Multi-session Visual Simultaneous Localisation and Mapping

John B. Mc Donald

A dissertation submitted for the degree of

Doctor of Philosophy

NUI MAYNOOTH

Ollscoil na hÉireann Má Nuad

Department of Computer Science

Faculty of Science and Engineering

National University of Ireland Maynooth

December 2013

Head of Department: Dr. Adam Winstanley

Supervisor: Prof. John J. Leonard,

Professor of Mechanical and Ocean Engineering,
Massachusetts Institute of Technology

Contents

List of Figures

List of Tables

Abstract

One of the principal aims of robotics is to develop robots that are capable of long term autonomy in unstructured and unknown environments. Such autonomy will only be achieved through algorithms that permit robots to perceive, interpret, and interact with the world they inhabit. The foundation to such algorithms is the ability to build and maintain a map of the environment and to estimate the robot's location relative to that map. This problem is referred to as *Simultaneous Localisation and Mapping* (or *SLAM*).

Over the past 25 years considerable progress has been made on the SLAM problem with a large number of solutions being reported in the literature. Although the majority of earlier systems depended on active ranging and proprioceptive sensors, more recently multiple approaches have been reported that rely purely on visual sensors. Visual sensors provide much richer measurements of the environment and bring with them a wealth of techniques from the field of computer vision in areas such as feature detection, tracking, and image matching. However, despite substantial recent progress in visual SLAM [103], many issues remain to be solved before a robust, general visual mapping and navigation solution can be widely deployed.

Central among these issues is *persistence* - the capability for a robot to operate robustly for long periods of time. As a robot makes repeated transits through previously visited areas, it cannot simply treat each mission as a completely new experiment, not making use of previously built maps. However, nor can the robot treat its complete lifetime experience as "one big mission", with all data considered as a single pose graph and processed in a single batch optimisation. In this thesis this problem is addressed through the development of a framework that achieves a balance between these two extremes, enabling the robot to leverage off the results of previous missions, while still adding in new areas as they are uncovered thereby improving its map over time.

The contribution of this thesis is the development of system for performing real-time multi-session visual mapping in large-scale environments. Multi-session mapping considers the above problem *i.e.* combining the results of multiple simultaneous localisation and mapping (SLAM) missions performed repeatedly over time in the same environment. The goal is to robustly combine multiple maps in a common metrical coordinate

system, with consistent estimates of uncertainty. Our work employs incremental smoothing and mapping (iSAM) as the underlying SLAM state estimator and uses an improved appearance-based method for detecting loop closures within single mapping sessions and across multiple sessions. A critical issue is how to pose the state estimation problem for combining the results of multiple mapping missions efficiently and robustly. We solve this problem by keeping each mission in its own relative frame of reference and employ spatial separator variables, called anchor nodes, to link together these multiple relative pose graphs.

The system architecture consists of a separate front-end for computing visual odometry and windowed bundle adjustment on individual sessions, in conjunction with a back-end for performing the place recognition and multi-session mapping. We provide a comprehensive quantitative analysis of the system's performance, demonstrating real-time multi-session visual mapping. The experimental datasets were captured using wheeled and handheld cameras and include indoor, outdoor, and mixed sequences captured over large-scale environments.

Acknowledgements

I would like to express my sincere gratitude to my supervisor John Leonard for his constant support, advice, patience and encouragement. His hospitality during my time as a visiting scientist at CSAIL made it one of the most enjoyable periods of my research career.

I would also like to thank my examining committee, Prof. José Neira, Dr. Simon Julier, and Dr. Kevin Kavanagh for their helpful comments and feedback on the thesis.

Whilst visiting CSAIL I had the pleasure of getting to know and work with a number of exceptionally talented people. A very special thanks to Michael Kaess for his invaluable discussions and advice throughout the research, and in particular for his help with the developments related to iSAM. I would also like to particularly thank Cesar Cadena for providing the place recognition module and for assisting with integrating it into the overall system. Thank you also to Maurice Fallon and Hordur Johannsson for their assistance with the data collection, and for providing the scripts for the Stata ground truth experiments. Many other members of the lab at CSAIL had a huge impact on my time there, including Abe Bachrach, Been Kim, and Rob Truax. Kind thanks to Mary Leonard for providing feedback on a number drafts of the thesis.

At Maynooth I would like to thank the members of my own research group for their patience over the past number of years including Tom Whelan,

Guillaume Gales, Karim Hammoudi, Eric McClean, and Yanpeng Cao. I would also like to say thank you to the members of staff in my Department and the wider University, and in particular Adam Winstanley, Charles Markham, and Ray O'Neill for their support during the process.

Research presented in this thesis was funded by a Strategic Research Cluster grant (07/SRC/I1168) by Science Foundation Ireland under the Irish National Development Plan. This funding is gratefully acknowledged.

To my parents I am thankful for everything they have done for me throughout my life and for every opportunity that they have made possible.

Finally and most importantly I would like to thank the three most wonderful people I know; my wife Lorraine, and my two daughters, Maia and Caroline. This thesis took us all on a journey, from Trim to Cambridge and back, and would not have been possible without your love, support, and encouragement. You kept me smiling through it all and for that I am eternally grateful.

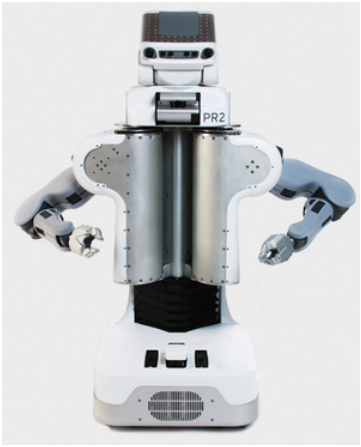
CHAPTER 1

Introduction

One of the principal aims of robotics is to develop robots that are capable of long term autonomy in unstructured and unknown environments. Such autonomy will only be achieved through algorithms that permit robots to perceive, interpret, and interact with the world they inhabit. The foundation to such algorithms is the ability to build and maintain a map of the environment and to estimate the robot's location relative to that map. This problem is referred to as *Simultaneous Localisation and Mapping* (or *SLAM*).

1.1 Motivation for Multi-Session Visual Mapping

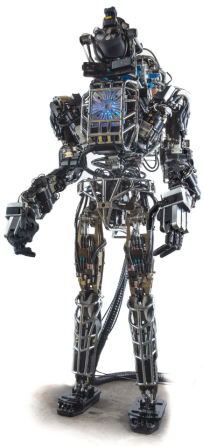
In a wider context mapping and localisation is a critical component in the development of situational awareness in any mobile computational system. Although situational awareness refers to a broad set of spatiotemporal cognitive capabilities, it can be identified with the ability to perceive the structure and state of the envi-



(a)

(b)

(c)



(d)

Figure 1.1: Example platforms with the potential to benefit from long-term SLAM. (a) PR-2 wheeled robot from WillowGarage[1]. (b) Astec Firefly unmanned aerial vehicle (UAV) from Ascending Technologies[2]. (c) Talos Land Rover LR3 autonomous vehicle MIT's entry in the DARPA Urban Grand Challenge[3]. (d) ATLAS humanoid robot from Boston Dynamics Incorporate (BDI)[4], currently one of three platforms in the DARPA Robotics Challenge[5]

ronment and to predict its future state. It is such capabilities within humans that permit intelligent interaction with one's environment. The benefits of emulating this capability in computational systems such as assistive robots, autonomous vehicles, security applications, *etc.* is immediately obvious.

The raw material of any SLAM algorithm is data captured from a sensor moving through the environment. Whether the sensor is a handheld camera carried by a user, or a laser scanner mounted on a mobile robot, mapping an environment involves using its data for both the determination of the scene structure and the motion of the sensor over time. As the robot explores the environment new areas are uncovered with the resulting local structure being added to the global map thereby providing the robot with an ever increasing representation of the world. The difficulty with the above

[1] `http://www.willowgarage.com/pages/pr2/overview`
[2] `http://www.asctec.de/uav-applications/research/products/asctec-firefly/`
[3] `http://grandchallenge.mit.edu/index.shtml`
[4] `http://www.bostondynamics.com/robot_Atlas.html`
[5] `http://www.theroboticschallenge.org/`

is that sensor data are prone to noise from multiple sources resulting in errors in both the map and the motion estimation process. Over time these errors accumulate resulting in drift in the estimated position and hence unbounded global error in the map.

The solution to this problem relies on *data association* between the sensor and the map. In fact data association occurs as two levels in the SLAM pipeline: locally and globally. As each new *frame* of measurements is acquired the motion relative to the map is typically computed by refining the estimated location such that the residual between the predicted and measured sensor data is minimised (based on some cost). In order to compute this residual we must first identify a mapping between the sensor data and the map. For odometry estimation we can assume a bounded motion between consecutive sensor acquisitions and hence the data association problem can be considered as a local search.

To correct for drift an alternative data association strategy must be employed whereby the system must first recognise when the robot revisits a previously mapped region, referred to as a *loop closure*. The significance of loop closures is that they constitute measurements of the global error. Here the location of the sensor can be computed relative to the immediate region of the map (i.e. through the above odometry estimation), and also relative to the region of the map due to the previous traversal of the region. Taking the difference between both estimates provides a measure of the drift. Alternatively the regions of the maps themselves can be aligned directly. In both cases instances of the data association problem arise.

The output of the above process is a set of positions and orientations, or poses, linked via constraints derived from the odometry and loop closure estimation. This graphical structure where the poses correspond to the nodes and the constraints correspond to edges between the nodes is referred to as a *pose graph* and leads to what has become the dominant paradigm for optimisation in the SLAM community.

The pose graph encodes a probability density function over the set of poses with the optimal trajectory typically computed as the *maximum likelihood* or *maximum a-posteriori* estimate of the model. Given that the map itself is computed from sensor data acquired in the frame of reference of the robot, computation of the global map is achieved by reprojecting the local structure at each pose into the optimised frame of reference.

SLAM is a long studied problem, starting with the work of Smith and Cheeseman [130]. Although the majority of earlier systems depended on active ranging and proprioceptive sensors, the past decade has seen a focus on approaches that rely purely on visual sensors. Visual sensors provide much richer measurements of the environment and bring with them a wealth of techniques from the field of computer vision in areas such as feature detection, tracking, image matching, and recognition. Given the above, strong arguments have been put forward that SLAM can now be considered a solved problem [46]. However, despite substantial recent progress in visual SLAM [103], many issues remain to be solved before a robust, general visual mapping and navigation solution can be widely deployed.

Central among these issues is *persistence* - the capability for a robot to operate robustly for long periods of time. As a robot makes repeated transits through previously visited areas, it cannot simply treat each mission as a completely new experiment, not making use of previously built maps. However, nor can the robot treat its complete lifetime experience as "one big mission", with all data considered as a single pose graph and processed in a single batch optimisation. In this thesis this problem is addressed through the development of a framework that achieves a balance between these two extremes, enabling the robot to leverage off the results of previous missions, while still adding in new areas as they are uncovered thereby improving its map over time.

The overall problem of persistent visual SLAM involves several difficult challenges

not encountered in the basic SLAM problem. One issue is dealing with dynamic environments, requiring the robot to correct for long-term changes, such as furniture and other objects being moved, in its internal representation. Another critical issue is how to pose the state estimation problem for combining the results of multiple mapping missions efficiently and robustly. In this thesis we concentrate on this this latter issue, referred to as *multi-session mapping*. In particular the central contribution of this thesis is the development of a system for performing real-time multi-session visual mapping in large-scale environments. Specifically the term multi-session mapping refers to combining the results of multiple simultaneous localisation and mapping (SLAM) missions performed repeatedly over an extended period of time in the same environment. The goal is to robustly combine multiple maps in a common metrical coordinate system, with consistent estimates of uncertainty.

Cummins defines the multi-session mapping problem as "the task of aligning two partial maps of the environment collected by the robot during different periods of operation [19]." We consider multi-session mapping in the broader context of life-long, persistent autonomous navigation, in which we would anticipate tens or hundreds of repeated missions in the same environment over time. As noted by Cummins, the "kidnapped robot problem" is closely related to multi-session mapping. In the kidnapped robot problem, the goal is to estimate the robot's position with respect to a prior map given no *a priori* information about the robot's position. In multi-session SLAM, in conjunction with this global localisation problem the robot should begin mapping immediately and upon localisation the map from the current session should be incorporated into the global map from previous sessions.

Also closely related to the multi-session mapping problem is the multi-robot mapping problem. In fact, multi-session mapping can be considered as a more restricted case of multi-robot mapping in which there are no direct encounters between robots (only indirect encounters, via observations made of the same environmental struc-

ture). On the other hand multi-session mapping may be considered a more general case of multi-robot mapping in that it does not require direct robot encounters and does not depend on any form of explicit collaboration between the sessions.

1.2 Is SLAM Solved?

Over the past 25 years considerable progress has been made on the SLAM problem with a large number of solutions being reported in the literature. State-of-the-art approaches have been demonstrated for a wide variety of sensors including sonar [140], lidar [85], radar [6], cameras [92, 26, 67, 93, 136], and even pedometers [2]. They have included systems that are capable of operating in indoor, outdoor, and mixed environments, attached to stable smooth moving platforms or highly dynamic handheld systems. Solutions exist for areas ranging in scale from small room sized environments [26, 67], to tens of kilometres [127].

