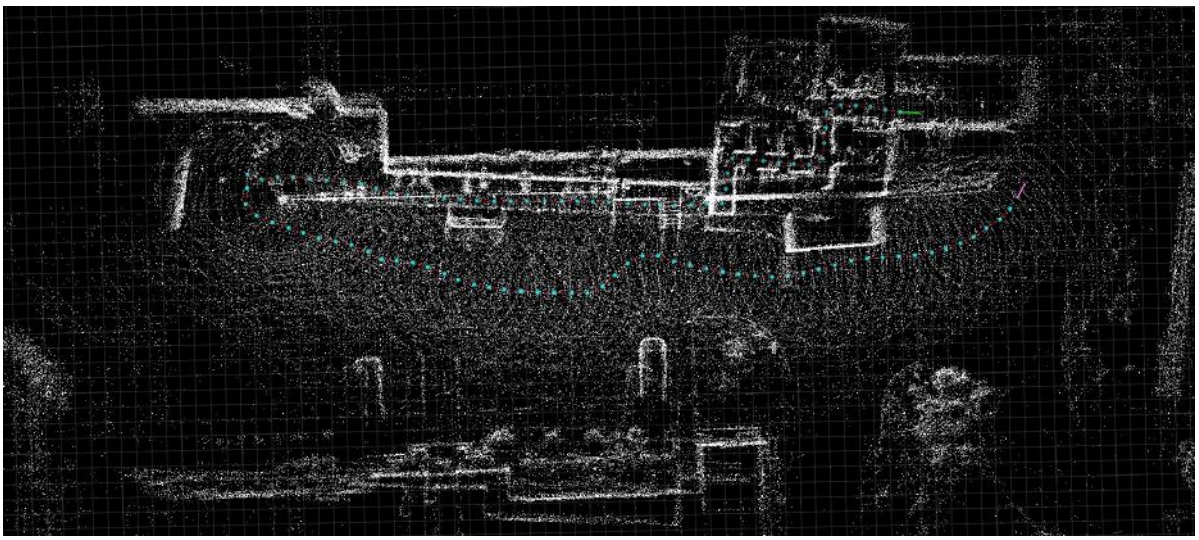


(a) (b)

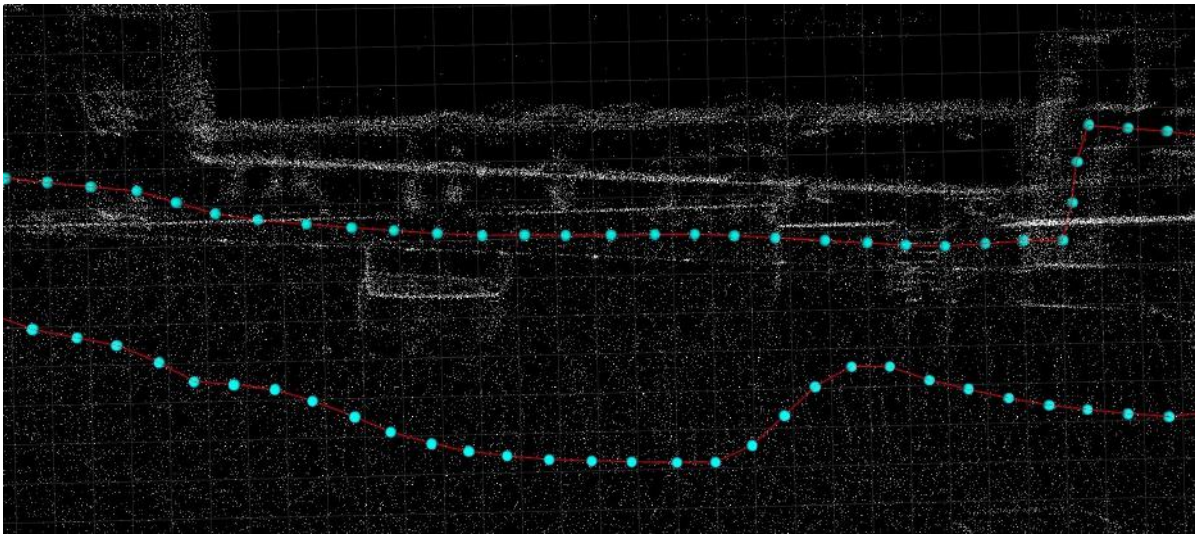
Figure 5.3. Inter-robot loop closures correcting for a calibration offset between two robots. The two robots traverse the same loop with an 8° mismatch in their initial calibration. (a) Shows the map obtained without loop closures, where each colour represents the data from a single robot. (b) Shows the unified map obtained in the same scenario with inter-robot loop closures active – loop closure edges are shown in yellow.

location on the far left of the map. Due to small inconsistencies in the fiducial calibration of the two robots, the fused map has several clear inconsistencies. This is emphasised in Figure 5.4 (b), in which one of the walls that was observed by both robots appears twice on the map. This is a common example of the types of map inconsistencies that can arise due to localisation drift of a single robot, as well as due to poor calibration of multiple robots.

Figure 5.5 shows the map obtained by LAMP on the same dataset, with inter-robot loop closures active. As highlighted in Figure 5.5 (a), only a single set of inter-robot loop closures is actually detected; after moving away from their initial location, the robots do not come close enough together again for a loop closure to be detected. However, this is



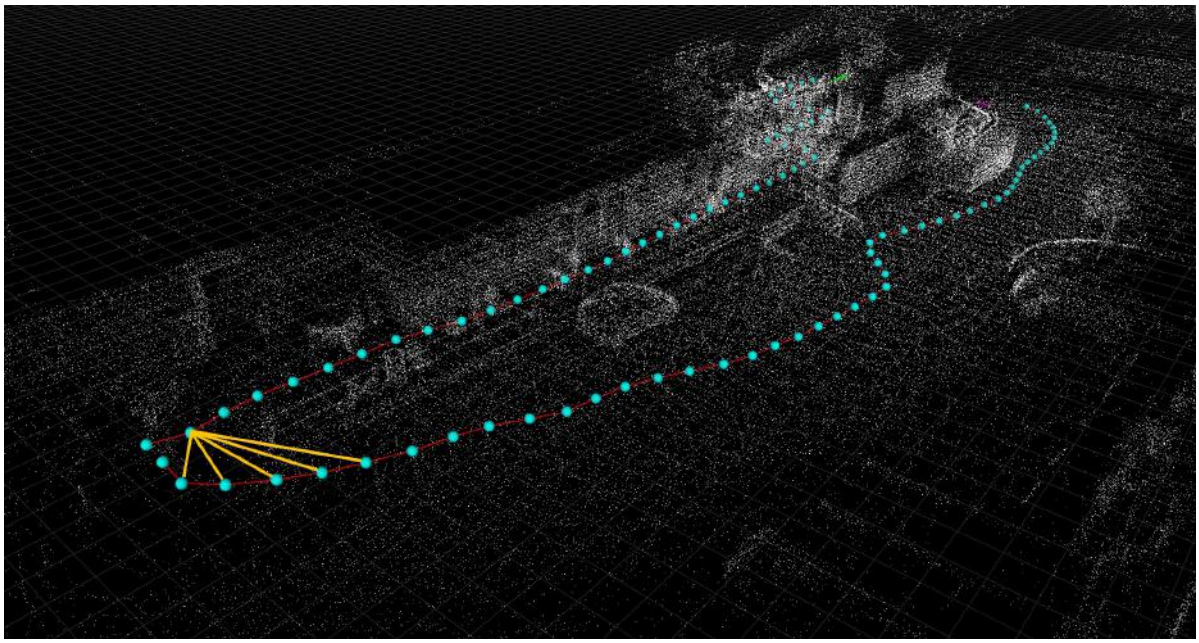
(a)



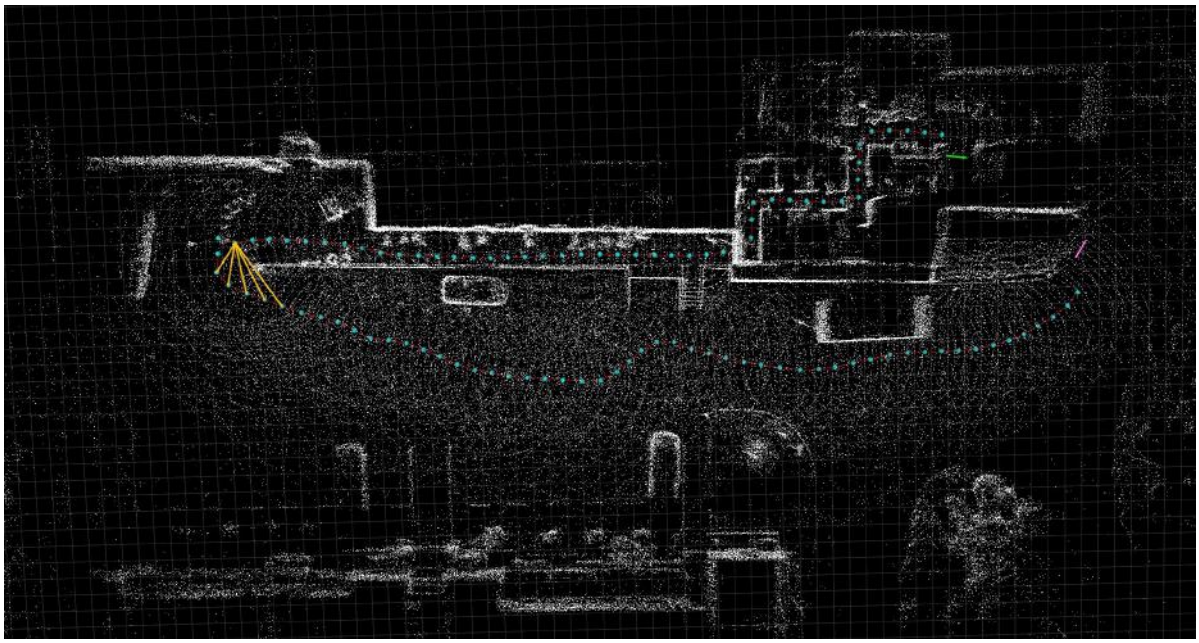
(b)

Figure 5.4. A multi-robot map without loop closures: (a) the full multi-robot map, and (b) a close-up of the inconsistencies at the centre of the map. Both robots started off at the far left of the map.

sufficient to produce a significant improvement in the quality of the map. Figure 5.5 (b) shows that the inconsistencies from Figure 5.4 are no longer visible, and the data from the two robots is fused into a single, consistent map.



(a)



(b)

Figure 5.5. A multi-robot map with loop closures: (a) an oblique view showing the loop closure near the robots' starting locations, and (b) the resulting environment map.