In fact, some consider the development of accurate and robust solutions to the SLAM problem as one of the outstanding achievements of the robotics community over the past two decades [32]. The tremendous progress in SLAM, especially in the past decade, has lead some to claim that SLAM is now well-understood, with the remaining issues to primarily relating to implementation.

In 2010, Frese published a debate between Thrun and Neira to discuss the question "is SLAM solved?" [46]. While the initial response of Thrun was that "SLAM for static environments is solved, as far as basic research is concerned" it is clear from the conversation that there is a considerable gap between such systems and the type of SLAM systems required for long-term autonomy in real-world environments.

Neira emphasised the fact that current systems do not adequately address issues such as dynamics, semantics, and real-time operation over large environments. In fact on closer consideration both Thrun and Neira are in close agreement in the

debate. Thrun argues that the "real goal is to impact the world through robotic technology". By this argument we can assess whether SLAM has reached maturity by considering the level of adoption of SLAM in real-world robotic systems. By any measure this is still quite low, however the demand is ever growing given the push towards autonomous vehicles, assistive robotics, etc. Our position is that it is precisely these limitations, such as those identified by Neira, that are the barrier to SLAM achieving this goal. In the past five years, much of the research in SLAM has shifted to addressing many of these issues. To achieve long-term autonomy algorithms, the key challenges are:

Scalable Representations As the size of the environments of operation have increased a number of research groups have focussed on the development of scalable representations for performing SLAM. Perhaps the earliest principled approach to handling the issue of scale was due to Bosse et al. [9] where he developed a sub-mapping technique modelling the environment through a set of interconnected submaps. Such a representation allows a divide-and-conquer approach to optimisation allowing tractable mapping over large scales. Klein and Murray [67] used a similar approach in their parallel tracking and mapping visual SLAM algorithm, although the resulting system is only applicable in small scale environments. An alternative set of approaches make the choice of discarding metric information and focus on computing topological maps of the environment [20, 21]. Although such algorithms can operate over vast scales, the lack of a geometric model of the environment restricts their applicability in many situations. More recently Sibley et al. [125] presented the relative bundle adjustment (RBA) algorithm which permitted a trade-off between the requirement for global consistency and computation through the use of an adaptive windowed approach to optimising the map. In [127] Sibley et al. demonstrated this technique over what they refer to as vast scales. Johannsson et al. [58] address the

scale issue by only introducing poses into the pose graph if they differ in the position and orientation of pre-existing poses by a set threshold. Results are shown over a dataset consisting of tens of hours of video spanning the interior of multiple floors of a building. The resulting system operates in real-time and produces globally consistent metric maps.

Robustness Although scalability in terms of computation time is the most immediate consequence of the requirement for large scale autonomy, an equally important issue arises from the fact that as the scale of the mapping task increases in both space and time, so too does the potential for invalid measurements. Of particular importance here is the potential for outlier pose-to-pose constraints, for example, from the place recognition system. Recently a number of groups have been addressing this problem through the development of robust estimation techniques including the switchable constraints approach of Sünderhauf and Protzel [138, 139], the max-mixture model of Olson and Agrawal [112], and the realizing, reversing, recovering (RRR) algorithm of Latif et al. [78].

Dense Mapping & Semantics Even with the advances in SLAM over the last 25, years until recently we were still at a point where the resultant maps typically consisted of low-level geometric primitives and/or appearance based models derived directly from images. For example most visual SLAM systems, that use passive sensors, including the one presented in this thesis, produce sparse feature based representations of the environment. In order to create truly semantic maps of the environment researchers have recently begun focussing on dense mapping techniques. Newcombe et al. [105, 106] developed a dense visual SLAM system for desktop scale environments that exploited the programmability and massive parallelism of modern GPUs to compute a dense volumetric representation of the environment. More recently the advent of consumer-level RGBD sensors has allowed researchers to side step the chal-

lenges of monocular vision by directly measuring depth. Furthermore RGBD sensors can measure depth in featureless environments, and compute the depth through an on-board system-on-a-chip (SoC), thereby relieving the CPU of disparity estimation, and more importantly permitting the use of far more pixels in the image for odometry estimation and mapping. In [104] Newcombe et al. presented *KinectFusion* which has now become a landmark technique for dense mapping with an RGB camera. Building on this work the author's own group has addressed a number of limitations of the KinectFusion algorithm so as to allow it to be used over extended scale environments [149], incorporating visual odometry to overcome areas of low *geometric texture* [150], and solving the loop closure problem [151]. The importance of these approaches is that the resulting maps provide a far richer representation of the environment, and hence are more amenable to object discovery and segmentation [41], and hence higher-level processing. In the debate between Thrun and Neira, one of the comments made by Thrun was "what would really be exciting is a semantic understanding of the individual objects in the environment, their relationship to each other, and their characteristics." Recently Salas-Moreno et al. [122] presented a new *object oriented* 3D SLAM paradigm which they call SLAM++. Here they utilise an *a-priori* database of object categories to recognise, segment and track objects in an RGBD sequence. As objects are detected they become variables within an graph of both objects and camera poses where the edges encode camera-object and object-object pose constraints. Given such a representation allows SLAM at the object level.

1.3 Thesis Scope

The goal of this thesis is to create a real-time multi-session visual SLAM system to enable persistent operation of autonomous mobile robots. Of the large literature now published in SLAM discussed above, two prior contributions exemplify the goals of

this thesis investigation: (1) The work of Konolige and Bowman [70] on lifelong visual mapping and (2) Kim *et al.*'s work on multiple relative pose graphs for cooperative mapping [66]. Konolige and Bowman write:

> "The typical SLAM mapping system assumes a static environment and constructs a map that is then used without regard for ongoing changes. Most SLAM systems, such as FastSLAM, also require a single connected run to create a map. In this paper we present a system of visual mapping, using only input from a stereo camera, that continually updates an optimised metric map in large indoor spaces with movable objects: people, furniture, partitions, etc. The system can be stopped and restarted at arbitrary disconnected points, is robust to occlusion and localisation failures, and efficiently maintains alternative views of a dynamic environment. It operates completely online at a 30 Hz frame rate." [70]

Our work shares similar goals but adopts a somewhat different formulation, based on the anchor node representation, combined with iSAM, as previously developed in two dimensions by Kim *et al.*. Our work is the first example of an anchor node based SLAM system that (i) uses vision as the primary sensor, (ii) operates in general 6-DOF motion, (iii) includes a place recognition module for identifying inter and intra-session loop closures in general environments, and (iv) derives 6-DOF pose constraints from those loop closures within these general environments (i.e. removing the need for fiducial targets, as were used in [66]).

Our system architecture for achieving these goals is shown in Fig. 1.2 and consists of a separate front-end for computing visual odometry and windowed bundle adjustment on individual sessions, in conjunction with a back-end for performing the place recognition and multi-session mapping. The modularity of the architecture allows the user to efficiently swap different components with minimal effect on other components.

As part of this thesis we introduce the use of homogeneous point representation to iSAM allowing the smoothing to take advantage of distant scene points tracked over long baselines. In fact in the seminal work of Triggs et al. [146] the authors argue

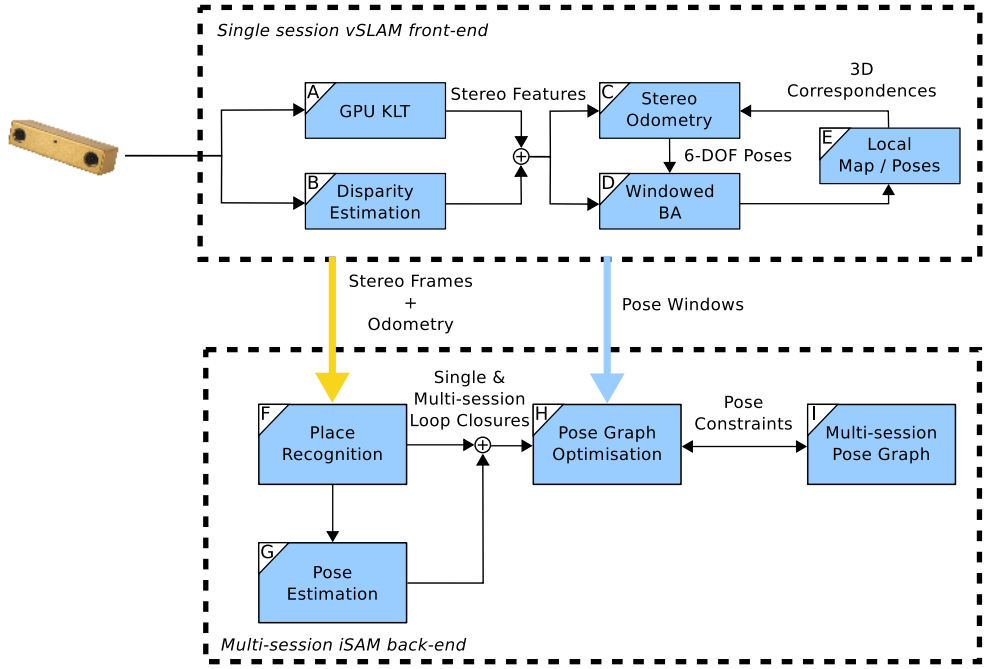


Figure 1.2: Multi-session visual SLAM system architecture. The major components of the system include: (A) 2-D feature tracker [153]; (B) dense stereo disparity estimation [10]; (C) stereo odometry [42, 109]; (D) windowed bundle adjustment using iSAM [61]; (E) local map representation; (F) visual place recognition [13]; (G) loop-closure pose estimation; (H) pose graph optimisation using iSAM; and (I) multi-session map pose graph representation.

that *the natural geometry and error model for visual reconstruction is projective rather than affine,* and as such failure to use a projective representation can be disastrous for distant points. In conjunction with the above, to represent the 6-DOF motion of the camera we employ a quaternion based representation of rotations.

Incorporating both homogeneous points and quaternions into the least squares optimisation of iSAM requires dealing with the over-parameterisation of the representations. In both cases we employ a minimal local parameterisation of the spaces through the use of an exponential map. That is, given that both representations can be identified with the unit sphere in $\mathbb{R}^4$ we can locally parameterise the optimisation in the tangent space $\mathbb{R}^3$ which can then be mapped to an update of the global parameterisation through the *same* exponential map.

We provide a comprehensive quantitative analysis of the system's performance, demonstrating real-time multi-session visual mapping. The experimental datasets were captured using wheeled and handheld cameras and include indoor, outdoor, and mixed sequences captured over large-scale environments.

1.4 Thesis Structure

The remainder of this thesis is structured as follows. Chapter 2 provides a general introduction to SLAM, starting with a formulation of the SLAM problem. Following on from this the chapter will provide a discussion of the three principal paradigms: extended Kalman filtering (EKF), particle filtering, and smoothing approaches. The chapter ends with a review of some of the relevant related approaches to visual SLAM.

Visual odometry, the capability to extract motion constraints directly from visual data, is central to any visual SLAM system. Chapter 3 provides an overview of the main components involved in stereo based visual odometry including: camera modelling and pose estimation, robust model fitting, feature detection and tracking,

and windowed bundle adjustment.

Chapter 4 describes the core contribution of the thesis – the design and implementation of a visual SLAM algorithm using iSAM and anchor nodes to achieves multi-session operation. A modular system design is presented, including a discussion of the design decisions taken to arrive at the component based architecture. Each component is treated in turn starting with the stereo odometry front-end, followed by the iSAM optimisation system, the place recognition system employed for loop closure detection and handling, and extension to incorporate multiple mapping sessions.

Chapter 5 provides an experimental evaluation for the system, documenting its performance for both single session and multi-session operation. Experimental results are provided for both wheeled robot and man-portable data collection. An analysis of the system's performance is provided, demonstrating the capability of the system to achieve real-time performance.

Chapter 6 reviews the contributions of the thesis and identifies future directions for the research. In particular the contributions are placed in the context extending SLAM for long-term autonomy and real-world applications, with a number of synergies between contemporary research identified and discussed.

CHAPTER 2

Simultaneous Localisation and Mapping

As was introduced in Chapter 1, SLAM poses the problem of computing a map of an unknown environment whilst simultaneously localising the sensor platform relative to the map. Visual SLAM (vSLAM) refers to the particular case of using camera(s) as the input sensor. In fact the SLAM problem predates the research efforts of the robotics community with earlier research in photogrammetry, where it is termed *bundle adjustment*, and more recently in the computer vision community [146], where it is termed *structure from motion* (or *SfM*). Although from an abstract point of view each of these terms refers to the same problem, there are key differences in the approaches and contributions made by each community. What differentiates the work of the SLAM community, at least initially, was the focus on online solutions to the problem. This paradigm stems from the fact that, as part of mobile robotics, SLAM typically focusses on situations where the sensor is mounted on a mobile robotic platform where the constructed map and the current location of the platform relative to that map forms part of a larger perception system. Hence batch approaches are

not applicable since at all points in time the robot requires a current estimate of the environment in order to complete its task.

A central element of the SLAM problem is the representation and quantification of uncertainty. This is due to the fact that the physical world in which the robot exists is inherently unpredictable and hence contains elements and events that are outside of the system's world model, or modelling capability. Furthermore given that sensors provide the link between the robot and the physical world, data from those sensors will inevitably be perturbed by measurement noise. Developing autonomous robots capable of dealing with such issues requires a probabilistic approach whereby both the map, $\mathbf{M}$, and the location of the robot over time within the map, $\mathbf{X}_t$, are modelled as random variables governed by some underlying joint density function.

$$p(\mathbf{X}_t, \mathbf{M}) \tag{2.1}$$

In this context, the SLAM problem becomes one of estimation of this distribution.

This chapter provides both a brief survey of the current state of the art of SLAM techniques and an introduction to the mathematics of the SLAM problem. Section 2.1 begins by providing details of the general probabilistic formulation of the SLAM problem. Next, Section 2.2 reviews the genesis of research into the SLAM problem and provides a historical review of the main milestones since that time. In particular this section will begin by reviewing EKF based approaches highlighting both the strengths and limitations of the earlier systems. Extensions to ameliorate these issues will be also reviewed. Aside from the EKF solutions, two other paradigms have received greatest attention in the literature: particle filtering (*a.k.a.* Monte-Carlo methods), and graph based approaches. Both of these approaches will be reviewed with the former focussing on the FastSLAM algorithms. The general graphical SLAM formulation will be presented of which there are now numerous algorithms. These ap-

proaches model the problem through the use of probabilistic graphical models where poses of the robot at discrete intervals and features of the environment are treated as latent variables, with measurements from sensors (*e.g.* odometry, visual features, laser returns, etc.) providing constraints between these variables. Such approaches have become predominant over the last decade and have been demonstrated in very large scale environments. Given that the SLAM method used in this thesis is the incremental smoothing and mapping (iSAM) approach of Kaess et al. [61], the section will focus on iSAM's graphical formulation both in terms of the underlying factor graph representation, and in terms of the optimisation approaches employed to permit incremental operation. The chapter completes by reviewing the visual SLAM problem.

2.1 Mathematical Formulation

As mentioned above, the SLAM problem aims to estimate the joint density of the map and trajectory of the robot as defined in Eq. 2.1. This estimation process is achieved through incremental integration of data from the robot platform including odometry control inputs, $\mathbf{U}$, and sensor measurements, $\mathbf{Z}$. We denote the dependency between all of these variables by rewriting Eq. 2.1 as the posterior:

$$p(\mathbf{X}_t, \mathbf{M} | \mathbf{U}_t, \mathbf{Z}_t) \tag{2.2}$$

where, the evolution of the problem over time is highlighted by the temporal index, t, for each of the variables $\mathbf{X}_t$, $\mathbf{U}_t$, and $\mathbf{Z}_t$. Note here the assumption that the environment is static and hence the lack of the subscript on $\mathbf{M}$. Here, $\mathbf{X}_t = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t\}$, represents all poses of the robot, $\mathbf{U}_t = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_t\}$, represents the control inputs, and $\mathbf{Z}_t = \{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_t\}$, represents the measurements at each time step.

Technically Eq. 2.2 represents the *full* SLAM problem due to the fact that it

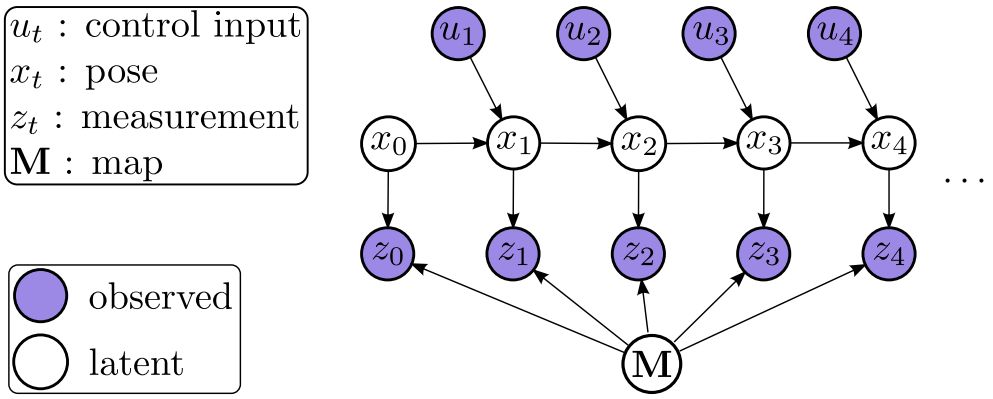


Figure 2.1: Bayesian network illustrating the structure of the SLAM problem.

includes both the current pose, $\mathbf{x}_t$, and all previous poses, $\mathbf{x}_{0:t-1}$. This is differentiated from the *online* SLAM problem where only an estimate of the current pose, $\mathbf{x}_t$, is maintained at each time step.

$$p(\mathbf{x}_t, \mathbf{M} | \mathbf{U}_t, \mathbf{Z}_t) \tag{2.3}$$

Note that the term online is somewhat of a misnomer and is used due to the fact that historically only batch solutions to the full SLAM problem were available. A number of the real-time incremental solutions to the full SLAM problem now exist.

Estimating either of the above posteriors becomes tractable through the use of both the product rule and Bayes rule resulting in a factorisation into a set of conditionally independent *motion* and *measurement* models. Here the measurement model, $p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{M})$, provides a distribution over the measurement space given the current pose and the map estimates. The motion model, $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$, on the other hand characterises the uncertainty in the position of the robot at time t, given the previous pose and the control input applied at that time. This factorisation is illustrated in the Bayesian network shown in Fig. 2.1.

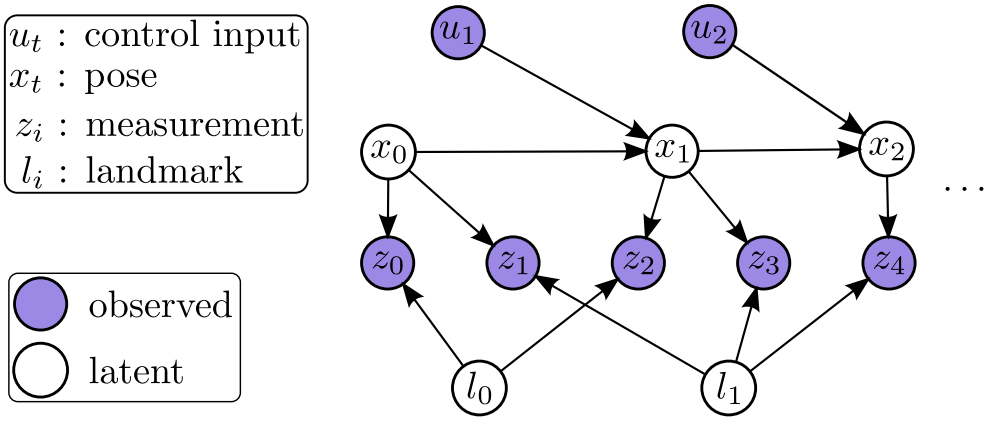


Figure 2.2: Bayesian network highlighting relationship between landmarks and measurements. Note that the landmarks are conditionally independent given the path.

2.1.1 Landmark-based SLAM

Although the above model is generic from the point of view of the map representation, a common representation (and the representation used throughout this thesis) is to consider the map as consisting of a set of landmarks, $\mathbf{M} = \{\mathbf{l}_0, \mathbf{l}_1, \ldots, \mathbf{l}_N\}$. Each measurement then corresponds to an observation of a particular landmark from a particular pose. Hence instead of collecting all sensor data at time t into a single measurement, $\mathbf{z}_t$, we associate multiple measurements with each pose; one for each landmark observed. To make this distinction explicit we now denote measurements, $\mathbf{z}_i$. This landmark based SLAM model is shown in Fig. 2.2. Note that although the most common approach is for each landmark to correspond to a single point location, this model does support more generic landmarks *e.g.* lines, or other geometric features.

From the structure of the Bayesian network shown in Fig. 2.2 we can immediately write the factorisation of Eq. 2.2 :

$$p\left(\mathbf{X}_t, \mathbf{M} | \mathbf{U}_t, \mathbf{Z}_t\right) \propto p\left(\mathbf{x}_0\right) \prod_{t=1}^{N-1} p\left(\mathbf{x}_t | \mathbf{u}_t, \mathbf{x}_{t-1}\right) \prod_{i=1}^{K} p\left(\mathbf{z}_i | \mathbf{x}_{t_i}, \mathbf{l}_{j_i}\right) \tag{2.4}$$

Solving the SLAM problem in this context now corresponds to computing the *maxi-*

mum a-posteriori (*MAP*) estimate of $\mathbf{X}_t$ and $\mathbf{M}$, given the measurements,

$$[\mathbf{X}_t^*, \mathbf{M}^*] = \arg\max_{\mathbf{X}_t, \mathbf{M}} \{p(\mathbf{X}_t, \mathbf{M}|\mathbf{U}_t, \mathbf{Z}_t)\} \tag{2.5}$$

In all cases in this thesis we will assume both the motion and measurement models to be Gaussian as follows:

$$\begin{aligned} \mathbf{x}_t &= g(\mathbf{x}_{t-1}, \mathbf{u}_t) + \boldsymbol{\omega}_t & (2.6)\\ p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) &\sim \mathcal{N}(g(\mathbf{x}_{t-1}, \mathbf{u}_t), \Omega_t) & (2.7)\\ &\propto \exp\left(-\frac{1}{2}\left(\mathbf{x}_t - g(\mathbf{x}_{t-1}, \mathbf{u}_t)\right)^T \Omega_t^{-1} \left(\mathbf{x}_t - g(\mathbf{x}_{t-1}, \mathbf{u}_t)\right)\right) & (2.8) \end{aligned}$$

$$\begin{aligned} \mathbf{z}_i &= h(\mathbf{x}_{t_i}, \mathbf{l}_{j_i}) + \boldsymbol{\sigma}_t & (2.9)\\ p(\mathbf{z}_i|\mathbf{x}_{t_i}, \mathbf{l}_{j_i}) &\sim \mathcal{N}(h(\mathbf{x}_{t_i}, \mathbf{l}_{j_i}), \Sigma_t) & (2.10)\\ &\propto \exp\left(-\frac{1}{2}\left(\mathbf{z}_i - h(\mathbf{x}_{t_i}, \mathbf{l}_{j_i})\right)^T \Sigma_i^{-1} \left(\mathbf{z}_i - h(\mathbf{x}_{t_i}, \mathbf{l}_{j_i})\right)\right) & (2.11) \end{aligned}$$

where, $g(\mathbf{x}_{t-1}, \mathbf{u}_t)$ and $h(\mathbf{x}_t, \mathbf{l}_j)$ are the deterministic motion and measurement functions and $\boldsymbol{\omega}_t$ and $\boldsymbol{\sigma}_t$ are an additive noise terms which are taken to be Gaussian with zero mean and covariances Ω_t and Σ_t, respectively.

2.1.2 Data Association

Reviewing Eq. 2.9 we note that each measurement, $\mathbf{z}_i$, can be considered as a constraint between a landmark, $\mathbf{l}_{j_i}$, and a pose, $\mathbf{x}_{t_i}$. Computing these constraints requires solving the *data association problem*; one of the most important and challenging problems in SLAM [102]. In simple terms the data association problem involves identifying correspondences between measurements acquired at different time instances (*i.e.*, so as to permit collection of all measurements for each landmark into separate subsets).

Often it is useful to consider the data association problem as three separate sub-problems. The first is to identify corresponding measurements between consecutive poses (or poses that are close in time). The second is to identify new landmarks *i.e.* which have not been previously observed. And finally the third is to identify correspondences between measurements when an area is revisited.

The benefit of distinguishing along these lines is that the first two here are typically handled by a separate feature detection and tracking system that takes advantage of the spatial proximity of features between poses in the sensor frame of reference. An example of this is that of tracking a visual feature in a video sequence captured from a moving camera. Assuming an adequately high frame rate, the induced motion of a feature due to the motion of the camera should be such that the features detected in a given frame should have a bounded displacement in the subsequent frame. Furthermore given the modelling of uncertainty, a visual SLAM system should be able to make an informed decision of the bounds on this displacement on a feature-by-feature basis.

This leads to the second task of identifying new features. A common approach to solving this problem is to take features that do not fall within these uncertainty bounds as being new features [31]. This test is normally coupled with a geometric or visual descriptor matching step to ensure robustness.

When revisiting a previously mapped area the accumulated drift can be significant and as a result the above approaches do not apply. Instead the approach is normally to first perform a global recognition step whereby the current sensor data is matched against a database of previously visited areas [20]. This process may identify a number of putative matches. Each of these matches is then compared against the current sensor data in order to derive a set of local correspondences. The output of this latter process is used to provide a measure of confidence in each of the matches. The final match is selected as the one with the highest confidence that is above a predefined

threshold, which is typically set quite conservatively to avoid false positives.

2.2 EKF Approaches

The earliest work on the SLAM problem within the robotics community is typically credited to Smith and Cheeseman [130] where they addressed the representation of uncertain spatial relationships through what they defined as *approximate transformations* (*AT*'s). AT's characterised the transformations between coordinate frames as random variables which they estimated through their mean and covariance. Importantly here the covariance matrix provided a measure of the uncertainty over the variables of the transformation.