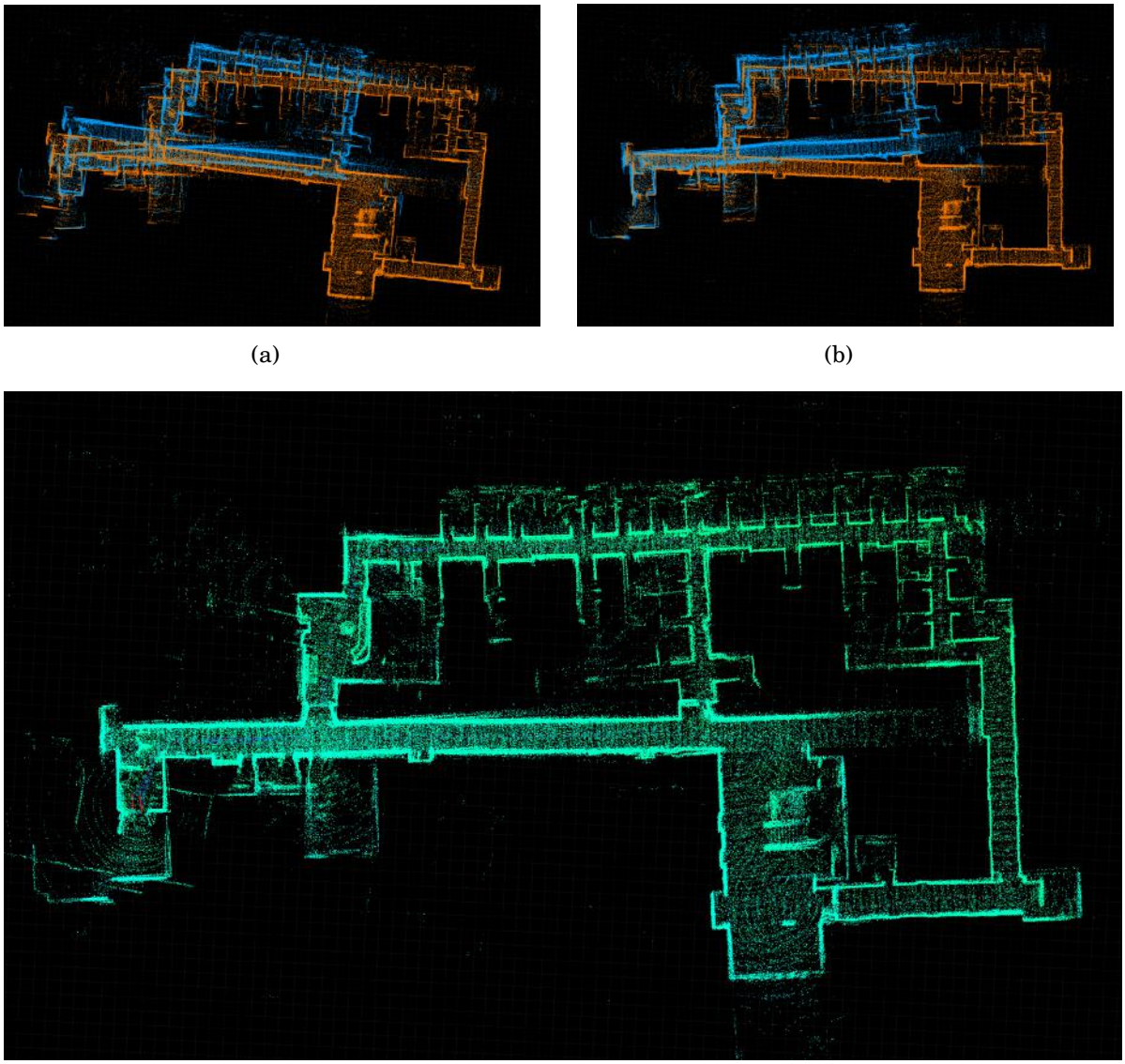
5.4.3 Large-Scale Multi-Robot Mapping

As a final demonstration of the enhanced mapping capability enabled by multi-robot loop closures, this section presents results obtained in a larger-scale environment. This environment, also seen in Chapter 4, is a basement area at the NASA Jet Propulsion Laboratory, with an exploration area spanning approximately 80×40 m. For this test, the two robots are sent to explore in different directions, on trajectories that have significant overlap but also each visit unique areas. The aim of this test is to demonstrate the importance of effective multi-robot data fusion in obtaining a unified map that is better than that obtained by any individual robot.

Figure 5.6 (a) shows the map obtained without *any* form of loop closures active. In this graphic, it is immediately apparent that the calibration between the two robots is not perfect, resulting in maps that diverge as the robots traverse farther from their starting point in the lower-left corner of the map. Each robot's map also clearly suffers from localisation drift, which produces visible inconsistencies when the robots return to their starting locations.

Figure 5.6 (b) demonstrates minor improvements achieved through *single-robot* loop closures. While this does improve the self-consistency of each robot's map, it does *not* reduce the calibration mismatch between the two robots.

Finally, Figure 5.6 (c) shows the map obtained with inter-robot loop closures active. Here, loop closures enable the data from the two robots to be fused despite the calibration offset, producing a consistent map that incorporates all of the data from both robots. Due to the different trajectories explored by the two robots, the combined map covers a larger area than that of either individual robot, representing effective incorporation of all available data into a single, unified map.



(a) (b)

(c)

Figure 5.6. A large-scale demonstration of inter-robot loop closures. Three maps are shown from the same dataset: (a) with no loop closures; (b) with single-robot loop closures only; and (c) with single- and inter-robot loop closures. In (a) and (b), each colour represents the map components from a different robot; in (c), the map incorporates fused data from both robots.

5.5 Conclusions

This chapter has presented work done in designing and implementing inter-robot loop closures in the LAMP system. This work extended from the existing logic for single-robot lidar loop closure, developing a computationally feasible implementation of centralised inter-robot loop closure search.

The results in Section 5.4 demonstrate the improvements in mapping accuracy obtained through multi-robot loop closures. The system gains an ability to recover from poor calibration between the robots and the world coordinate frame, allowing data from multiple robots to be reconciled even in the presence of significant errors. This in turn allows for effective fusions of the maps from multiple robots – with no inter-robot loop closure factors in the pose graph, maps cannot typically be fused without producing significant inconsistencies. The unified map obtained in this way is superior to that obtained by any individual robot, incorporating more information and reducing the impact of individual errors on the overall result.

This chapter highlights the enhanced exploration and mapping capabilities of *collaborative* multi-robot SLAM systems compared with single-robot or non-collaborative multi-robot ones. In addition to allowing for more ground to be covered in exploration within the same time frame, effective fusion of data from multiple robots provides demonstrable benefits in overall mapping accuracy. Inter-robot loop closures are shown to be a *necessary* component of effective multi-robot mapping, since they provide a way to combat the inconsistencies that inevitably arise between the robots, fusing their data into an accurate, unified map.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This thesis has presented a series of contributions towards the goal of developing a robust multi-robot SLAM solution for subterranean exploration. Effective exploration of subterranean environments remains an open problem in robotics, one which will likely continue to see significant research interest in the coming years due to its high potential impact.

The contributions in this thesis have been developed through three primary research areas, each of which is the subject of a chapter in the thesis. This chapter summarises the research work presented for each of these primary research areas, including discussion of potential avenues for future development, and concludes with a few closing remarks on the thesis as a whole.

6.1 System Architecture Design

The system architecture developed in this thesis addressed all of the design goals laid out at the beginning of Chapter 3, providing a modular and extensible platform for further research into multi-robot SLAM in challenging subterranean environments. The pre-existing LAMP system was redesigned from the ground up, guided by a clear set of

design goals and principles, resulting in an improved system that is far more intuitive to work with. This also provided the foundation for the rest of the work in the thesis, which is implemented within the redesigned LAMP architecture.

Future Work

Although the system design work in this thesis successfully addressed its major design goals, there are still several areas with opportunities for further improvement to the system as a whole.

The modularity of the system could be increased further by separating the internal logic and stored data from the outward-facing ROS interfaces. In the existing LAMP system, ROS modules are tightly coupled with the internal logic of the system, resulting in added complexity and many inefficiencies through unnecessary data conversions. A more efficient system would limit the use of ROS data types to interfaces with other ROS nodes – internally, each node would have no direct dependencies on ROS data types, instead using the best-suited formats for their particular purposes.

Evaluation of computational efficiency under the new architecture has thus far been fairly rudimentary, consisting mostly of manually confirming ROS publisher frequencies and CPU load. While this does confirm that the system is producing outputs without accumulating delays due to computational overload, it does not provide a holistic view of the performance of the system. Quantitative analysis of the entire system, such as by running system profiling software during operation, would allow for identification of less obvious bottlenecks and inefficiencies in the system.

6.2 Performance Evaluation

The ability to meaningfully evaluate the performance of SLAM systems operating in challenging subterranean environments is essential for effective continued development and improvement. However, it is often not possible within the practical constraints of subterranean environments to obtain the desired ground truth information. The

techniques presented in this thesis allow for analysis to be performed on an approximated ground truth trajectory, obtained by incorporating sparse ground truth measurements into the pose graph. The results presented in Chapter 4 demonstrate that, while there are some subtleties to making the technique viable in scenarios of non-uniform error, it is a viable mode of analysis that enables richer performance evaluation in subterranean environments than would typically be possible.

Future Work

The performance evaluation method as presented in this thesis is largely a proof of concept. While these initial results are promising, the technique would benefit from further analysis in order to confirm that the information it provides is reasonably accurate.

One method for rigorously testing the system would be to deploy a robot in an environment where ground truth data is available for a relatively large number of landmarks. The accuracy of the technique can then be evaluated by repeated application using different subsets of the available ground truth data, through a subsampling technique such as k-fold cross-validation [100]. This would provide a quantitative estimate of the accuracy of the technique, validating its use for further analysis.

The technique was also found to produce poor results in cases of highly non-uniform localisation error, such as the lidar slip scenario explored in Section 4.5.3. This failure in a sense represents reasonable behaviour of the technique in cases where insufficient ground truth data is available to constrain more complex errors in the map. However, further testing in more challenging environments would likely provide additional insights into when exactly these failures are likely to occur, and how much additional ground truth data is required to offset their effect.

6.3 Multi-Robot Data Fusion

In a multi-robot SLAM system, the quality of the map that can be produced depends not only on the quality of each individual robot's data, but on the ability to effectively fuse data from multiple robots in an accurate and consistent manner. Chapter 5 presented contributions to this end, developing and testing an implementation of lidar-based inter-robot loop closures built on top of the LAMP system. The results presented in this chapter demonstrated that effective inter-robot loop closures provide a mechanism through which the system can recover from calibration errors and inconsistencies, producing a multi-robot map that seamlessly fuses data from each robot.

Future Work

Despite the promise of the results presented for multi-robot loop closures in this thesis, the system would benefit from further testing in even larger-scale and more challenging environments. Such tests could provide greater insight into the ability of inter-robot loop closures to enable recovery from worse localisation drift, or to reconcile data from multiple robots with more significant inconsistencies.