In [131] Smith et al. built on their previous work by defining the *stochastic map* which captured the relationships between all spatial variables (*e.g.* the structure of the environment, and, the location of the sensor) in a single state vector. This representation allowed the definition of a probability density function over this state space, again estimated through the mean and covariance of the vector. Critically the covariance matrix models not only the variance in the individual variable estimates but also the correlations between the variables.

As the vehicle moves, uncertainty in the motion accumulates resulting in drift in the global pose estimate. Given that measurements are represented relative to the vehicle frame their global uncertainty is correlated with that of the vehicle and hence grows in tandem. This network of correlations is effectively what is captured by the stochastic map.

As explained in [32], successive sets of measurements result in greater and greater correlations between variables. From a representational point of view this is quantified in the off diagonal elements of the covariance, and as result, over time, the covariances becomes fully dense requiring accurate modelling in order to avoid inconsistencies [18].

Another important contribution of Smith et al. [131] was an EKF approach to incrementally constructing and updating the map which constituted the first solution to the SLAM problem. This work signalled the start of a period of intense activity into filtering solutions to SLAM problem. A thorough review of the chronology of developments during this period can be found in Durrant-Whyte and Bailey [32].

Although the EKF approach to SLAM has received considerable focus, and in fact was the first paradigm employed in vSLAM ([26]), it does suffer from a number of limitations. Principal amongst these is that although the state vector grows linearly with the map, the covariance grows quadratically as does both the space and computational complexity. This means that EKF SLAM systems are limited in the scale of the areas over which they can operate.

Dealing with this issue was a key driver for early SLAM research. Early approaches to mitigate this issue include the compressed filter algorithm developed by Guivant and Nebot [49], sequential map joining developed by Tardos *et al.* [140], and the constrained local submap filter developed by Williams *et al.* [152]. Each of these three approaches reduced the computational burden by a constant factor but still had quadratic complexity. Estrada *et al.* [35] improved on this by achieving linear complexity with their Hierarchical SLAM algorithm.

One of the key shortcomings of EKF based approaches is that as measurements accumulate all of the variables in the map become completely correlated and hence the covariance becomes dense. An alternative set of approaches is based on using the inverse form of the covariance, also known as the information matrix [143, 36, 148]. The benefit of using the information matrix is that whilst the covariance becomes dense over time, the information matrix maintains its sparsity as long as the entire robot trajectory is maintained in the state vector. In fact this key insight is also exploited in pose graph methods (see Section 2.4). Eustice exploited this insight to realise the exactly sparse delayed state filter, which was applied to underwater camera

data to reconstruct the wreck of the Titanic [36]. Walter developed the exactly sparse extended information filter (ESEIF) which discarded selected odometry measurements to maintain sparsity whilst performing landmark based SLAM.

An issue with both the EKF and EIF approaches, as shown by Julier and Uhlmann [59], is that over extended timescales, the state estimates are guaranteed to become inconsistent. Furthermore, the representation employed assumes a unimodal distribution which can be adequately approximated through the mean and covariance. This limitation means that the approach is unable to cater with ambiguity and aliasing within the environment and hence is unsuitable for global localisation, (*e.g.* the *kidnapped robot* problem), which often involve a number of distinct hypotheses [144].

2.3 Particle Filtering

Montemerlo et al. [98] present the FastSLAM algorithm that uses a non-parametric particle filter representation which permits approximation of multimodal distributions and is therefore much more robust to problems such as the above. FastSLAM represents the joint density (see Eq. 2.1) over the path and the map as a set of particles. Each particle consists of an estimate of the path and a series of independent Gaussian estimates of the locations of each of the landmarks within the map. More specifically at time t the posterior is defined as a set of sampled particles, $\mathbf{S}_t = \left\{\mathbf{S}_t^0, \ldots, \mathbf{S}_t^{K-1}\right\}$. Here the i^{th} particle is defined as, $\mathbf{S}_t^i = \left\{\mathbf{X}_t^i, \left(\boldsymbol{\mu}_{t,1}^i, \Sigma_{t,1}^i\right), \ldots, \left(\boldsymbol{\mu}_{t,N}^i, \Sigma_{t,N}^i\right)\right\}$, where $\mathbf{X}_t^i$ is the full path estimate and $\left(\boldsymbol{\mu}_{t,j}^i, \Sigma_{t,j}^i\right)$ are the mean and covariance of landmark j.

An important aspect of the FastSLAM algorithm is the use of Rao-Blackwellization whereby the path is estimated using a modified particle filter and the landmarks are estimated in closed-form using a set of independent Kalman filters. A complete iteration of the algorithm therefore consists of a path update step followed by a map update step. The path update step first involves using the known control input, u_t, to

predict the latest pose for each particle by sampling from motion model, $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$. An important point to note is that although the particles within FastSLAM represent the full trajectory, the algorithm is still a type of filtering in that only the most recent pose is used in prediction step. An importance weight is then computed for each resulting particle by evaluating the likelihood of the measurements, $p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m})$. These weights are then used in a *resampling* step to generate a new set of particles by sampling with replacement from the current set, where the probability of sampling a given particle is given the by normalised importance weight.

Given an updated path each of the landmarks become independent and hence can be estimated separately (see Fig. 2.2). This is achieved by a set of NK Kalman filters (*i.e.* N KF's for each of the K particles) which update the $\left(\boldsymbol{\mu}^i_{t,j}, \Sigma^i_{t,j}\right)$ vectors. Note that unlike EKF SLAM, here the Kalman filters operate over a much lower fixed dimension and therefore are computationally efficient. Given that a weakness of particle filters lies in the fact that the number of particles required to approximate the distribution grows exponentially with the number of dimensions, this combination of conditional independence with Rao-Blackwellization is critical to making the FastSLAM algorithm tractable. Furthermore, although the above suggests a update complexity of $O(NK)$, this is reduced to $O(N \log K)$ through the use of a tree based data-structure in the management of the map [98].

2.4 Smoothing Approaches

Despite the considerable breath of approaches based on filtering, a fundamental issue with applying filtering approaches in the context of the non-linear SLAM problem is that they have been shown to be inconsistent [59]. An alternative SLAM paradigm, sometimes referred to as *smoothing* [28], is to represent the problem as a graph of non-linear constraints where nodes in the graph represent the latent variables and

edges correspond to soft constraints between the variables *e.g.* due to measurements or odometry information.

Such a representation was first introduced by Lu and Milios [84] where they defined the *pose graph* as a series of poses connected via pose relations, which were in turn derived by alignment of the sensor scan data of poses involved in the relationship. In this work the authors did not represent the scene structure explicitly and instead restrict the estimation problem to the pose graph. Two types of edges between pose nodes are identified: (i) odometry constraints between consecutive poses, and, (ii) loop-closure constraints due to revisiting a previously mapped region of the environment. On computation of the smoothed pose graph the map is constructed by projecting each lidar scan into the local frame of the corresponding pose.

Since then a considerable collection of graph based approaches have been reported in the literature [50, 28, 113, 75, 72]. In fact over the last decade such graphical approaches have become the predominant method in large scale SLAM systems, demonstrating mapping solutions at much greater scales than those shown by filtering approaches.

The reason for this is that with graphical approaches the optimisation is performed over the current pose and all previous poses, and hence solves the full SLAM problem. Maintaining all poses in this fashion results in a sparsity in the underlying matrix representation of the graphical structure which can in turn be exploited through the use of sparse matrix optimisation techniques [25]. Although initially only batch solutions were available, more recently real-time online solutions that exploit incremental solvers have recently been presented in [61, 63].

Although many graphical representations have been used in formulating the SLAM problem, given the focus of this thesis on incremental smoothing and mapping (iSAM), we will follow the approach of Dellaert and Kaess [29] and use the factor graph representation (see Chapter 8 of Bishop [8] for an excellent tutorial introduction).

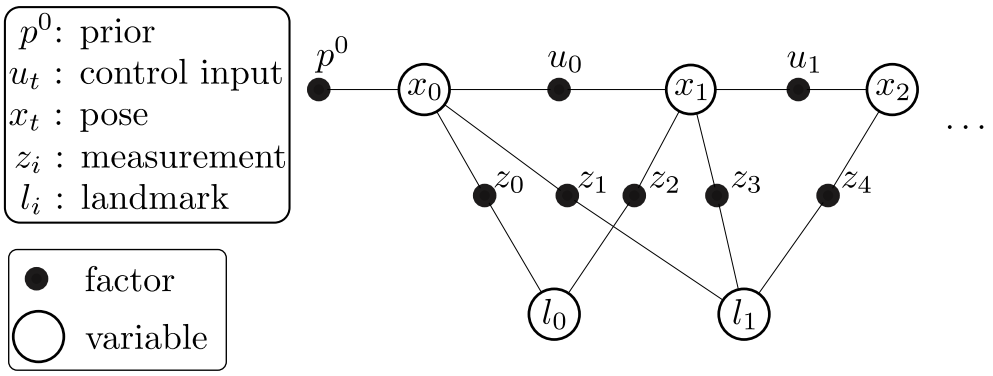


Figure 2.3: Factor graph of the SLAM problem corresponding to the Bayesian network shown in Fig. 2.2.

A factor graph is a bipartite graph consisting of two types of nodes: variable nodes and factor nodes. Edges in the factor graph connect variable nodes to factor nodes, where a factor node represents a function over all of the variables to which it is connected. A factor graph then represents a factorisation of a joint distribution over the variables, $\mathbf{x}$, as the product of all of the factors [8],

$$p(\mathbf{x}) = \prod_s f_s\left(\mathbf{x}_s\right) \tag{2.12}$$

where, $\mathbf{x}_s$, is the subset of variables in $\mathbf{s}$ associated with the factor f_s. So, whereas other graphical models, such as Bayesian networks, represent the factorisation of the joint distribution as a product of conditional probability distributions, with factor graphs the factors are general functions and hence permit more general factorisations. Furthermore by making the factors explicit (*i.e.* as nodes), factor graphs can include multiple separate factors involving the same subset of latent variables.

In order to derive the graphical formulation of the SLAM problem using factor graphs we start with the general formulation provided in Section 2.1 and in particular equations 2.4 and 2.5. The factor graph for the SLAM problem is shown in Fig. 2.3

(corresponding to the Bayesian network shown in Fig. 2.2) where,

$$p\left(\mathbf{X}_t, \mathbf{M} | \mathbf{U}_t, \mathbf{Z}_t\right) = \prod_s f_s\left(\mathbf{x}_s\right) \tag{2.13}$$

$$= f\left(\mathbf{x}_0; \mathbf{p}^0\right) \prod_{n=1}^{N-1} f\left(\mathbf{x}_n, \mathbf{x}_{n-1}; \mathbf{u}_n\right) \prod_{k=1}^{K} f\left(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}; \mathbf{z}_k\right) \tag{2.14}$$

As can be seen from the figure three factor types are shown; measurement factors, $f(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}; \mathbf{z}_k)$, pose-to-pose factors, $f(\mathbf{x}_n, \mathbf{x}_{n-1}; \mathbf{u}_n)$, and an anchor factor, $f(\mathbf{x}_0; \mathbf{p}^0)$, that acts as a prior on the position of the initial pose. Each of these factors is a function of the unknown variables, parameterised by the measurements. Note that the pose-to-pose factors are constructed by both incremental motion constraints (*i.e.* between poses from neighbouring time instants) and from *loop closure* constraints (not shown in the figure).

Assuming the Gaussian motion and measurements models from equations 2.7 and 2.10 we can use the standard technique of recasting Eq. 2.5 as minimising the negative log likelihood as

$$\left[\mathbf{X}_t^*, \mathbf{M}^*\right] = \underset{\mathbf{X}_t, \mathbf{M}}{\arg\min} \left[-\log\left(\prod_s f_s\left(\mathbf{x}_s\right)\right)\right] \tag{2.15}$$

$$= \underset{\mathbf{X}_t, \mathbf{M}}{\arg\min} \sum_{n=1}^{N-1} \left[\left(\mathbf{x}_n - g(\mathbf{x}_{n-1}, \mathbf{u}_n)\right)^T \Omega_n^{-1} \left(\mathbf{x}_n - g(\mathbf{x}_{n-1}, \mathbf{u}_n)\right)\right] + \sum_{k=1}^{K} \left[\left(\mathbf{z}_k - h(\mathbf{x}_{i_k}, \mathbf{l}_{j_k})\right)^T \Sigma_k^{-1} \left(\mathbf{z}_k - h(\mathbf{x}_{i_k}, \mathbf{l}_{j_k})\right)\right] \tag{2.16}$$

Equation 2.16 corresponds to the maximum likelihood (ML) SLAM estimate which can be estimated using non-linear optimisation techniques such as Gauss-Newton or Levenberg-Marquardt [89]. Such approaches involve iterating the process of linearising around the current estimate and then solving for an update to the estimate that reduces the cost. This iterative update of the current estimate continues until a minimum is reached (or some other stopping criteria is met).