The tests in this thesis were also limited to only two robots running concurrently, due to availability of robots at the time of writing. However, all algorithms and implementations were designed to apply to generic multi-robot systems – further tests with four or five robots operating concurrently could provide deeper insight into the strengths and weaknesses of the loop closure framework. The algorithm could be modified to dynamically adjust parameters such as the loop closure search radius based on computational load, thereby making more effective use of the computational resources available in any given scenario.

6.4 Closing Remarks

Subterranean environments are some of the most challenging domains for robotic exploration, and also present many of the most exciting and important applications for robotic technologies, ranging from planetary exploration to terrestrial disaster response.

This thesis contributes on several fronts to the development of a multi-robot SLAM system that is purpose-built for the diverse challenges of subterranean exploration. The system architecture is redesigned and re-implemented from the ground up, adopting a modular framework that naturally allows for the many sensing modalities and system components that are required to tackle the unique challenges of subterranean environments. A new technique is proposed for system performance evaluation, providing rich and visually intuitive trajectory-based analysis which is typically unavailable in non-laboratory environments. Finally, a framework for inter-robot lidar loop closures is implemented within LAMP, which effectively leverages the information provided by multiple robots to construct a unified and improved map estimate.

The work presented in this thesis represents a small but important step towards accurate and robust multi-robot SLAM for subterranean exploration.

APPENDIX A

CASE STUDY: MECH4601 PROFESSIONAL ENGINEERING 2

This chapter presents a case study of the subject MECH4601 Professional Engineering 2, exploring in depth the content and learning outcomes of the subject in relation to my industrial placement at JPL. In particular, we demonstrate how this work has addressed all of the key outcomes and attributes developed in the subject.

Courses at the University of Sydney are designed based on *attributes*, or *key course goals*, which broadly define the purpose of the course. Each attribute has associated *learning outcomes*, which are more specific descriptors for skills that students should develop throughout the course.

The chapter first provides an overview of the content covered in the unit of study, contextualised broadly to the work done at JPL. We then consider each course attribute in turn, exploring in detail how the relevant subject matter and learning outcomes were addressed by my research at JPL. All unit of study information is from the University of Sydney Course and Unit of Study Portal (CUSP) [101].

A.1 Unit Overview

Professional Engineering 2 aims to develop students' awareness of the issues and challenges of project management in an engineering workplace, as well as fostering improved oral and written communication skills. Accreditation of tertiary studies across all engineering disciplines in Australia is handled by Engineers Australia [102]; this course also aims to partially address the requirements of Engineers Australia for training in management theory and professional engineering skills [101]. On completion of the course, students should be better equipped to conduct themselves appropriately in professional engineering contexts. This includes a deeper understanding of management structures and professional conduct; knowledge of when and how to effectively consult with experts; and improved communication and presentation skills.

A.2 Attributes and Learning Outcomes

This section considers each of the attributes of MECH4601, addressing all learning outcomes encapsulated within each attribute.

A.2.1 Professional Effectiveness and Ethical Conduct

Awareness of ethical and other issues which can arise in the workplace.

NASA JPL is committed to upholding ethical conduct in its operations as a business, which is reflected in the behaviour expected of its employees [103]. At the start of their employment at JPL, each employee completes mandatory online training which covers the key expectations of employees as described in the JPL Ethics Handbook [104].

The ethical practices and principles outlined in the handbook include:

- **Integrity**: employees are held accountable for their actions, and are encouraged to hold others accountable for theirs. This includes speaking up if another employee is engaging in unsafe or unethical behaviour.

- **Conflicts of Interest**: as an organisation that relies on public funding, it is imperative for JPL to avoid not only conflicts of interest, but also *perceptions* of conflicts. Principles emphasised to employees include not using their JPL position for improper personal gain, avoiding activities that could give the appearance of adversely affecting their objectivity, and removing themselves from JPL processes for which the employee might have external financial interests.
- **Openness**: JPL values openness of its people and processes, in the belief that this fosters a better work environment for everyone. Employees are encouraged to voice doubts or concerns to their managers, or where appropriate to either the JPL Ethics Office or Human Resources.
- **Fostering a Supportive and Diverse Community**: JPL has a culturally diverse community, and ensures that all employees are treated with dignity and respect. As a research organisation, its standing is strengthened by the value placed on the different ideas and perspectives that each employee can contribute.
- **Responsible Conduct**: JPL is an environmentally conscious organisation, and encourages employees to take on a similar responsibility. Sustainable practices such as recycling, reuse of equipment, and offsite recycling are just a few of the ways in which this principle is enacted.
- **Maintaining a Safe Work Environment**: JPL recognises the importance of a safe work environment to the physical, mental and emotional well-being of its employees and the wider community. This is discussed in detail in the context of Workplace Health and Safety in Appendix C.

Understanding of what is required in the conduct and management of an engineering project.

At JPL, I was part of the CoSTAR team for the DARPA Subterranean Challenge, a collaboration between NASA JPL, MIT, Caltech, KAIST and LTU. Management of a team with over 50 members spread across multiple locations and time zones is a challenging

task, and the team had many structures in place to make day-to-day operations as smooth as possible.

An important part of the team's management was their regular meetings, which typically included the following:

- **Weekly All-Hands**: each week, the full CoSTAR team attended an hour-long meeting in which updates were shared by management, as well as the sub-team leads. This ensured that all team members understood the big picture of how the team's work was progressing, and how their own work contributed to the overall team objectives.
- **Leads' Meetings & Stand-Ups**: the sub-team leads would also meet regularly for more in-depth discussions of progress, and to make decisions on the path forward. This included a weekly hour-long meeting, as well as daily stand-ups in which each lead provided a brief progress update from their sub-team.
- **Sub-Team Meetings & Stand-Ups**: each sub-team also conducted regular meetings in which every team member could share their progress and discuss concerns with the people working on similar tasks. This ensured that each team member was clear about their priorities, and could ask for help on any difficulties that were holding them back. The sub-team lead could also update the team on any shifts in priorities that arose from discussions in the leads' meetings.

Team members could also arrange additional group or one-on-one meetings to supplement the regular meetings as required. The team also made use of a variety of software tools to assist in management of different aspects of the project:

- **Slack**: an instant messaging platform used for daily communication across the whole team. Slack is used JPL-wide, and CoSTAR made use of separate channels for the full team, as well as for different sub-teams and project areas. It could also be used for direct messaging between team members.

- **Gitlab**: a code management tool used for all of CoSTAR's code. The team relied on Gitlab for version control through Git repositories, as well as issue tracking and CI/CD (continuous integration & continuous deployment).
- **Trello**: a versatile project management tool, used by some CoSTAR sub-teams to keep track of progress on active issues and priorities for future work.
- **Skype & WebEx**: video conferencing software is extremely important to CoSTAR, since many team members work from locations outside of JPL. Skype and WebEx were the team's primary tools to allow participants to join meetings remotely.
- **Google Docs**: web-based word-processing software, used by the team for online planning, note-taking and real-time collaborative editing.

Each of these tools played an essential role in the management of my team at JPL, allowing daily operations to proceed smoothly despite the size and complexity of the project.

Ability to recognise the range of expertise you may need to call on in your role as an engineer working on a project.

NASA JPL is a research centre which employs over 6,000 people with expertise in a diverse array of fields, ranging from aerospace and robotics to environmental sciences and astrophysics. This diversity is extremely important to JPL's success as a research institution, as it allows for rapid innovation through interdisciplinary collaboration and consultation with world-leading experts in particular fields.

The group structure at JPL encourages this type of collaboration – it is common for employees to divide their time between multiple projects, rather than working exclusively on a single project at a time. This is extremely beneficial for all JPL teams, as they can more easily bring on expert engineers or scientists as consultants for particular projects. These people may devote only 10-20% of their time to the project, providing valuable input without contributing directly to day-to-day engineering work.

An example of this arose when CoSTAR was having issues with their traversability analysis, which aims to determine which areas in a robot's field of view are safe for it to traverse. Robots were encountering difficulties with obstacles such as rocks that were small enough to be difficult to detect, but large enough to cause problems if a robot were to drive over them. The team found another JPL employee who had worked on traversability analysis for an autonomous rover which would explore the surface of the Moon. This employee was able to lend his time and expertise to CoSTAR for a few weeks, allowing the team to resolve the issue much more quickly and effectively.

Consulting with experts was also important in my own daily work. In working on a complex robotic system developed over the course of a year by dozens of experienced engineers, I inevitably encountered problems that I could not resolve easily on my own. From listening to many team members' presentations at regular meetings, I developed a good sense for who was responsible for each part of the system. Understanding the expertise of other team members then made it easy for me to know whom to ask for advice whenever new issues arose. Talking to experts in different fields also allowed me to develop a much broader understanding of robotics as a whole, rather than limiting my learning to the specific sub-areas that I was working on directly.

A.2.2 Project and Team Skills

Ability to work effectively in a small team to produce a technical report.

During my placement, several coworkers and I wrote a paper for submission to the International Conference on Robotics and Automation (ICRA), one of the foremost conferences in the field of robotics. The paper presented several aspects of the work done by the sub-team in the first year of the DARPA Subterranean Challenge. My work on this paper was primarily in two areas: firstly, designing and running experiments with our software to collect results for the paper; and secondly, collaboration on writing and editing of the text to ensure that it was of suitable quality.

Contributing to this paper was a unique opportunity to learn how technical papers are written at the highest level in the field of robotics. Throughout the weeks of writing the paper, I constantly asked for feedback on my contributions and received extremely valuable guidance from my more experienced co-authors. Figure A.1 shows two figures I contributed to the paper, which went through many iterations of refinement before reaching the final form shown in the figure. In this iterative process, I learned how to better present information visually in a robotics paper, as well as how to select aspects of the data that are most appropriate to present. Particularly valuable feedback during this process came from one of CoSTAR's collaborators, an Assistant Professor at MIT with extensive experience in writing papers for robotics conferences. The opportunity to receive rapid and detailed feedback early on in my work from an expert who has contributed to dozens of well-recognised papers in the robotics literature allowed me to develop my own skills even further.

Ability to plan small projects, and contribute effectively to planning of larger projects.

Working on the DARPA Subterranean Challenge at JPL regularly involved planning small projects with coworkers. The workflow of each sub-team typically revolved around cycles of formulating plans at weekly meetings, progressing or completing the work over the following week, and then converging to update priorities at the next meeting. This is similar to the concept of a *sprint* in agile software development. This workflow gave me regular exposure to the team planning processes, allowing me to voice my own opinions on the directions that the team priorities should take over the upcoming weeks. Projects would often take longer than anticipated, or bring to light new issues that significantly increased the scope of the project, thus requiring re-planning or redistribution of work in order to maintain a productive workflow.

I also contributed to planning of several larger projects, which came with further challenges of parallelisation and prioritisation of work. One such project was a major code refactor of CoSTAR's Large-Scale Autonomous Mapping and Positioning (LAMP) software, the core SLAM system developed by CoSTAR. Over the year since its develop-

LAMP: Large-Scale Autonomous Mapping and Positioning for Exploration of Perceptually-Degraded Subterranean Environments

Kamak Ebadi[1], Yun Chang[2], Matteo Palieri[1], Alex Stephens[1], Alex Hatteland[1], Eric Heiden[3], Abhishek Thakur[4], Benjamin Morrell[1], Luca Carlone[2], Ali-akbar Agha-mohammadi[1]

(a)

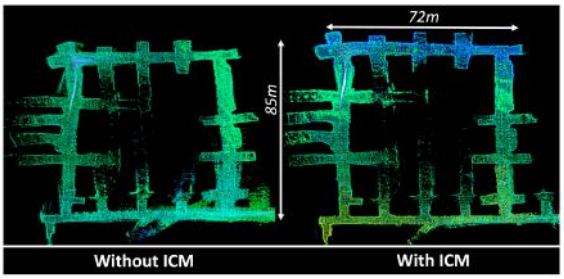


Fig. 7: A section of Bruceton Safety Research mine. Left: PGO without ICM; the map is visibly distorted. Right: PGO with ICM; map has no distortion since outlier loop closures are rejected.

(b)

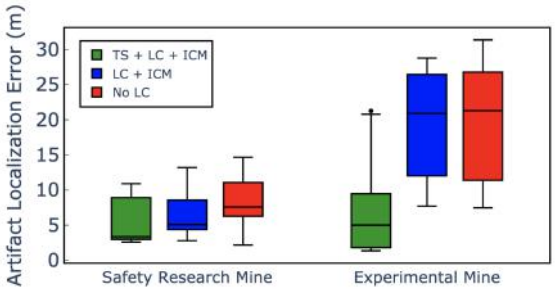


Fig. 8: Artifact localization accuracy using LAMP, with and without loop closures (LC) and ICM, and with total station (TS) measurements in the Bruceton Safety Research and Experimental mines during the Tunnel Circuit of the DARPA Subterranean Challenge.

(c)

Figure A.1. Excerpts from the paper submitted to the International Conference on Robotics and Automation (ICRA) 2020: (a) the title and author list, and (b-c) figures contributed.

ment began, LAMP had developed into a complex system consisting of dozens of modules and many thousands of lines of code – redesigning and refactoring it was a collaborative effort between nine people, taking several weeks of work.

As a larger project, this required more structured planning than most of the other projects I worked on at JPL. Since I had significant involvement in designing the new software architecture (the subject of Chapter 3), I was also partly responsible for deciding on the prioritisation of tasks. Figure A.2 shows an online spreadsheet that my team used for both planning and updates on progress. We created this spreadsheet by first listing all software modules that needed to be written, and allocating each to the most suited person (or in some cases, multiple people). We then identified which modules it made sense to prioritise, by considering dependencies of each module as well as requirements for a minimum working version of the core software. Based on this, we constructed a

				Timeline			
	Class Name	Owner	Status (%)	Templates	Minimal Capability	Tunnel Capability	Enhanced Capability
Data Handler	LampBase	Alex / [redacted]	100				
	LampRobot	Alex / [redacted]	80				
	LampBaseStation	Alex	10				
	PoseGraphMerger	Alex / [redacted]	90				
	LampDataHandlerBase	[redacted]	40				
	OdometryHandler	[redacted] / [redacted]	85				
	ArtifactHandler	[redacted] / [redacted]	90				
	AprilHandler	[redacted] / [redacted]	60				
	IMUHandler	[redacted]	0				
	TotalStationHandler	[redacted]	0				
	ManualLoopClosureHandler	Alex	30				
	UWBHandler	[redacted]	0				
	LoopClosureHandler	[redacted]	70				
Loop Closure	LoopClosureBase	[redacted]	80				
	VisualLoopClosureNode	[redacted]	20				
	CostMapLoopClosureNode	[redacted] / [redacted]	0				
	LaserLoopClosureNode	[redacted]	80				
	PoseGraphVisualizer	[redacted]	80				
	PoseGraphSubscriberHandler	[redacted]	20				
	LampPgo	[redacted]	85				
	UWB Node	[redacted]	10				

Figure A.2. Planning task distribution and prioritisation for the LAMP code redesign.

timeline of different capability stages, giving everyone a clear sense of their priorities going forward. The spreadsheet was supplemented by additional tools that the team regularly uses, including Gitlab for issue tracking, Slack for instant communication, and Trello for bigger-picture organisation. This entire process was an enlightening experience of the considerations involved in planning a complex and fast-moving project with many collaborators, providing me with valuable experience in project management and leadership.

A.2.3 Communication and Inquiry / Research

Ability to prepare an interesting presentation on aspects of your work for your peers or senior managers.

Presenting work to peers and managers is an essential skill in any engineering workplace, and JPL is no exception to this. Over the course of my time at JPL, I presented my work in a variety of different contexts.

Each week, the sub-teams of CoSTAR have meetings to share updates on progress and discuss issues with people working on similar tasks. At the meetings for my sub-team – whose focus is on robotic perception, localisation and mapping – each team member was asked to spend a few minutes updating the rest of the team on their progress. This could be done purely verbally, or accompanied by slides or videos where appropriate. Presenting regularly in this way allowed me to develop my ability to quickly and effectively communicate key information to a small group of experts in the field, including the team lead who oversees the work of each sub-team member. These short update presentations would often lead to questions or in-depth discussion, which provided further opportunities to develop my confidence in more ad hoc communication of complex ideas.

Additionally, as part of the requirements of the ESIPS program I prepared two seminars to present at JPL, the first taking place around four months into my internship and the second at the end of the full six months. These seminars were presented on-site to robotics engineers with a variety of different specialisations, as well as via remote link to students and academics at the University of Sydney. The diversity of this audience presented a challenge in developing the presentation, which had to be at an appropriate level to be interesting and informative to all attendees. I approached this by incorporating feedback from my team lead – an expert in the subject matter – as well as other CoSTAR members who were not directly familiar with my work. By doing this, I was able to deliver a presentation on my work that was engaging and informative to attendees of all backgrounds.

Ability to write a concise, technical engineering report.

The paper written for ICRA – discussed in Section A.2.2 – was a valuable experience of writing a concise, technical engineering report.

In addition to the paper for ICRA, however, this thesis itself falls within the scope of this learning outcome. While writing the literature review, I gained valuable experience with the process of researching a topic in-depth and synthesising large volumes of infor-

mation. The body of the thesis required describing software design and methodologies, as well as presenting results and analysis in a clear and concise manner. As I wrote the thesis, I regularly discussed the structure and presentation of information with my industry supervisor. I also submitted drafts to both my industry and university supervisors, whose expert feedback allowed me to further improve the thesis.

A.3 Conclusion

Working at NASA JPL gave me not only a wealth of new technical knowledge, but also the valuable experience of working in a professional engineering environment. I contributed to a variety of collaborative projects including implementation of new robotic perception features, writing a paper for submission to ICRA, and a major redesign of one of CoSTAR's largest and most complex software packages.

Applying myself to all of these projects, as well as this thesis itself, broadened my understanding of the challenges of project management in a professional engineering setting. It also allowed me to develop my own skills in communication and time management, thereby comprehensively addressing all learning outcomes of MECH4601 Professional Engineering 2.

APPENDIX B

CASE STUDY: MTRX5700 EXPERIMENTAL ROBOTICS

This chapter presents a case study of the subject MTRX5700 Experimental Robotics, exploring in depth the content and learning outcomes of the subject in relation to my work at JPL. In particular, we demonstrate how this work has addressed all of the key outcomes and attributes developed in the subject.

Similarly to the previous chapter, we first provide an overview of the unit of study as a whole, followed by a breakdown of the learning attributes in relation to the work done at JPL. However, many of the learning outcomes of Experimental Robotics are closely related to one another, so rather than addressing each outcome individually, we address several of them in the form of a case study on robotic system design in the following section. This case study explores the interaction of the many components in a robotic system for autonomous subterranean exploration, justifying design choices in terms of the requirements of the DARPA Subterranean Challenge. All unit of study information is from the University of Sydney Course and Unit of Study Portal (CUSP) [105].

B.1 Unit Overview

Experimental Robotics aims to equip students with a broad understanding of the full stack of technologies associated with typical industrial and mobile robotic systems. Students are introduced to the major sub-fields of robotics, including sensing, kinematics, mapping, navigation and control. The primary aim of the course is to equip students with the skills necessary to design and develop robotic systems for practical applications. Throughout the course, students develop familiarity with common sensor technologies, conventions in robot kinematics, and essential algorithms in mapping, navigation and control. The course also covers robotic system design processes, contextualised to applications such as manufacturing, automotive systems and field robotics. The course touches on a breadth of additional areas in robotics including multi-robot systems, perception, system architectures, obstacle avoidance, path planning, robotic learning and computer vision.

B.2 Attributes and Learning Outcomes

This section considers each of the attributes of MTRX5700. Some of the learning outcomes are addressed directly in this section; others are addressed in the case study on robotic system design in the following section.

B.2.1 Communication and Inquiry / Research

This attribute addresses fluency and effectiveness in communication of engineering research ideas. The learning outcomes encapsulated by this attribute are addressed in detail in Section A.2.3 in the case study for Professional Engineering 2.

B.2.2 Design

Apply a systematic approach to the design process for robotic systems. Students will gain an understanding of the components that make up a robotic system and will have the opportunity to exercise these skills on a major project of their choosing.

This learning outcome will be addressed in Section B.3.

B.2.3 Engineering / IT Specialisation

Examine advanced topics in robotics including obstacle avoidance, path planning, robot architectures, multi-robot systems and learning as applied to robotic systems.

This learning outcome will be addressed in Section B.3.

Be familiar with sensor technologies relevant to robotic systems. Specifically work with laser and vision data and examine techniques for processing this data.

This learning outcome will be addressed in Section B.3.

Have implemented navigation, sensing and control algorithms on a practical robotic system. Examine methods for fusing multiple data sources to improve a navigation solution. Using the navigation solution, students will also examine mapping techniques used in mobile robotic systems.

This learning outcome will be addressed in Section B.3.

Understand conventions used in robot kinematics and dynamics. In particular, methods for assigning frames of reference to robotic systems and techniques for transforming between frames will be described.

Reference frames are an important concept in modern robotics, used to clearly define the spatial relationships between different system components. For example, if a robot observes a landmark through its camera, that observation will give the position (and

perhaps also the orientation) of the landmark *with respect to the camera*. In order to determine the position of the landmark in a global fixed reference frame, it must be possible to construct a transformation between the sensor frame and this global frame.

CoSTAR uses the tf package from ROS to manage reference frames. This package provides a standardised method through which all reference frames within the system can be managed. Coordinate system information is stored in a graph called the *tf tree*, with reference frames represented as nodes and transforms between them represented as edges. The transform between two frames can then be constructed by composing the transforms on the edges connecting the corresponding nodes [106].