The remainder of this section provides a summary of the approach used by iSAM. More details can be found in Kaess [60], Kaess et al. [61] and Dellaert and Kaess [29]. Taking $\theta_n = (\mathbf{x}_n, \mathbf{l}_n)$ as the combined poses and landmarks we can linearise Eq. 2.16 about the current estimate to arrive at a cost function of the form

$$\begin{aligned} C(\boldsymbol{\delta\theta}) &= \sum_{n=1}^{N-1} \|G_{n-1}\boldsymbol{\delta}\mathbf{x}_{n-1} - \boldsymbol{\delta}\mathbf{x}_n + g(\hat{\mathbf{x}}_{n-1}, \mathbf{u}_n) - \hat{\mathbf{x}}_n\|_\Omega^2 \\ &\quad + \sum_{k=1}^{K} \|H_k\boldsymbol{\delta}\mathbf{x}_{i_k} + J_k\boldsymbol{\delta}\mathbf{l}_{j_k} + h(\hat{\mathbf{x}}_{i_k}, \hat{\mathbf{l}}_{j_k}) - \mathbf{z}_k\|_\Sigma^2 \end{aligned} \tag{2.17}$$

where $\|\mathbf{x}\|_\Gamma^2 = \mathbf{x}^T\Gamma^{-1}\mathbf{x}$ corresponds to the squared Mahanalobis distance, $\boldsymbol{\delta\theta} = (\boldsymbol{\delta}\mathbf{x}, \boldsymbol{\delta}\mathbf{l})$ is the combined update to the poses and the landmarks, and $\hat{\mathbf{x}}$, $\hat{\mathbf{l}}$ denotes the current estimates. G_{n-1} is the Jacobian of g with respect to $\mathbf{x}_n$ evaluated at $\hat{\mathbf{x}}_{n-1}$. Whilst H_k and J_k are the Jacobians of h with respect to $\mathbf{x}_{i_k}$ and $\mathbf{l}_{j_k}$, respectively, again evaluated at the current estimate.

Both terms in Eq. 2.17 can be simplified and combined to form a standard least squares cost function

$$C(\boldsymbol{\delta\theta}) = \|A\boldsymbol{\delta\theta} - \mathbf{b}\|^2 \tag{2.18}$$

Here the measurement Jacobian, A, has a block structure collecting together the G, H, and J, matrices, and $\mathbf{b}$ is a column vector of residuals between measurements and the estimates. Hence the measurements (or factors) correspond to contiguous blocks of rows of A whilst the variables correspond to contiguous blocks of columns. Note that in Eq. 2.18 we have absorbed the covariances into the Jacobian matrix and residual vector. This is achieved as follows: if we take the Mahanalobis distance for a given variable $\mathbf{x}$ to be given by $\|\mathbf{x}\|_\Sigma^2 = \mathbf{x}^T\Sigma^{-1}\mathbf{x}$, then by the change of variables $\mathbf{x}' = \Sigma^{-\frac{T}{2}}\mathbf{x}$ we have $\|\mathbf{x}\|_\Sigma^2 = \|\mathbf{x}'\|^2$.

The key to efficient solutions to Eq. 2.18 is to exploit the sparse structure of A. For example the factor graph shown in Fig. 2.3, will result in the block matrix form

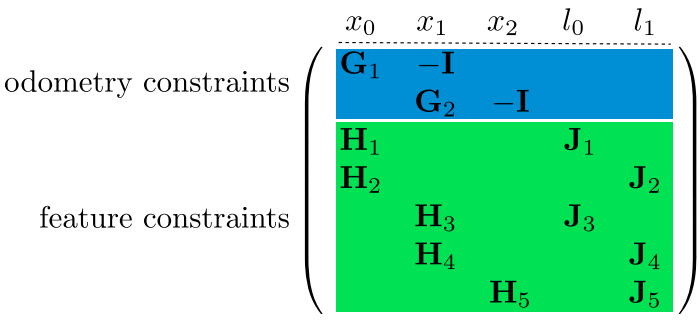


Figure 2.4: Measurement Jacobian of the SLAM problem corresponding to the factor graph shown in Fig. 2.3.

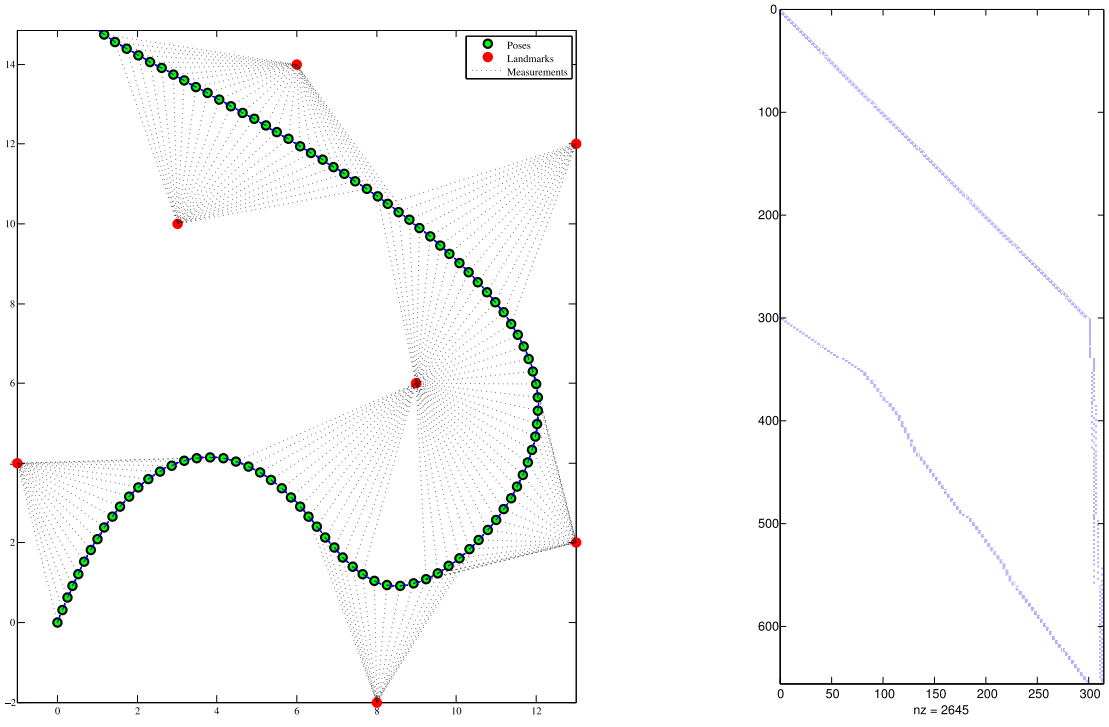


Figure 2.5: Example of a more complex synthetic SLAM scenario with the corresponding measurement Jacobian where the sparse structure of odometry followed by feature measurement constraints is clearly visible.

shown in Fig. 2.4. Here the separation of the odometry versus the feature constraints shows two differing sparsity patterns. A more complex simulated example is shown in Fig. 2.5 where this sparse structure becomes even more pronounced. Here the system consists of 100 poses, 7 landmarks, and 177 measurements. Given the poses are 3D variables, whilst the landmarks are 2D, the resulting measurement Jacobian has dimensions $(100 * 3 + 177 * 2) \times (100 * 3 + 7 * 2) = (654 \times 314)$.

The standard approach to computing $\boldsymbol{\delta\theta}$ is to set the derivative of C with respect to $\boldsymbol{\delta\theta}$ equal to zero, resulting in the well-known *normal equations*

$$A^T A \boldsymbol{\delta\theta}^* = A^T \mathbf{b} \tag{2.19}$$

Solving for $\boldsymbol{\delta\theta}^*$ is typically achieved via Cholesky decomposition of $A^T A$. The Cholesky decomposition results in a factorisation of the form $A^T A = R^T R$, where R is a right upper-triangular matrix. Given this decomposition we solve for $\boldsymbol{\delta\theta}^*$ by first solving $R\mathbf{y} = A^T \mathbf{b}$ followed by $R^T \boldsymbol{\delta\theta} = \mathbf{y}$.

Alternatively, and as is the case in iSAM, one can apply QR-factorisation directly to the measurement matrix $A = Q \begin{bmatrix} R \\ \mathbf{0} \end{bmatrix}$, where Q is an orthogonal matrix. Here the solution to Eq. 2.18 is given by [60]:

$$R \boldsymbol{\delta\theta}^* = Q^T \mathbf{b} \tag{2.20}$$

where $\boldsymbol{\delta\theta}^*$ is computed through back-substitution.

A key innovation of Kaess's PhD thesis research was to develop an efficient and exact solution for incrementally updating the QR factorisation, recalculating only the matrix entries that actually change [60]. Periodic variable reordering is used to prevent unnecessary fill-in in the factor matrix due to when loops get closed. Kaess *et al.* also developed efficient algorithms to access the estimation uncertainties of

interest based on the factored information matrix [61]. More recently, the iSAM2 algorithm was developed, which uses the Bayes Tree data structure to perform fluid relinearisation and periodic variable reordering [63].

Two open source iSAM implementations are currently available – Kaess's iSAM library implemented at MIT and the GTSAM library made available in 2012 by Dellaert's group at Georgia Tech. This thesis uses the former implementation, which has also been deployed for a wide range of applications [39, 66, 38, 40, 147, 58, 4, 65, 64, 76]. A variety of other packages for pose graph optimisation are available, for example at `http://www.openslam.org`. At present the most widely used alternative to iSAM or GTSAM is likely the g2o graph optimisation algorithm developed by Kümmerle *et al.* [75].

An important extension of pose graph SLAM is to perform mapping with multiple robots. Notable examples include Fox *et al.* [45, 71], Howard [56], and Cunningham *et al.* [22, 24]. While most of the multi-robot SLAM literature has been focussed on land robots using lidar, multi-robot visual SLAM systems have recently been created [154, 43]. Multi-robot SLAM is highly relevant to the multi-session SLAM investigated in this thesis, and will be discussed in more detail in Chapter 4.

2.5 Visual SLAM

Although early SLAM systems mainly used active sensors such as sonar, radar, and lidar, over the last 15 years the principal focus has been on the development of visual SLAM systems. This effort started with the work of Davison [27] where he developed an active stereo based SLAM system capable of building a consistent sparse map of landmark features, using an actively steered stereo camera system. A key aspect of this work was maintaining the full covariance matrix for the Kalman filter state estimates of camera and landmark locations. This was not performed in previous

approaches, such as the work of Ayache and Faugeras [3]. In subsequent work, Davison and colleagues achieved real-time SLAM with a monocular camera [26] and introduced the highly effective inverse depth parameterisation [99]. Concurrently, a wider group of visual SLAM algorithms and systems emerged [103].

A key milestone in the evolution of visual SLAM was the creation of visual mapping systems that could achieve real-time operation for extended input sequences in limited scale environments Klein and Murray [67], Eade and Drummond [33], and Davison [26], Strasdat et al. [135]. Klein and Murray's system is highly representative of this work, and is targeted at the task of facilitating augmented reality applications in small-scale workspaces (such as a desktop). In this approach, the processes of tracking and mapping are performed in two parallel threads. Mapping is performed using bundle adjustment. Robust performance was achieved in an environment as large as a single office. While impressive, these systems are not designed for multi-session missions or for mapping of large-scale spaces (e.g., the interior of a building).

One exception to this has been the extension to PTAM developed by Castle et al. [16] to permit several cameras to work in multiple maps, both separately or simultaneously. Here the approach to dealing with large scale environments is to permit the user to decide what regions to map. Each map is bounded in size and operates independently of other maps in the system. To switch between maps the current frame is matched against a set of subsampled keyframes from all existing maps. The authors provide impressive results of the technique's operation in a building scale environment. A key difference between this approach and our work is that the system does not estimate the transformation between the submaps and therefore does not provide a global estimate of the environment.

There have also been a number of approaches reported for large-scale visual mapping. Perhaps the earliest of these was the work of Nistér et al. [111] on stereo

odometry. In the robotics literature, large-scale multi-session mapping has been the focus of several recent efforts, including Sibley *et al.* [126, 127], who used relative bundle adjustment combined with FABMAP for appearance-based loop closing, and Konolige *et al.* [69, 70, 73] who developed the FrameSLAM system for large-scale and lifelong mapping.

The research in this thesis is most closely related to the work of Konolige et al. [73], but has several differences. A crucial new aspect of our work is the method we use for joining the pose graphs from different mapping sessions. In the view-based mapping approach, Konolige et al. employ the Toro incremental optimisation algorithm to allow for real-time performance. Due to the fact that Toro requires that all poses are connected in a single graph, at the beginning of each new session the new pose-graph is immediately connected to the last pose from the previous session through what they refer to as a "weak link". The weak links are added with a very high covariance and subsequently deleted after place recognition is used to join the pose graphs[73].

In our approach, described in Chapter 4, we use anchor nodes [66] as an alternative to weak links. Here each session is represented initially as a disjoint pose-graph with each pose stored relative to that pose-graph's anchor node. When the place recognition system identifies an encounter between two separate sessions, the encounter induces a constraint between the two associated poses and the anchor nodes of the associated pose-graphs. Since the pose-graphs are each represented relative to the anchors nodes, their use provides a more efficient and consistent way to stitch together the multiple pose graphs resulting from multiple mapping sessions. In addition, our system has been applied to hybrid indoor/outdoor scenes, with hand-carried (full 6-DOF) camera motion. To our knowledge, the multi-session mapping results published by [70] are restricted to planar motion.

2.6 Summary

This chapter has reviewed the literature in SLAM, describing the three main techniques for SLAM state estimation and providing a review of some of the key approaches to visual SLAM. The capability to extract motion constraints directly from visual input is at the heart of any visual SLAM system. The next chapter, focusses on this problem, describing each of the components of a visual odometry system, including camera modelling, camera pose estimation, feature detection and tracking, robust estimation, and windowed bundle adjustment.

CHAPTER 3

Visual Odometry

The aim of this chapter is to provide an overview of the main components involved in the visual odometry processing with a particular focus on the approaches involving stereo based camera systems. One of the main contributions of this thesis is the development and evaluation of a state-of-the-art visual SLAM system. The front-end of the system consists of a set of modules for the computation of stereo based visual odometry. Hence this chapter provides the necessary background for placing the design of this aspect of the system in the context of other systems reported in the literature and, to assist the reader in understanding the basis of the design decisions taken during its development.

3.1 Introduction

The goal of any visual SLAM system is to estimate a robot's motion through an environment whilst simultaneously building a map of that environment. This is achieved

through a combination of local motion constraints between consecutive poses (or *keyframes*) and global loop closure constraints. The separation of these two types of constraints is important for two reasons: (*i*) they each employ different methods to be computed, and, (*ii*) they each play a different role in the overall computation. In particular, local constraints permit the incremental estimation of the motion and map, whilst loop closure constraints correct for accumulated drift and hence are critical in achieving global consistency.

In this chapter the process of estimating local motion constraints by computing the ego-motion of the sensor platform is discussed. In particular we address the situation where only camera sensors are used. From the point of view of the SLAM estimator the output of this process is akin to that of wheel odometry, and hence it is referred to as *visual odometry* (VO). The term itself was first coined in the seminal work by Nistér et al. [111], however the problem had been studied for many years previously by Moravec [100], Matthies [90] and others (see Scaramuzza and Fraundorfer [123] and the references therein for a comprehensive historical review of the field).

The significance of Nister's work is that it is considered the first VO system that could operate robustly in real-time over large distances (*i.e.* of the order hundreds of meters). The approach was demonstrated in both monocular and stereo platforms, where the latter was shown to be more accurate and reliable due to the determination of scale from knowledge of the baseline distance. A secondary benefit of the stereo based approach was the fact that no special handling was required in the cases of little or no motion, which in the monocular case is ambiguous with situations where all features are at a large distance from the camera. It is also worth noting that since estimation in the monocular case is only possible up to a scale factor, intermittent failures in the processing will result in reinitialisation and hence an unknown relative scale factor. Again given the known baseline this problem does not occur in the case of stereo.

In essence, from the point of view of visual odometry, the differences between monocular versus stereo cameras is that the former is a bearing-only sensor whereas the latter provides both bearing and range. For monocular sensors this raises the problem where a single measurement of a feature point does not provide sufficient information for landmark initialisation. This problem has received much attention in the monocular visual SLAM community. A consequence of the fact that the problem does not occur in stereo (at least within a limited range [133]) is that fields of stereo *vs.* mono based approaches have been investigated somewhat independently over the years.

3.2 Rigid Body Transformations

In order to simplify the notation, we will use the homogeneous representation of the rigid body (*i.e.* Euclidean) transformations between two frames,

$$\mathbf{p}_B = T_A^B \mathbf{p}_A \tag{3.1}$$

where, $\mathbf{p}_i \in \mathbb{R}^4$ are homogeneous vectors, the subscript on the vector identifies the coordinate frame in which the point is represented, and the subscript and superscript on the transformation, T_A^B, identifies that the transformation is from coordinate frame A to coordinate frame B. For any given matrix R representing a rigid body rotation $SO(3) = \{R \in \mathbb{R}^{3\times3} : R^T R = I, det(R) = 1\}$ and translation vector $\mathbf{t} \in \mathbb{R}^3$, the corresponding homogeneous transformation matrix T is given by

$$T = \begin{pmatrix} & R & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

From Eq. 3.5 on page 39 it can be seen that for a calibrated camera, visual odom-

etry seeks to recover a series of camera poses relative to some chosen world coordinate system, W, one for each image frame, I_t, taken at time t, from the given input sequence. However, rather than compute the set of *absolute* transformations, T_t^W, most systems will instead compute a series of incremental transformations either between each consecutive pair of image frames (*i.e.* T_t^{t-1}) or between the current frame and the most recent *keyframe,* k_i (*i.e.* $T_t^{k_i}$). The use of keyframes reduces the number of poses represented in the final pose graph by only adding a pose, or keyframe, when the translational or rotational motion of the sensor exceeds a predefined threshold (or in some systems when the number of features successfully tracked from previous frames drops below a predefined threshold). The approach of using keyframes is discussed in Chapter 4.

Given the above, the absolute motion of the camera in the world coordinate system can be recovered by,

$$T_t^W = T_{t_0}^W T_{t_1}^{t_0} \dots T_t^{t-1} \tag{3.3}$$

In most cases and without loss of generality we choose the world coordinate system as the coordinate system of the camera at t_0, (i.e. $T_{t_0}^W = I$) and so,

$$T_t = T_{t_1}^{t_0} \dots T_t^{t-1} \tag{3.4}$$

where the absence of the superscript on the left hand side is taken to imply that the matrix represents the absolute pose of the camera.

Many parameterisations of the set of rotations $SO(3)$ exist [74] including the explicit matrix representation (as used above), Euler angles (e.g. pitch-yaw-roll triples), rotation vectors [30] (closely related to axis-angle representation), quaternions of unit norm [55], and exponential coordinates [87]. Since the number of degrees of freedom of R_t is greater than the rotation that it represents, estimating it directly requires extra constraints to ensure that the resulting matrix is orthogonal. Instead it is typ-

ical to use one of the above more compact representations in the estimation process. Hence this leads to an important choice in any visual odometry system.

The main differences between the various parameterisations are the number of parameters used and the constraints to ensure that points within the parameterisation lie on the manifold $SO(3)$. Given that the intrinsic dimensionality of the manifold is 3 this corresponds to the minimum dimensionality of any parameterisation. Many SLAM systems have historically used Euler angles to parameterise rotation due to the fact that their parameterisation is both minimal and intuitive. However such global parameterisations suffer from singularities and as such can cause problems within minimisation procedures [141]. Instead a more appropriate approach is to perform the optimisation by computing updates in the local tangent space of $SO(3)$ at the current estimate and then to map the resulting update back into the manifold. In this thesis we develop such an approach utilising quaternions, detailed in Section 4.2.3, thereby avoiding the problems with singularities.

3.3 Camera Models

Camera estimation refers to the process of estimating the projection effected by a camera transforming scene points, represented in some world frame, to image points, represented in the image retinal plane. Throughout this thesis this transformation is modelled using the standard perspective projection model,

$$\mathbf{x} = KT_W^C\mathbf{p} = K\left(\begin{array}{c|c} R & \mathbf{t} \end{array} \right)\mathbf{p} \tag{3.5}$$

Here, $\mathbf{p} = (x, y, z, 1)^T \in \mathbb{R}^4$, is a homogeneous 3D world point, and $\mathbf{x} = (u, v, w)^T \in \mathbb{R}^3$ is the homogeneous coordinate corresponding to the inhomogeneous 2D image point, $\tilde{\mathbf{x}} = (u/w, v/w)^T \in \mathbb{R}^2$. $R \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$ are the rotation matrix and inhomogeneous translation vector defining the transformation between the world and

camera coordinate system. The matrix, $K \in \mathbb{R}^{3\times3}$, is the camera projection matrix,

$$K = \begin{pmatrix} f_x & 0 & u_c \\ 0 & f_y & v_c \\ 0 & 0 & 1 \end{pmatrix} \tag{3.6}$$

where f_x and f_y are the focal length in the x and y directions, and $\tilde{\mathbf{p}}_c = (u_c, v_c)^T$ is a vector of the inhomogeneous coordinates of the principal point of the camera *i.e.* point intersection between the principal ray of the camera and the image plane. It should be noted that real cameras suffer from radial and tangential distortions which are not modelled in the above expression. Instead it is assumed that standard preprocessing steps have been applied to calibrate and remove these distortions [54].

As can be see from Eq. 3.5 the projection of a camera is defined by a 3×4 projective matrix,

$$P = K \left(\begin{array}{c|c} R & \mathbf{t} \end{array} \right). \tag{3.7}$$

To invert this projection we may use the (right) pseudo-inverse $P^\dagger = P^T \left(PP^T\right)^{-1}$. In the case when the world and camera coordinate systems are the same (*i.e.* $R = I$ and $\mathbf{t} = (0,0,0)^T$),

$$P^\dagger = \begin{pmatrix} K^{-1} \\ 0 \quad 0 \quad 0 \end{pmatrix} \tag{3.8}$$

and $P^\dagger \mathbf{x} = \begin{pmatrix} K^{-1}\mathbf{x} \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{x}' \\ 0 \end{pmatrix}$. Here $\mathbf{x}'$ is referred to as the normalised image coordinates which can be thought of as the projection of the original scene point by a camera where $K = I$. More importantly however we see that multiplication of the original image point by the camera inverse results in a homogeneous vector $\mathbf{D} = \begin{pmatrix} \mathbf{x}' \\ 0 \end{pmatrix}$ with 0 as its fourth coordinate. Such vectors represent points at infinity and correspond to the direction of the back projected ray through the camera

centre. Hence given a single *calibrated* camera viewing a scene, we may consider it geometrically as a *bearing only* sensor.

3.3.1 Stereo Imaging and 3D Reconstruction

In order to recover both bearing and range of features within an image therefore requires two or more images of the corresponding scene points taken from different perspectives. Given two such views the geometric relationship between the images of the scene is defined by the epipolar geometry (see Figs. 9.1 and 9.2 of [54]). The importance of the epipolar geometry is that it provides a constraint between the two views such that points in one view are constrained to lie on the corresponding epipolar line in the other view. This means that when we detect a feature point in one view for which we wish to find the correspondence in the other view, the number of points searched is reduced from $O(n^2)$ to $O(n)$.

The most straightforward stereo camera configuration is when the pair of cameras are *frontal-parallel*, that is, where the transformation between the two sensors is restricted to a translation in the horizontal direction by a distance, b, known as the baseline distance. Here the epipoles (*i.e.* the projection of the left and right camera centres in the alternative camera view) are at infinity, and as a consequence the epipolar lines become horizontal. In this situation the left- and right-imaging systems are governed by the following projection equations,

$$\mathbf{x}_l = K \left(\begin{array}{c|c} & 0 \\ I & 0 \\ & 0 \end{array} \right) \mathbf{X} \qquad\qquad \mathbf{x}_r = K \left(\begin{array}{c|c} & b \\ I & 0 \\ & 0 \end{array} \right) \mathbf{X} \tag{3.9}$$

where it can be seen that in normalised image coordinates the $\mathbf{x}_l = \lambda(x, y, 1)^T$and $\mathbf{x}_r = \lambda(x - d, y, 1)^T$*i.e.* the points only differ by a shift in the horizontal direction

known as the disparity.

Recovery of the homogeneous coordinates of a 3D scene point from a pair of corresponding features can be achieved using the following reprojection matrix ([10, pp. 435]),

$$Q = \begin{pmatrix} 1 & 0 & 0 & -u_c \\ 0 & 1 & 0 & -v_c \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/b & 0 \end{pmatrix} \tag{3.10}$$

as follows,

$$\mathbf{p}_C = Q \begin{pmatrix} x \\ y \\ d \\ 1 \end{pmatrix}. \tag{3.11}$$

3.4 Camera Pose Estimation

Camera pose estimation is the problem of determining the position and orientation of a camera based on a known scene or object. Solutions to this problem have a wide variety of applications in areas including augmented reality, model-based tracking, visual servoing, photogrammetry, and of course, visual odometry.

In the context of visual odometry the majority of approaches are based around detecting and matching, or, detecting and tracking features across frames. Although we may estimate camera motion purely based on 2D-to-2D correspondences (*e.g.* via the essential matrix), it is more common to use 3D-to-3D or 3D-to-2D correspondences. In the monocular case this implies that at least 3 images are required, however with stereo odometry 3D features may be directly computed via the disparity between the image pair and hence only a set of feature correspondences from a pair of views is required.

Solving each of the above scenarios requires very different approaches. The problem of estimating the pose of a *calibrated* camera given n 2D image points corresponding to n known 3D world points is a specific class of the pose estimation problem, known as the *Perspective-n-Point* (or PnP) problem. Given the significance of the PnP problem in both computer vision and photogrammetry, numerous solutions have been reported in the literature. These solutions can be categorised based on the value of n *i.e.* the number of correspondences that they require. The minimum number of correspondences in order for the problem to be well-posed is $n = 3$.