Transforms between coordinate frames are often time-dependent, and can be published by multiple sources. The tf library allows processes to query for a transform at a specific time, and computes the result by interpolating between the nearest transforms. In order for this process to return accurate results, the transforms must be published at a sufficiently high frequency relative to the rate of change of the transform itself.

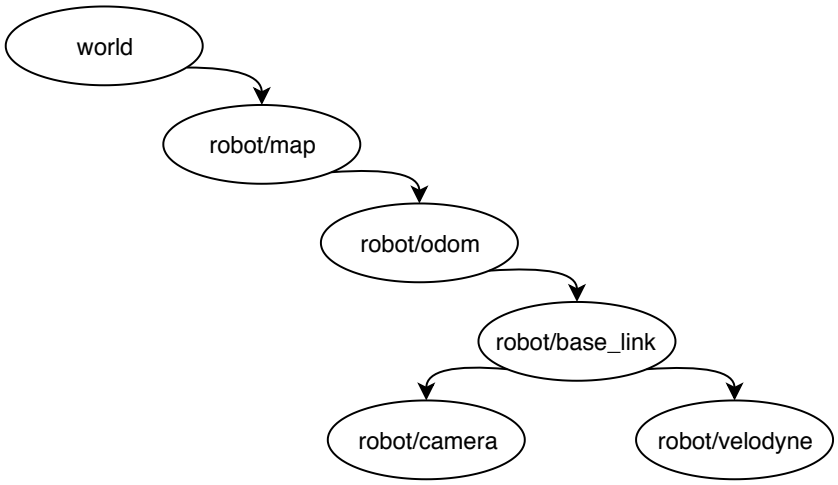


Figure B.1. A simplified representation of the tf tree used in CoSTAR systems.

The reference frames used in CoSTAR systems, as depicted in the simplified tf transformation tree in Figure B.1, are as follows:

- `robot/sensor`: each sensor on-board the robot has its own reference frame in which it makes observations (e.g. `robot/camera`, `robot/velodyne`).

- `robot/base_link`: this frame defines a central reference frame affixed to the body of the robot. Transforms between the base_link frame and sensor frames are constant, calculated based on a CAD model which describes where the sensors are mounted on the robot.

- `robot/odom`: this frame measures the position of the robot over time with respect to its *initial* position, based on incremental updates from odometry measurements. This frame is used for local trajectory planning and collision avoidance.

- `robot/map`: this frame defines the localisation estimate of the robot with respect to its initial position, as computed through lidar SLAM. The difference between the map and odom frames is that the latter does not incorporate loop closures, which produce discontinuities in the transforms. Such discontinuities present difficulties for path planning and collision avoidance, which therefore operate in the odom frame rather than the map frame. The map frame can therefore be thought of as defining corrections to the odom frame based on loop closures.

- `world`: the DARPA-defined global reference frame, with respect to which all artifact locations must be reported. The origin and orientation of this frame are defined by a fiducial gate placed at the entrance to competition areas. The transform between each robot's map frame and the global reference frame is defined by calibration with the fiducial gate at the start of a competition run.

B.2.4 Problem Solving and Inventiveness

Develop the capacity to think creatively and independently about new design problems.

Much of the work described throughout my thesis required the capacity to analyse problems and creatively design solutions. Chapter 3 in particular describes a system design that was settled upon after many discussions and iterations. This final software architecture made use of similarities between seemingly unrelated processes, such as lidar odometry and artifact detection, to significantly reduce code volume and complexity.

Another area in which I took a creative approach to an engineering design problem was in my contribution to the ICRA paper written by my team (discussed in Section A.2.2). After the Tunnel Circuit, the first competition stage in the DARPA Subterranean Challenge, the team wanted to quantitatively evaluate the performance of each robot's localisation and mapping during the competition run. However, there were no ground truth sensors present in the tunnels. This meant that the only obvious metric we could use was localisation error of the artifacts, for which DARPA did provide ground truth locations.

However, my feeling was that it would be helpful to have approximate ground truth information over the full trajectory, not just at locations where artifacts were detected. With this goal in mind, I added a new feature to the existing SLAM system that allowed the operator to manually input ground truth locations of artifacts, which would then be added as prior factors to the pose graph. The result is that the map would "anchor" to the known artifact locations, allowing us to compare the adjusted trajectory with the original one to approximate localisation drift throughout the trajectory. This work is the subject of Chapter 4.

Undertake independent research and analysis and to think creatively about engineering problems.

Throughout my time at JPL, I frequently conducted independent research and analysis, most significantly for this thesis itself. In order to deeply understand the problems I wanted to tackle, it was necessary to learn about similar existing systems, as well as relevant concepts that I might not have initially been familiar with.

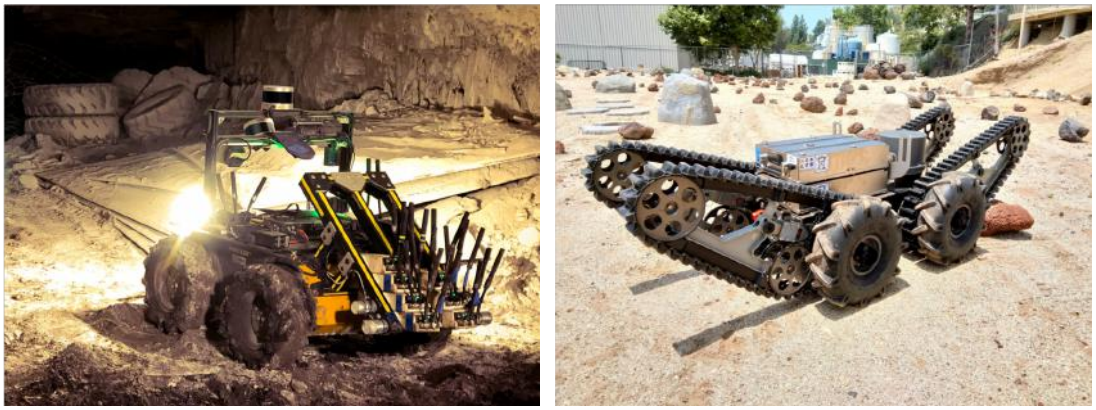
After the upgraded design for the SLAM architecture used by CoSTAR was decided upon, I led the implementation of the software for the base station. This required independent analysis of the computational load of different processes, in order to ensure that there would be no bottlenecks in the main ROS node. The results of this analysis informed my path toward implementing the software, resulting in a final product that ran smoothly and efficiently.

B.3 Robotic System Design

My work at JPL has been primarily on perception and mapping, as part of the CoSTAR team in the DARPA Subterranean Challenge. This is only one of many subsystems that make up the full robotic system used by the team. Over the course of my placement, I developed a broad understanding of how the many components of the system interact to achieve autonomous exploration, navigation and mapping in subterranean areas. This section demonstrates this understanding by presenting an overview of the full system architecture used by CoSTAR.

B.3.1 Hardware and Sensors

The system is built primarily on two ground robot platforms, the Husky A200 and the Telemax Pro, both shown in Figure B.2. The Husky is a versatile platform that is able to traverse some level of rugged terrain, while also being comparatively user-friendly and safe to operate. The Telemax is a much more powerful robot which can handle more difficult terrains, capable of climbing over rocks and ascending stairs, but is also much more expensive and dangerous to operate. The Husky is therefore used as the primary platform, since in most scenarios it is capable of the required actions.



(a) (b)

Figure B.2. The two primary ground vehicles used by CoSTAR for the DARPA Subterranean Challenge: (a) the Husky A200, and (b) the Telemax Pro.

Mounted on each robot are the four different types of sensors shown in Figure B.3, each chosen to address particular system requirements.

(a) (b) (c) (d)

Figure B.3. The sensors used on-board CoSTAR ground robots: (a) the Velodyne VLP-16 3D lidar, (b) the Intel RealSense D435 depth camera, (c) the Boson LWIR thermal camera, and (d) the VectorNav VN-100 IMU.

3D Lidar

In order to navigate a subterranean environment, the robot must be able to detect walls and obstacles around it. This can be achieved using a variety of different sensors, as discussed in Section 2.1. 3D lidar is chosen due to its suitability for perception and navigation in unevenly illuminated and feature-poor subterranean environments. The sensor used is the Velodyne VLP-16 (Figure B.3 (a)).

Camera

In the DARPA Subterranean Challenge, the robots must be able to identify artifacts within the environment, such as a backpack or a fire extinguisher. The most straightforward way to do this is through visual recognition, so multiple cameras are mounted on the robot facing to the front and sides. These cameras can also be used for visual odometry, which is particularly helpful in self-similar corridors in which lidar odometry may fail. The visual sensor used is the Intel RealSense depth camera D435 (Figure B.3 (b)), which can provide RGB colour information as well as depth information at each pixel

(known as RGB-D). This depth information improves the accuracy with which an artifact can be localised relative to the robot.

Thermal Camera

One of the artifacts specified by DARPA is a human "survivor" mannequin, which has a heat signature akin to that of a real human. The thermal sensor used is the Boson Longwave Infrared (LWIR) thermal camera (Figure B.3 (c)).

Inertial Measurement Unit (IMU)

Robotic navigation and control require accurate estimation of the state of the robot at all times. While this can be achieved purely through visual or lidar sensors, an IMU can improve accuracy significantly at low computational cost. The sensor used is the VectorNav VN-100 (Figure B.3 (d)). The IMU can also be paired with the lidar, RGB-D camera or thermal camera to achieve lidar, visual or thermal inertial odometry respectively.

B.3.2 Software Framework

Software on CoSTAR robots is built on the Robot Operating System (ROS). ROS is an open-source robotics middleware platform that provides hardware abstraction, device drivers, software libraries and visualisation tools [107]. ROS is the software platform of choice for many robotics applications, and thus has libraries and application programming interfaces (APIs) for many popular sensors, actuators, robotic platforms and software tools. CoSTAR makes use of existing ROS tools and interfaces for a variety of system components, including:

- ROS APIs to control the Husky A200 robot.
- ROS drivers for sensors, including the VLP-16 lidar and the Intel RealSense RGB-D camera.

- A ROS wrapper for the Point Cloud Library (PCL) [25], a library for handling of point clouds which provides functionality such as segmentation, feature extraction, filtering and registration.
- rviz, a visualisation tool provided with ROS which can natively visualise point clouds, pose graphs, robot models, cost maps, and many other useful data types.

Another powerful feature of ROS is its abstraction of software processes in the form of ROS *nodes*. A typical robotic system running on ROS will have multiple nodes, each encapsulating a particular process or interface – for example, there might be a node to read the raw input from the VLP-16 lidar unit and convert it to a form that can be used by other processes. These nodes interact by publishing and subscribing to *messages*, data packets that are transmitted between nodes. These publications and subscriptions are tied to specific *topics*, communication buses which anonymously pass around messages of fixed types. This system of nodes, topics and messages provides user-friendly abstractions over concepts such as resource allocation and parallel processing, allowing users to implement complex robotic systems more easily.

B.3.3 Software Modules and Control Systems

The behaviour of the robot is governed by a control system. This control system consists of software modules that implement the desired control logic at multiple levels of abstraction. This ranges from low-level control of actuators which cause the robot to move in a particular direction, up to high-level autonomy that decides where the robot should explore next.

Mobility subsystem

In order to move through the environment, the robot uses the lidar scans of its surroundings to generate a *costmap*. The costmap is a 2D representation of the area around the robot, which tells it how difficult its immediate surroundings are to traverse. Given a target position and orientation, the mobility subsystem uses the costmap to plan a path

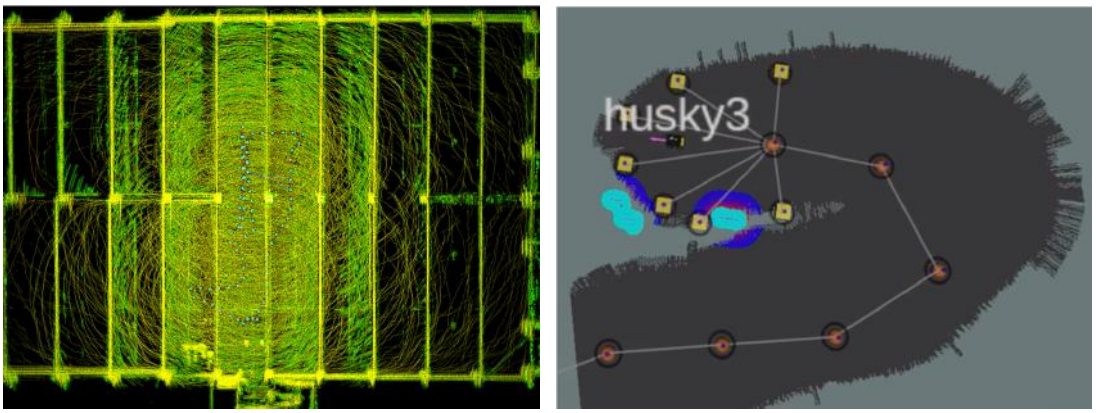
to the goal state, and then executes the appropriate commands to get the hardware to traverse this path.

State estimation subsystem

This module is responsible for maintaining an estimate of the robot's pose throughout its exploration. This is important for mobility, as well as mapping and navigation. Data is fused from multiple sensors to estimate the robot pose, including the IMU, lidar and cameras.

Mapping subsystem

This subsystem uses lidar-based SLAM to construct a map of the environment as the robot traverses through it, such as the one shown in Figure B.4 (a). It makes use of the state estimation output and the lidar point clouds to construct the map, closing loops in the trajectory based on a variety of data inputs. This subsystem is explored in greater detail in Chapter 5.



(a) (b)

Figure B.4. Example outputs from the mapping and autonomy subsystems. (a) A point cloud map created by the mapping subsystem, and (b) an IRM with waypoints (orange) and frontiers (yellow) from the autonomy subsystem.

Autonomy subsystem

This subsystem is responsible for high-level path planning and robot behaviours. This is primarily implemented by way of a graph called the information roadmap (IRM). The IRM is constructed on top of the pose graph from the mapping subsystem, and stores a series of *waypoints* along the path traversed by the robot. It also locates *frontiers*, nodes in the graph representing unexplored pathways. An example of an IRM is shown in Figure B.4 (b).

In order to explore the environment, autonomy system plans a path across consecutive waypoints to a target waypoint or frontier. The steps in this planned path are sent to the mobility subsystem, which handles local path planning in order to incrementally navigate toward the goal node.

Artifact Detection

The system also has an artifact detection module, which is responsible for identifying and localising artifacts relative to the robot using the RGB-D camera data. Recognition of artifacts is done using YOLO [108], a machine learning algorithm for real-time object detection. The relative position is then transformed into a global position using information from the state estimation and mapping subsystems.

B.3.4 On-Board Computers

In order to accomodate the significant computational power demanded by the sensors and software systems, each robot has a powerful on-board computer. The primary on-board computer used by CoSTAR is an Intel NUC with an Intel Core i7-8650U CPU.

Additionally, a Nvidia Jetson TX2 embedded computing device is used to provide GPU-accelerated execution of the YOLO algorithm for the artifact detection module, as this module would consume too much computational power if it ran purely on the main CPU.

B.3.5 Communications

One of the major challenges of autonomous robotics in subterranean environments is communication. In the absence of line-of-sight between the robot and base station, it is difficult to guarantee reliable radio communication over ranges of hundreds of metres.

In order to achieve a reliable communication link, the Husky robots periodically drop communication nodes, which form a mesh network that can relay radio signals between the base station and the robots. Each robot's on-board radio can also act as a communication node and relay signals through network.

Figure B.5. A communication node dropper mounted on the back of a Husky robot.

The communications nodes are deployed using a *comm node dropper* which is mounted on the back of the robots, as shown on a Husky robot in Figure B.5. The comm node dropper allows a single robot to carry up to 10 comm nodes, which can be triggered to drop either autonomously or through a direct command from the operator.

B.4 Conclusion

Working at NASA JPL gave me a broad set of new insights into the components and complexities of experimental robotics. Although the work described in the body of this

thesis is confined largely to a single robot subsystem, working as part of a fast-moving larger team developing a multi-robot system presented me with many opportunities to learn about the system as a whole. The scope of the DARPA Subterranean Challenge meant that the team's work encompassed all of the major areas of experimental robotics, including perception, mapping, autonomy, communications, mobility and state estimation. Working as part of this group allowed me to develop an understanding of how the different robotic subsystems interact, and the considerations involved in working on one subsystem which may be depended upon by several others.

The work presented in this thesis, as well as other work completed throughout the placement, therefore allowed me to develop my skills in robotic system design, research and problem solving, comprehensively addressing all learning outcomes of MTRX5700 Experimental Robotics.

APPENDIX C

Workplace Health and Safety

Across the United States, NASA operates 10 major field centres, in which an incredible breadth of engineering research and development is conducted. This work ranges from experimental rocket engine testing at the Stennis Space Centre, to cutting-edge aviation research at the Goddard Space Flight Centre, to developing robots to explore the solar system at the Jet Propulsion Laboratory. The experimental nature of the work often involves placing researchers and staff in close proximity to volatile chemicals, high-powered machinery, unpredictable autonomous robots and many other potential sources of danger. As such, NASA places strong emphasis on creating a culture in which safety is a top priority.

This chapter starts by providing an overview of the workplace health and safety procedures followed at NASA JPL, with emphasis on the practices most relevant to the CoSTAR team for the DARPA Subterranean Challenge, as part of which the work in this thesis was conducted. This is followed by case studies in which particular hazards are examined in detail, including overviews of the measures taken to reduce and mitigate risk.

C.1 The Hierarchy of Controls

At JPL, and across NASA as a whole, workplace health and safety is managed with reference to the hierarchy of controls, which provides a systematic description of the most effective ways to manage workplace hazards, as shown in Figure C.1. These range from eliminating the hazard entirely, which is considered the most effective, to providing workers with personal protective equipment (PPE), which is considered the least effective.

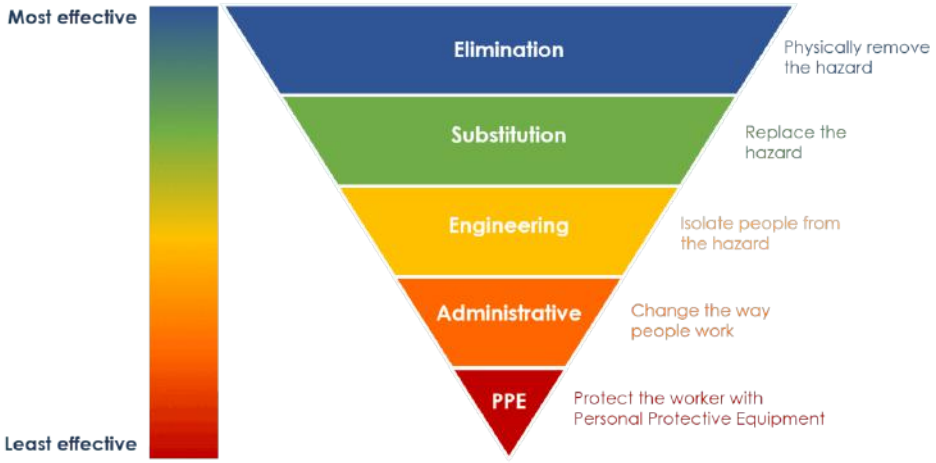


Figure C.1. The hierarchy of controls for workplace health and safety, as defined by the United States National Institute for Occupational Health and Safety (NIOSH) [109].

The levels of the hierarchy as applied to the work of CoSTAR are as follows:

Elimination: any hazards that are present and not needed for the testing are removed completely from the environment. In the case of on-site testing environments, this may include lab equipment left behind by previous occupants, personal computers or workstations, electrical cables, or any other objects that present tripping hazards. For off-site testing, the hazards that may be present depend heavily on the specific environment. Any personnel present at the site who are not required for CoSTAR testing are removed from the environment to ensure that their safety is not put at risk during testing.

Substitution: where hazards cannot be removed entirely, in some cases they may be replaced with a safer alternative. For example, testing may be performed on-site instead of off-site to avoid the increased risk associated with testing in more hazardous environments.

Engineering Controls: physical barriers such as temporary walls, netting or locked doors may be used to ensure isolation of the testing environment from people who do not need to be inside it.

Administrative Controls: team members and other people present for testing receive instructions on how to conduct themselves in the environment. This includes information such as emergency exit locations, hazards present in the environment, and any safe behaviour procedures that are specific to the environment. All personnel present for testing are also made aware of emergency contact procedures and the location of the nearest hospital.

Personal Protective Equipment (PPE): protective equipment such as hard hats, steel-capped boots or high-visibility clothing is often required, particularly for off-site testing in mines or caves.

The implementation of the hierarchy of controls at JPL is closely connected with the organisational structure of the company as a whole, which is described in detail in Appendix A. The safety controls are implemented by the more senior employees in the team structure, who take responsibility for creating a safe work environment for the rest of the team.

C.2 Lab Safety

The majority of CoSTAR work is conducted in a lab at JPL. This work includes electronics design, hardware construction, software development, and testing of aerial and ground robots. The safety of this environment is managed as specified in JPL internal

documentation [110]. This includes safety measures for different types of equipment in the lab, as shown in Table C.1.

Table C.1. Safety measures taken for lab equipment.

Object	**Hazards**	**Controls**
General lab equipment	Personal injury from tripping, knocking over equipment, misuse of equipment.	Must have a full-time employee in the lab at all times, nobody allowed in the lab alone.
Electrical equipment	Fire hazards, power surges damaging equipment, electrocution, blackouts.	Emergency power cut-off buttons, fire extinguishers in the lab, fire alarm buttons in the lab which call the on-site fire department.
Soldering station	Burns, inhalation of toxic fumes, damage to equipment.	Soldering iron is only operated by trained personnel, turned off whenever not in use, and never left unattended while turned on.
Ground robots	Personal injury, damage to lab equipment, damage to lab.	Only operated in presence of trained personnel, emergency stops (remote and on-board) must be implemented, operator must cycle through emergency stops in order to turn on robot, personnel in area are warned before testing begins, obstructions are removed from area.
Aerial robots	High speed crashes into people or lab equipment.	Only operated in presence of trained personnel, only operated within netted section of lab, drone attached to tether if operations permit.