Solutions to the PnP problem can be divided into direct or iterative approaches. Iterative solutions have been shown to be the most accurate [83] but are slower than direct methods [79] and only applicable when a good initial value is available. In photogrammetry this is typically provided manually at a level of accuracy that is sufficient for iteration to the correct solution. In computer vision however this is not the case and so some automatic initialisation is required.

Direct approaches as their name suggests provide closed form solutions to the problem and hence are typically more computationally efficient than iterative approaches. The first direct solution in the computer vision community is due to Fischler and Bolles [42], however Haralick et al. [52] identify solutions from the field of photogrammetry dating back as far as Grunert [48].

A common geometry for the direct solution to the P3P problem is shown in Fig. 3.1(a). Here we see the tetrahedron formed by the perspective projection of three known world points, $\{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\}$, the camera centre, $\mathbf{C}$, and the rays joining each of the three world points to the camera centre. In this case both the back-projected rays, $\{\overrightarrow{\mathbf{Cx}_1}, \overrightarrow{\mathbf{Cx}_2}, \overrightarrow{\mathbf{Cx}_3}\}$, of each of the image points and the distances between the world points, $\{R_{12}, R_{23}, R_{13}\}$ are known. The problem is now to first determine the distances $|\mathbf{CX}_i|$, and therefore the coordinates of the three world points in the camera's coordinate system. Using the 3D coordinates of the three points in both coordinate

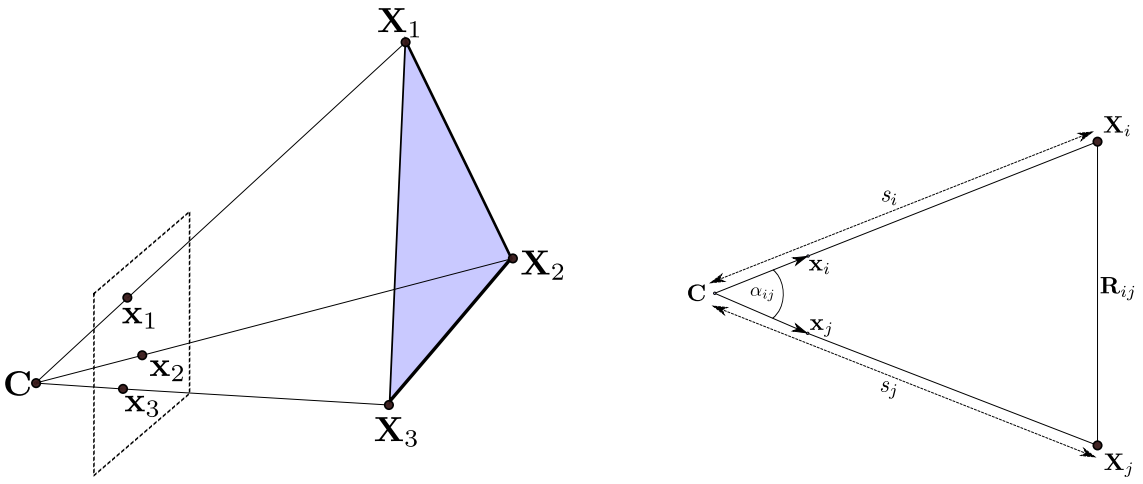


Figure 3.1: Geometry of the P3P problem.

systems, computing the pose of the camera can be found by solving for the absolute orientation [55].

Haralick et al. [52] compare six direct P3P solutions all of which start by taking each possible pair of world points, $\mathbf{X}_i$ and $\mathbf{X}_j$. Combining the chosen points with the camera centre, $\mathbf{C}$, results in the triangle shown in Fig. 3.1b, where by the cosine rule,

$$s_i^2 + s_j^2 - 2s_i s_j \cos(\alpha_{ij}) = R_{ij}^2 \tag{3.12}$$

Each combination of points yields such a constraint between the corresponding side-lengths s_i and s_j. Next a change of variables is applied which results in two of the s_i's being cast as scalar multiples, u and v, of the remaining value, s_j. For example Fischler and Bolles [42] use

$$s_2 = us_1 \qquad \text{and} \qquad s_3 = vs_1 \tag{3.13}$$

The approach is then to first solve for the multiplicative factors, u and v, which allows the value of s_j to be recovered, which in turn allow the s_i's to be computed. The principal differences between the approaches is the order in which they combine the

tetrahedron side lengths and the techniques used to solve for u and v.

One of the most commonly used solutions due to Fischler and Bolles [42] starts with the change of variable shown in Eq. 3.13 and successively eliminates the variables to arrive at a quartic equation in u,

$$\sum_{i=0}^{4} a_i u^i = 0 \tag{3.14}$$

where, each of the a_i's are functions of the knowns listed above. In general, Eq. 3.14 has a maximum of four different solutions. In practice to overcome this ambiguity four points are used in combinations of triples, where each triple generates up to four solutions. The correct solution is identified as the one that is common to each solution set.

Exact solutions to the PnP problem, such as those outlined above, only exist for cases involving a maximum of four point correspondences. Using more than this minimal number of correspondences results in solutions that are more robust to noise due to the fact that the systems are overdetermined. In these situations the most obvious approach is to employ nonlinear optimisation techniques such as Gauss-Newton (GN) or Levenberg-Marquardt (LM) algorithms to minimise the reprojection error (*i.e.* the error between the projected world point and the measured image point). However their performance is very much dependent on suitable initialisation.

Lu et al. [83] present the *Orthogonal Iteration* (*OI*) algorithm based on object-space collinearity equations which measure the collinearity between the camera centre, image point, and object point (*i.e.* C, x_i, and X_i) for each pair of correspondences. For a measured image point (in normalised image coordinates), $\mathbf{v}_i = (u, v, 1)^T$, and a corresponding inhomogeneous world point, $\mathbf{p}_i$, the object space collinearity error is given by,

$$\mathbf{e}_i = (I - V_i)(R\mathbf{p}_i + \mathbf{t}) \tag{3.15}$$

where, R and $\mathbf{t}$ are the estimated camera rotation matrix and translation vector, and

$$V_i = \frac{\mathbf{v}_i \mathbf{v}_i^T}{\mathbf{v}_i^T \mathbf{v}_i} \tag{3.16}$$

is the *line-of-sight projection matrix*. Taking the sum or the square of these error terms the OI algorithm alternates between minimising for the translation vector and rotation matrix. Importantly they prove that the algorithm is globally convergent and show empirically that the true pose is returned for a wide range of initial guesses. As demonstrated by [79] the OI algorithm is one of the most accurate algorithms available and is fast when compared with other iterative methods.

Moreno-Noguer et al. [101], Lepetit et al. [79] provide an non-iterative approach to the PnP problem for $n \geq 4$, known as the *efficient PnP* (*ePnP*) algorithm that, rather than directly estimate the depths of individual world points, recovers a set of four virtual control points. These virtual control points provide a coordinate system in which all other points are represented as weighted combinations of the control points. More specifically, if the control points are denoted as $\mathbf{c}_j$, $j = 1 \ldots 4$, then any point $\mathbf{p}_i$ may be represented through the weighted combination,

$$\mathbf{p}_i = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j \tag{3.17}$$

Since the α_{ij}'s are invariant to Euclidean transformations the same set of weights will represent the points in either the world or camera coordinate system. Hence the ePnP algorithm, given the control points, $\mathbf{c}_j^W$, represented in the world coordinate system, aims to recover their representation in the camera coordinate system, $\mathbf{c}_j^C$. A major advantage of their technique is that it is an $O(n)$ non-iterative solution, and also that it can be applied with values of $n \geq 4$.

3.5 Robust Model Fitting

One of the central issues in applying pose estimation algorithms such as P3P solutions in real-world scenarios is dealing with noise in the measurements. Given the resolution of the imaging sensor, the number of features output by a feature detector will be at least of the order of hundreds and therefore results in a highly overdetermined problem.

In the standard non-linear least squares (NLSQ) formulation [88] the cost function to be minimised is given by,

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \{F(\mathbf{x})\} \tag{3.18}$$

where,

$$F(\mathbf{x}) = \frac{1}{2}\sum_i (f_i(\mathbf{x}))^2 \tag{3.19}$$

Such a cost function provides the maximum likelihood (ML) estimate of the parameters under the assumption of Gaussian noise.

In the case of camera pose estimation f_i measures the *reprojection error* of individual points such that given a set of correspondences, $\mathbf{p}_i \leftrightarrow \mathbf{P}_i$,

$$f_i(\mathbf{x}) = \|\Pi(\mathbf{x}, \mathbf{P}_i) - \mathbf{p}_i\|_{\Sigma_i} \tag{3.20}$$

where, $\Pi(\mathbf{x}, \mathbf{P})$ is the projection of a 3D point $\mathbf{P}$ into a camera of given 3D pose $\mathbf{x}$, and Σ_i represents the covariance on the measurement.

Although the use of a least-squares approach aims to compensate for Gaussian perturbations in measurements it does not cater for *outliers* in the data. Such measurements deviate arbitrarily from the correct model and are not modelled via the standard normal distribution assumed in the least-squares paradigm. In particular, given that in Eq. 3.19 $F(\mathbf{x})$ increases quadratically as a function of the deviation

of the data from the model, outliers will have an unduly large influence on the cost function and the resulting estimate.

Common sources of outliers in visual odometry are objects moving within the scene (e.g. pedestrians) or incorrect matching of features between frames due to failure of the feature tracking algorithm or the disparity estimation process. In the former case we can view the data as consisting of multiple separate populations each satisfying its own motion model. On the other hand, in the latter case we have a population of inliers drawn from a single model and a separate population of outliers which do not agree with any underlying model.

Robust estimation techniques aim to overcome this problem by fitting models to data under the assumption that the data contains outliers. The assumption is that data can be segmented into one or multiple inlier sets each of which can then be modelled independently. The two main approaches to robust estimation used in the computer vision literature are hypothesise-and-test methods and M-estimators [134].

3.5.1 Hypothesise-and-test methods

Perhaps the most common approach to dealing with outliers is the *random sampling consensus* method, or RANSAC, due to Fischler and Bolles [42]. Where previous methods to model fitting started by fitting the model to all the data, with the intention of smoothing out the noise, RANSAC takes the opposite approach. In the first step of RANSAC, known as the *random sampling step*, a model is estimated by fitting to a randomly sampled *minimal* set of data points. The second step, known as the *consensus step*, evaluates the resulting model by checking its consistency with the remainder of the data. The result of this second step is to segment the data into inlier and outlier sets relative to the estimated model. The decision as to whether to accept the model is based on the number or percentage of data points in the inlier set. These two steps are iterated until a set of points are sampled that results in a model

Algorithm 3.1: Random Sampling Consensus (RANSAC)

Input: $\mathbf{x} = \{\mathbf{x}_0 \dots \mathbf{x}_n\}$ set of sampled data points
τ inlier residual threshold
C consensus threshold
N maximum iteration threshold
Output: ζ model parameters
do
  $k \leftarrow 0$
  $n_{max} \leftarrow 0$
  repeat
    $\mathbf{x}_s \leftarrow$ choose_minimal_sample($\mathbf{x}$)
    $\zeta_s \leftarrow$ fit_model($\mathbf{x}_s$)
    $n \leftarrow 0$
    $\mathbf{x}' \leftarrow \emptyset$
    foreach $\mathbf{x}_i$ **do**
      if $\|f_{\zeta_s}(\mathbf{x}_i)\| < \tau$ **then**
        $\mathbf{x}'_n \leftarrow \mathbf{x}_i$
        $n \leftarrow n + 1$
    if $n > n_{max}$ **then**
      $\mathbf{x}_{max} \leftarrow \mathbf{x}'$
      $\zeta_{max} \leftarrow \zeta_s$
      $n_{max} \leftarrow n$
  until $n < C$ *and* $k \leq N$
  $\zeta \leftarrow$ refine_model($\mathbf{x}_{max}, \zeta_{max}$)
end

with a high consensus, or until a maximum number of iterations are reached. Given the resulting inlier set the model is refined through the use of a standard model fitting approach (e.g. NLSQ-estimation). Algorithm 3.1 provides a pseudo-code listing of the standard RANSAC algorithm.

When applying RANSAC the user must choose values for τ, C, and N. Fischler and Bolles [42] provide a discussion on each of these parameters. For choosing, τ, the tolerance on deviation from the model for deciding between inliers and outliers, they suggest determining via an empirical procedure. To choose C requires that the value is high enough to both ensure that the algorithm does not detect the incorrect subset of the data and that the final model fitting step is sufficiently over-determined. The number of iterations, N, required for a given success rate can be determined by the

expression

$$N = \frac{\log(1-p)}{\log\left(1-(1-\varepsilon)^{s}\right)} \tag{3.21}$$

where, p is the desired probability of success, ε is the percentage of outliers in the data, and s is the number of points required to compute the model hypothesis.

Since its original publication RANSAC has been used widely throughout the field of computer vision and has received considerable further investigation with many variants being reported in the literature. Choi et al. [17] provide a comprehensive comparative study of a large number of these algorithms, categorising approaches along seven different attributes. For example one family of approaches use *guided sampling* where rather than sampling from the data set uniformly, *a-priori* knowledge or heuristics are used to guide the search, thereby reducing the number of iterations required. Another family of approaches attempt to reduce computation time by terminating the consensus evaluation early through the use of either preliminary test or by monitoring the evaluation online. For example Preemptive RANSAC due to Nistér [109] permits model fit under time constraints by considering multiple hypothesis in parallel with a preemption scheme for terminating low scoring hypothesis early in the consensus evaluation.