JPL is located in California, a state that is frequently subjected to earthquakes of varying severity. This is accounted for in the lab, where equipment and boxes are never stacked above chest height in order to reduce the injury hazard that tall stacks of heavy objects might present during an earthquake.

C.3 Approval for Testing Environments

The research work conducted by CoSTAR frequently requires testing of autonomous robots in new environments. These environments may be on-site, such as a new building or lab at JPL, or off-site such as a mine or external facility.

In order to obtain approval for testing at a new location on-site, the team must first submit a Pre-Operational Safety Review (Pre-OSR) to the JPL Safety Office, which details the controls that will be put in place to ensure safe conduct of testing [110]. A Pre-OSR is also required for new types of testing in existing test environments, such as testing of flying vehicles in a lab previously only used for ground vehicles. For off-site testing, such as field trips to a mine or a cave, the team must submit an Off-Site Safety Plan for approval, which serves an equivalent purpose to the Pre-OSR.

In order to ensure that the conduct of the team in new environments meets the safety standards required by JPL, the Pre-OSR and Off-Site Safety Plan perform a risk assessment, identifying risks and proposing controls at *all levels* of the hierarchy.

Importantly, the Pre-OSR and Off-Site Safety Plan are not static documents – they are reviewed yearly and may change if required. For example, CoSTAR has in the past performed off-site testing at Eagle Mine, a gold mine in Julian, California. The Pre-OSR for Eagle Mine was updated when snow was first encountered at the mine, as this was a new hazard that had not been anticipated during the original review.

C.4 JPL Safety Culture

As a workplace in which new hazards arise all the time, and proper safety procedures are often complex and difficult to follow, it is important that JPL maintains accessible systems by which safety concerns can be reported and dealt with ethically. Incident reporting is handled through the JPL Safety Office, which ensures that protocols and policies are updated accordingly in the wake of accidents to minimise the risks present in future operations.

The Safety Office also has a system through which employees can anonymously report unsafe environments or behaviours. This ensures that employees who are uncomfortable with the safety standards in their workplace are able to bring this to the attention of the Safety Office without fear of retribution, thereby encouraging all employees to contribute to maintaining a safe environment.

Table C.2. The University of Sydney risk matrix [111].

			Potential Consequences				
			L6	L5	L4	L3	L2
			Minor injuries or discomfort. No medical treatment or measurable physical effects	Injuries or illness requiring medical treatment. Temporary impairment.	Injuries or illness requiring hospital admission.	Injury or illness resulting in permanent impairment.	Fatality
			Not Significant	**Minor**	**Moderate**	**Major**	**Severe**
Likelihood	Expected to occur regularly under normal circumstances	**Almost certain**	Medium	High	Very high	Very high	Very high
	Expected to occur at some time	**Likely**	Medium	High	High	Very high	Very high
	May occur at some time	**Possible**	Low	Medium	High	High	Very high
	Not likely to occur in normal circumstances	**Unlikely**	Low	Low	Medium	Medium	High
	Could happen, but probably never will	**Rare**	Low	Low	Low	Low	Medium

C.5 Risk Management at JPL

Any workplace hazard presents some form of *risk* to employees – the potential for consequences that negatively affect their health or safety. Risks are often analysed with reference to a risk matrix, such as the one used by the University of Sydney reproduced

in Table C.2. This table categorises risks based on likelihood of occurrence and severity of consequences. For example, a hazard might carry the risk of permanent impairment, but if the hazard is likely to never actually cause harm then the overall risk level is low.

In this section, we present case studies on several hazards that members of CoSTAR are frequently exposed to, as well as the measures taken to mitigate the associated risks.

C.5.1 Case Study: Autonomous Robot Safety

Many of the most hazardous aspects of CoSTAR's work are connected to the use of autonomous robots by the team. Figure C.2 shows the Telemax Pro and Husky A200, the two primary robots used by CoSTAR in the Tunnel Circuit of the DARPA Subterranean Challenge. The Telemax in particular is quite dangerous – it weighs 90 kg, has many pinch points and protruding features, and is capable of speeds up to 10 km/h. The Husky robots are less dangerous, weighing 60 kg and with top speeds of 3.6 km/h. During testing of the robots for autonomous exploration, bugs in the control algorithms can lead to erratic behaviours such as sharp accelerations and crashing into obstacles or people. The weight and speed of these robots mean they could easily cause injuries requiring hospital admission, and in an unregulated work environment one would expect this to occur before too long. As such, CoSTAR's autonomous robots present a high risk as per the University of Sydney risk matrix.

This risk is mitigated by introducing a variety of controls, as required by JPL safety policies:

- The Telemax is much more capable of traversing difficult terrain than the Husky, but is also much more dangerous. As such, CoSTAR substitutes the Telemax for a Husky in environments where it is sufficient, as it is much safer to operate.
- Testing areas are cordoned off so that employees who are not involved in the test are isolated from the hazard.
- A remote emergency stop is implemented on the robot, which is linked to an Xbox controller that is always in the hands of a safety operator while the robot is live.

Figure C.2. Ground robots used by CoSTAR in the Tunnel Circuit of the DARPA Subterranean Challenge: (left) Telemax Pro, and (right) four Husky A200s. Photo courtesy of Kamak Ebadi, NASA JPL.

The operator can immediately stop the robot from moving through a single button press if a dangerous situation arises.

- JPL employees who work with robots undergo online training to ensure that they are aware of the risks and know how to work safely in an environment with robots.
- If testing at night or in a hazardous environment, equipment such as high-visibility vests, hard hats and steel-capped boots are often mandatory for personal protection.

These measures represent controls from multiple levels of the hierarchy. With all controls in place, it becomes unlikely that an accident relating to autonomous robots should occur in normal circumstances, so the risk is reduced to low or medium.

C.5.2 Case Study: On-Site Testing

For some types of testing, CoSTAR requires environments with more complex topologies than those available in the main lab, so testing is performed in a cubicle farm in another building at JPL. This type of testing is *already* a substitution for testing in more dangerous environments such as mines and tunnels, but is not without its own risks.

This environment is not designated solely to CoSTAR, and thus opens up a new layer of hazards for JPL employees in the area who are not involved in the tests. These include personal injury from the robots, as well as damage to equipment and the work environment itself. Since there are employees in the environment who may not be aware of the testing being conducted, it is very likely for serious injury to occur over the course of testing, implying an initial risk level of very high.

Measures taken to control these risks include:

- Performing testing late at night (starting at 7pm), when most employees have vacated the test site.
- Moving objects in the surroundings out of the way that might obstruct the robots, such as backpacks on the floor or computer cables across corridors.
- Cordoning off the test site to isolate people from the hazards.
- Notifying employees in the area of the testing being conducted, and placing warning signs at entrances to the testing area. This ensures awareness of the risks, and allows employees to behave safely with full knowledge of the situation.

With these measures in place, the likelihood of an accident in the cubicle farm is scarcely higher than the likelihood of an accident in the CoSTAR lab. The risk has therefore been reduced to low or medium, through applying controls across multiple levels of the hierarchy.

C.6 Conclusion

The work conducted by the CoSTAR team at NASA JPL, as well as much of the other work conducted throughout the institution, gives rise to a work environment in which employees are exposed to many potential hazards on a daily basis. The comprehensive safety policies of NASA JPL, which are upheld by each of its employees, ensure that such

hazards are managed appropriately, allowing work to be conducted with minimal risk to the health and well-being of JPL employees or the wider community.

REFERENCES

[1] T. Chung, "DARPA Subterranean (SubT) Challenge," https://www.darpa.mil/program/darpa-subterranean-challenge, 2019, [Online; accessed 26-August-2019].

[2] Ackerman, Evan, "DARPA Subterranean Challenge: Teams of robots compete to explore underground worlds," https://spectrum.ieee.org/automaton/robotics/industrial-robots/subt-the-next-darpa-challenge-for-robotics, 2019, [Online; accessed 31-August-2019].

[3] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[4] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, vol. 2, 2006.

[5] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, June 2001.

[6] P. Newman and K. Ho, "SLAM-loop closing with visually salient features," in *proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 635–642.

[7] N. Muhammad and S. Lacroix, "Loop closure detection using small-sized signatures from 3D LIDAR data," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*. IEEE, 2011, pp. 333–338.

[8] A. Ratter and C. Sammut, "Local map based graph slam with hierarchical loop closure and optimisation," in *Proceedings of the Australasian Conference on Robotics and Automation (ACRA), Canberra, Australia*, 2015, pp. 2–4.

[9] J. Fuentes-Pacheco, J. Ascencio, and J. Rendon-Mancha, "Visual simultaneous localization and mapping: A survey," *Artificial Intelligence Review*, vol. 43, 2015.

[10] S. Ullman, "The interpretation of structure from motion," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 203, no. 1153, pp. 405–426, 1979.

[11] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[12] J. Sola, "Simulataneous localization and mapping with the extended Kalman filter," *Avery quick guide with MATLAB code*, 2013.

[13] J. Sola, T. Vidal-Calleja, J. Civera, and J. M. M. Montiel, "Impact of landmark parametrization on monocular EKF-SLAM with points and lines," *International journal of computer vision*, vol. 97, no. 3, pp. 339–368, 2012.

[14] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller, "An Atlas framework for scalable mapping," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, Sep. 2003, pp. 1899–1906 vol.2.

[15] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *The international journal of robotics research*, vol. 23, no. 7-8, pp. 693–716, 2004.

[16] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[17] F. Dellaert, M. Kaess *et al.*, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

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

[19] S. Thrun and M. Montemerlo, "The graph SLAM algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[20] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

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

[22] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3281–3288.

[23] S. Agarwal, K. Mierle, and Others, "Ceres Solver," http://ceres-solver.org, [Online; accessed 24-September-2019].

[24] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613.

[25] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1–4.

[26] H. Alismail, L. D. Baker, and B. Browning, "Continuous trajectory estimation for 3D SLAM from actuated lidar," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 6096–6101.

[27] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–606.

[28] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[29] K. S. Chong and L. Kleeman, "Accurate odometry and error modelling for a mobile robot," in *Proceedings of International Conference on Robotics and Automation*, vol. 4. IEEE, 1997, pp. 2783–2788.

[30] Z. M. Omohundro, *Robot configuration for subterranean modeling*. Carnegie Mellon University, 2009.

[31] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, 2014, p. 9.

[32] Q. Li, S. Chen, C. Wang, X. Li, C. Wen, M. Cheng, and J. Li, "LO-Net: Deep real-time lidar odometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8473–8482.

[33] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2174–2181.

[34] D. L. Lu, "Vision-enhanced lidar odometry and mapping," Ph.D. dissertation, Carnegie Mellon University Pittsburgh, PA, 2016.

[35] M. Bosse and R. Zlot, "Map matching and data association for large-scale two-dimensional laser scan-based SLAM," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 667–691, 2008.

[36] I. Rekleitis, J.-L. Bedwani, and E. Dupuis, "Autonomous planetary exploration using lidar data," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3025–3030.

[37] H. Xue, H. Fu, and B. Dai, "IMU-aided high-frequency lidar odometry for autonomous driving," *Applied Sciences*, vol. 9, no. 7, p. 1506, 2019.

[38] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3D lidar inertial odometry and mapping," *arXiv preprint arXiv:1904.06993*, 2019.

[39] "Velodyne Puck VLP-16 datasheet," Velodyne Lidar, California, USA.

[40] C. L. Gentil, T. Vidal-Calleja, and S. Huang, "IN2LAAMA: Inertial lidar localisation autocalibration and mapping," *arXiv preprint arXiv:1905.09517*, 2019.

[41] M. Attia and Y. Slama, "Efficient initial guess determination based on 3D point cloud projection for ICP algorithms," in *2017 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2017, pp. 807–814.

[42] M. Alatise and G. Hancke, "Pose estimation of a mobile robot based on fusion of IMU data and vision data using an extended Kalman filter," *Sensors*, vol. 17, no. 10, p. 2164, 2017.

[43] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Information fusion in navigation systems via factor graph based incremental smoothing," *Robotics and Autonomous Systems*, vol. 61, no. 8, pp. 721–738, 2013.

[44] H. Pfister, M. Zwicker, J. Van Baar, and M. Gross, "Surfels: Surface elements as rendering primitives," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 335–342.

[45] B. Oehler, J. Stueckler, J. Welle, D. Schulz, and S. Behnke, "Efficient multi-resolution plane segmentation of 3D point clouds," in *International Conference on Intelligent Robotics and Applications*. Springer, 2011, pp. 145–156.

[46] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments." in *Robotics: Science and Systems*, 2018.

[47] D. Droeschel and S. Behnke, "Efficient continuous-time SLAM for 3D lidar-based online mapping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[48] D. Droeschel, M. Nieuwenhuisen, M. Beul, D. Holz, J. Stückler, and S. Behnke, "Multilayered mapping and navigation for autonomous micro aerial vehicles," *Journal of Field Robotics*, vol. 33, no. 4, pp. 451–475, 2016.

[49] R. F. Salas-Moreno, B. Glocken, P. H. Kelly, and A. J. Davison, "Dense planar SLAM," in *2014 IEEE international symposium on mixed and augmented reality (ISMAR)*. IEEE, 2014, pp. 157–164.

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

[51] X.-F. Hana, J. S. Jin, J. Xie, M.-J. Wang, and W. Jiang, "A comprehensive review of 3D point cloud descriptors," *arXiv preprint arXiv:1802.02297*, 2018.

[52] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3D LIDAR point clouds," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2798–2805.

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

[54] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification," in *2011 IEEE international conference on robotics and biomimetics*. IEEE, 2011, pp. 2987–2992.

[55] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

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

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

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

[59] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "A fast and incremental method for loop-closure detection using bags of visual words," *IEEE Transactions on Robotics*, pp. 1027–1037, 2008.

[60] D. Filliat, "A visual bag of words method for interactive qualitative localization and mapping," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3921–3926.

[61] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, October 2012.

[62] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, "Pairwise consistent measurement set maximization for robust multi-robot map merging," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2916–2923.

[63] Y. Li, Y. Ushiku, and T. Harada, "Pose graph optimization for unsupervised monocular visual odometry," *arXiv preprint arXiv:1903.06315*, 2019.

[64] Q. Wu and J.-K. Hao, "A review on algorithms for maximum clique problems," *European Journal of Operational Research*, vol. 242, no. 3, pp. 693–709, 2015.

[65] B. Pattabiraman, M. M. A. Patwary, A. H. Gebremedhin, W.-k. Liao, and A. Choudhary, "Fast algorithms for the maximum clique problem on massive graphs with applications to overlapping community detection," *Internet Mathematics*, vol. 11, no. 4-5, pp. 421–448, 2015.

[66] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.

[67] T. A. Vidal-Calleja, C. Berger, and S. Lacroix, "Event-driven loop closure in multi-robot mapping," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1535–1540.

[68] Y. U. Cao, A. S. Fukunaga, and A. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous robots*, vol. 4, no. 1, pp. 7–27, 1997.

[69] I. Navarro and F. Matía, "An introduction to swarm robotics," *Isrn robotics*, 2012.

[70] A. Chapman and S. Sukkarieh, "A protocol for decentralized multi-vehicle mapping with limited communication connectivity," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 357–362.

[71] E. Nettleton, S. Thrun, H. Durrant-Whyte, and S. Sukkarieh, "Decentralised slam with low-bandwidth communication for teams of vehicles," in *Field and Service Robotics*. Springer, 2003, pp. 179–188.

[72] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-efficient decentralized visual SLAM," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2466–2473.

[73] J. Wan, S. Bu, J. Yu, and L. Zhong, "Distributed simultaneous localization and mapping for mobile robot networks via hybrid dynamic belief propagation," *International Journal of Distributed Sensor Networks*, vol. 13, no. 8, 2017.

[74] A. Cunningham, M. Paluri, and F. Dellaert, "DDF-SAM: Fully distributed SLAM using constrained factor graphs," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 3025–3030.

[75] M. Windolf, N. Götzen, and M. Morlock, "Systematic accuracy and precision analysis of video motion capturing systems—exemplified on the Vicon-460 system," *Journal of biomechanics*, vol. 41, no. 12, pp. 2776–2780, 2008.

[76] OptiTrack, "OptiTrack – motion capture for robotics," https://optitrack.com/motion-capture-robotics/, 2019, [Online; accessed 30-Oct-2019].

[77] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of SLAM algorithms," *Autonomous Robots*, vol. 27, no. 4, p. 387, 2009.

[78] O. Wulf, A. Nüchter, J. Hertzberg, and B. Wagner, "Benchmarking urban six-degree-of-freedom simultaneous localization and mapping," *Journal of Field Robotics*, vol. 25, no. 3, pp. 148–163, 2008.

[79] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[80] P. F. Alcantarilla, J. J. Yebes, J. Almazán, and L. M. Bergasa, "On combining visual SLAM and dense scene flow to increase the robustness of localization and mapping in dynamic environments," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 1290–1297.

[81] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 298–304.

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

[83] S. Khattak, F. Mascarich, T. Dang, C. Papachristos, and K. Alexis, "Robust thermal-inertial localization for aerial robots: A case for direct methods," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 1061–1068.

[84] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Autonomous Robots*, vol. 41, no. 2, pp. 401–416, 2017.

[85] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[86] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: 3D segment mapping using data-driven descriptors," *arXiv preprint arXiv:1804.09557*, 2018.

[87] X. Ji, L. Zuo, C. Zhang, and Y. Liu, "LLOAM: Lidar odometry and mapping with loop-closure detection based correction," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2019, pp. 2475–2480.

[88] E. Nelson, "B(erkeley) L(ocalization) A(nd) M(apping)!" https://github.com/erik-nelson/blam, 2016, [Online; accessed 8-November-2019].

[89] P. Lamon and R. Siegwart, "3D-Odometry for rough terrain–towards real 3D navigation," Swiss Federal Institute of Technology, Lausanne (EPFL), Tech. Rep., 2003.

[90] Intel, “Intel RealSense Tracking Camera T265,” https://www.intelrealsense.com/tracking-camera-t265/, 2019, [Online; accessed 10-December-2019].

[91] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, “Kimera: An open-source library for real-time metric-semantic localization and mapping,” 2019. [Online]. Available: https://github.com/MIT-SPARK/Kimera

[92] Defense Advanced Research Projects Agency, “DARPA Subterranean (SubT) Challenge,” https://www.subtchallenge.com/resources/SubT_Challenge_Tunnel_Rules.pdf, 2019, [Online; accessed 27-August-2019].

[93] J. Wang and E. Olson, “AprilTag 2: Efficient and robust fiducial detection,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4193–4198.

[94] G. Deak, K. Curran, and J. Condell, “A survey of active and passive indoor localisation systems,” *Computer Communications*, vol. 35, no. 16, pp. 1939–1954, 2012.

[95] R. C. Martin, “Design principles and design patterns,” *Object Mentor*, vol. 1, no. 34, p. 597, 2000.

[96] ——, *Clean code: a handbook of agile software craftsmanship*. Pearson Education, 2009.

[97] L. Li, J. Liu, X. Zuo, and H. Zhu, “An improved MbICP algorithm for mobile robot pose estimation,” *Applied Sciences*, vol. 8, no. 2, p. 272, 2018.

[98] Y. Tian, K. Khosoussi, M. Giamou, J. P. How, and J. Kelly, “Near-optimal budgeted data exchange for distributed loop closure detection,” *arXiv preprint arXiv:1806.00188*, 2018.

[99] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame, “DOOR-SLAM: Distributed, online, and outlier resilient SLAM for robotic teams,” *arXiv preprint arXiv:1909.12198*, 2019.

[100] T. Fushiki, "Estimation of prediction error by using k-fold cross-validation," *Statistics and Computing*, vol. 21, no. 2, pp. 137–146, 2011.

[101] Course and Unit of Study Portal (CUSP), University of Sydney, "MECH4601: Professional Engineering 2," https://cusp.sydney.edu.au/students/view-unit-page/alpha/MECH4601, 2019, [Online; accessed 6-October-2019].

[102] Engineers Australia, "Accreditation of Engineering Education Programs," https://www.engineersaustralia.org.au/About-Us/Accreditation, 2019, [Online; accessed 26-October-2019].

[103] NASA Jet Propulsion Laboratory, "The JPL Ethics Program," https://ethics.jpl.nasa.gov/index.cfm, 2019, [Online; accessed 27-October-2019].

[104] ——, *JPL Ethics Handbook*, 3rd ed., March 2019.

[105] Course and Unit of Study Portal (CUSP), University of Sydney, "MTRX5700: Experimental Robotics," https://cusp.sydney.edu.au/students/view-unit-page/alpha/MTRX5700, 2019, [Online; accessed 6-October-2019].

[106] T. Foote, "tf: The transform library," in *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, 2013, pp. 1–6.

[107] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[108] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[109] Centers for Disease Control and Prevention, "Hierarchy of Controls," https://www.cdc.gov/niosh/topics/hierarchy/, January 2015, [Online; accessed 18-September-2019].

[110] National Aeronautics and Space Adminisration (NASA), “JPL Internal Documents,” 2019.

[111] *Risk Assessment Form*, The University of Sydney, 2018.