The approach used by RANSAC to choose between different hypothesis can be considered as comparing a total cost based on a sum of responses of a cost function for each of the samples in the dataset. The chosen hypothesis is then the one with the lowest cost. For the original RANSAC algorithm this cost function simply attributes a cost of zero to inliers and one to outliers *i.e.* the total cost is equal to the total number of outliers. A number of approaches attempt to enhance the performance of RANSAC through the use of alternatives to this binary cost function. For example Torr and Zisserman [145] presented two such techniques. Firstly MSAC (M-estimator sample and consensus) used the square of the residual as cost for inliers (i.e. residuals less than τ) and a constant cost of τ^2 for outliers. MLESAC (maximum

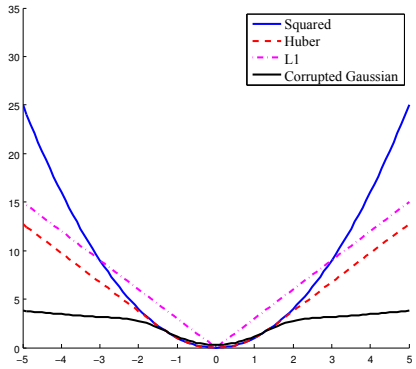


Figure 3.2: Comparison of robust cost functions.

likelihood consensus) builds on this idea by treating the data as being drawn from a mixture of two Gaussians and then uses expectation maximisation (EM) to compute the segmentation into inliers and outliers with the highest likelihood.

3.5.2 Robust Cost Functions

M-estimators replace the $f_i(\mathbf{x})$ in Eq. 3.19 with a robust cost function, $\rho_i(\mathbf{x})$, that increases sub-quadratically as a function of the deviation of the data from the model. This is based on the idea that outliers are drawn from much wider tailed distributions.

Many robust cost functions are available from the literature. For a comprehensive list and comparison the reader is referred to Hartley and Zisserman [54, Appendix A6.8]. A subset of those covered in [54] are shown in Fig. 3.2, where the squared cost is the non-robust cost of Eq. 3.19.

The Corrupted Gaussian makes the assumption that the data is drawn from a mixture of two Gaussians; one that accounts for the inliers, and a second with a much larger standard deviation. This leads to a cost function of the form,

$$C(\delta) = -log(\alpha \exp(-\delta^2)) + (1-\alpha)\exp(-\delta^2/w^2)/w \tag{3.22}$$

where, α is the ratio inliers to outliers, and w is the ratio of the standard deviations of the outlier and inlier distributions. The advantages of this cost function is that it

is based on a PDF, however it results in a non-convex cost function.

The $L1$ cost function is based on a scaled $L1$-norm and is given by

$$C(\delta) = 2b|\delta| \tag{3.23}$$

where b is a positive scale factor. This cost gives less weight to outliers than the square cost, with the interesting property that the minimum corresponds to the median of the data. The principal disadvantage of this cost function is that it is non-differentiable at the origin.

The Huber cost function evaluates to the squared cost function for small residuals (i.e. inlier) but evaluates to the $L1$-type cost when the residual goes above a threshold. It is given by

$$C(\delta) = \begin{cases} \delta^2 & \text{if } d < b^2 \\ 2b|\delta| - b^2 & otherwise \end{cases} \tag{3.24}$$

The advantage of the Huber cost function is that it is C^1 differentiable. Both the $L1$ and Huber costs are convex and therefore does not have any local minima.

Incorporating a robust cost function into non-linear least squares optimisation turns out to be quite straight forward once the cost function is defined. In particular it corresponds to computing a standard non-linear least squares optimisation but where the residuals are weighted by an attenuation function

$$w = \sqrt{\frac{C(\|\delta\|)}{\|\delta\|^2}} \tag{3.25}$$

From the above it can be seen that the attenuation function attenuates the influence of points whose robust cost is lower than the standard squared cost function.

3.6 Feature Detection and Tracking

Given the above discussion, the motion estimation process of VO requires an ability to identify correspondences across frames in the input image sequence. Although methods do exist for dense correspondence estimation, referred to as optical flow, these approaches require significant computational resources and typically only provide real-time processing in constrained circumstances. With this said, recent approaches have begun to address this issue through the use of general-purpose computing on graphics processing units (GPGPU) implementations [150].

Due to these computational requirements considerable effort has been focussed on the development of methods for computing correspondences based on salient features, where saliency is evaluated over a local region of the image. These methods typically divide the task into three steps. Firstly, the feature detection step identifies locations that have a characteristic structure amenable to matching across views. Next, the feature description step computes a feature vector that describes the local appearance at each of the feature locations. Finally the feature matching step uses the resulting features vectors to match features across views. Often this last step is replaced by a feature tracking step which, rather than independently detect features in the second image, instead performs a local search in the region of the features detected in the previous frame for points where the local image appearance matches. This approach is particularly suitable in areas such as VO where high frame rates result in neighbouring frames having similar view points.

3.6.1 Feature Detection

One of the earliest feature detectors was due to Moravec [100], as part of one of the first efforts in visual navigation of a wheeled robotic platform. Here he defines a feature as *"conceptually a point in the three-dimensional world ... that can be located*

unambiguously in different views of a scene". The requirements on such points are that they be inhomogeneous in appearance and that they contain regions of high contrast (*e.g.* textured regions, or regions with edge features in multiple directions). Moravec hypothesised that such regions were characterised as having low self-similarity with neighbouring regions and proposed a detector based around this principle. To measure self similarity the detector computed the sum-of-squared (SSD) differences between a neighbourhood of pixels around the current pixel and the immediately adjacent neighbourhoods in the horizontal, vertical and diagonal directions. The final feature score was chosen as the minimum SSD along each of the four directions. Local maxima of this measure were identified as *distinctive* and hence classified as features.

A weakness of the Moravec detector is that it is directionally biased due to the four directions along which the comparisons are performed. Harris and Stephens [53] improved on Moravec's approach through the introduction of the structure tensor (or the second moment matrix) which is related to the auto-correlation function. In particular they formulate the Moravec operator as detecting features that locally maximise the minimum value of the following expression over a local neighbourhood of x and y.

$$E(x,y) = \sum_{u,v} w(u,v) \left| I(x+u, y+v) - I(u,v) \right|^2 \tag{3.26}$$

In the Moravec detector $w(u,v)$ is a binary valued function having value one in the rectangular region of the window and zero elsewhere. Harris and Stephens approximate Eq. 3.26 through a Taylor series expansion as,

$$E(x,y) = \begin{bmatrix} x & y \end{bmatrix}^T \begin{bmatrix} \langle I_x^2 \otimes w(u,v) \rangle & \langle I_x I_y \otimes w(u,v) \rangle \\ \langle I_x I_y \otimes w(u,v) \rangle & \langle I_y^2 \otimes w(u,v) \rangle \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{x}^T A \mathbf{x} \tag{3.27}$$

where, I_x and I_y are the first derivatives of the image in the x and y directions, respectively, and $\langle \cdot \rangle$ denotes averaging over the spatial window. Here the terms of

the central matrix, A, are averages over the pixel neighbourhood weighted by the window function $w(u, v)$. By choosing a circularly symmetric window function such as a Gaussian the response of the Harris filter becomes isotropic.

The key to the Harris detector is that corners in the image correspond to locations where both eigenvalues of this matrix are large. Given the relationship of this matrix to the second moment matrix, this corresponds to locations where the gradient response is high in two separate directions. The basis of the Harris detector is that such locations can be repeatedly and accurately localised under varying viewpoints. To avoid the complexity of explicitly calculating both eigenvalues Harris and Stephens suggest thresholding the following measure,

$$M_c = \lambda_1\lambda_2 - \kappa\left(\lambda_1 + \lambda_2\right)^2 = \det\left(A\right) - \kappa\left(\text{trace}(A)\right)^2 \tag{3.28}$$

Shi and Tomasi [124] start by addressing the feature tracking problem by developing a six degree-of-freedom affine model of image motion. This model provides a more accurate means of comparing features over larger displacements than the simple translational model offered by previous approaches. However over small displacements estimation of the full 6 parameters of this more general model is unstable, and therefore they use the displacement model for frame to frame tracking. This reduces to solving a linear system involving the matrix A from Eq. 3.27 [7].

Given that the purpose of detecting features is to allow them to be tracked, they define good features as those that are amenable to tracking and hence A must be well conditioned (*i.e.* both eigenvalues should be large and should not vary by several orders of magnitude). Given that both eigenvalues are bounded due to the finite range of permitted gray values in an image, they use a criteria whereby the minimum eigenvalue is greater than some threshold λ,

$$\min(\lambda_1, \lambda_2) > \lambda \tag{3.29}$$

Förstner and Gülch [44] also make use of the second moment matrix for feature detection, however they introduce a second step to localise the feature within the detection window, which they point out is not in general located at the window centre. This second step is achieved by finding the point in the window with the minimum perpendicular distance to all of the lines in the neighbourhood.

Smith and Brady [132] defined the Smallest Univalue Segment Assimilating Nucleus (SUSAN) framework for low-level image processing. Here a circular window is centred on the pixel of interest with a measure of similarity computed between the gray value of the centre pixel and all other pixels within the window. Thresholding the resulting measure at different levels allows the identification of edges or corners. Furthermore thresholding on the similarity allows control over resilience to noise *vs.* false detection and rejection rates.

Rosten et al. [119, 120] take a similar approach to SUSAN by analysing the similarity between a pixel of interest's value and those of a surrounding *Bresenham* circle of 16 pixels. Their detector, known as FAST (Features from Accelerated Segment Test), defines a feature as any pixel with more than n pixels in this circle that are all brighter or all darker by a threshold, t. In order to accelerate the test, the FAST algorithm employs a machine learning approach that derives a decision tree that can be implemented as a series of *if-then-else* statements, which in turn are compiled and tuned using reordering optimisations. To train the detector a set of images from the training domain are collected and processed using a brute force algorithm. The resulting features are then used as input to the ID3 algorithm which uses an entropy based measure to order the tests of each location on the Bresenham circle. Finally, since the segment test does not provide a response function non-maxima suppression is achieved through the use of the sum-of-absolute-differences (SAD) between the

pixels in the contiguous arc and the centre pixel.

Scale Invariance

In order for a feature detector to be effective it must be robust to variation in the underlying image data due to changes in the extrinsic and intrinsic parameters of the camera. Such variations result in a hierarchy of transformations in the image plane known as the Klein hierarchy [54]. In particular we can consider feature detectors in terms of their robustness to (*i*) changes in scale and rotation, (*ii*) affine transformations, and (*iii*) projective transformations. The Klein hierarchy shows how each of these is a sub-group of the next. Hence invariance to projective transformations implies invariance to affine transformations, which in turn implies invariance to changes in scale and rotation. It is common to model the scene as locally planar and as a consequence if we only consider local image operators it is sufficient to only cater for variations up to the affine group (*i.e.* (*i*) & (*ii*) above).

In his now seminal work [81], Lindeberg argued that in understanding and describing the appearance of the world a multi-scale representation is crucial. Over the years a number of multi-scale representations (*e.g.* pyramids, wavelets, *etc.*) have played a key role in the development of scale invariant feature detection and matching. Multi-scale pyramids take each input image which they recursively blur and sub-sample resulting in a representation where different levels of detail become apparent at different levels of the pyramid. Hence applying a fixed sized operator (*e.g.* a 11×11 filter) at each level corresponds to detecting features at each scale represented in the pyramid.

The most common type of pyramid is the *Laplacian pyramid* due to Burt and Adelson [12]. To generate an *octave* pyramid the original image is blurred and then subsampled by a factor of 2. Repeating this process generates what is known as a *Gaussian pyramid*. The final Laplacian pyramid is then created by taking each layer

of the pyramid, G_{i-1}, from the second layer up, upsampling it to the resolution of the layer below, G_i, and computing the Laplacian image, $L_i = G_i - \uparrow G_{i-1}$, where $\uparrow G$ denotes the upsampled version of G. Note that the image at the top of the pyramid corresponds to a low resolution version of the original image. If we think of the G_i's as low-pass filtered images then the L_i's are band-passed images, hence each level captures the features at a specific spatial frequency range in the original image. Since any G_i can be reconstructed as $G_i = L_i + \uparrow G_{i-1}$ recovery of the original image from the Laplacian pyramid can be achieved by recursively applying this operation starting at the top of the pyramid.

Both of these pyramids have been used in different ways in addressing the problem of scale invariant feature detection and matching. For example Brown et al. [11] detect Harris features at every level of an image's Gaussian pyramid. Here the matching is applied across feature points detected at the same levels (i.e. scales). So although it does attempt to characterise the image in a scale invariant manner it is only effective in situations where both images have a small variation in the imaging parameters.

Lindeberg [81] argues that the correct approach is for the detector to simultaneously detect interest points that are localised both spatially and in scale by localising extrema in the scale space of the image. He uses a principled approach to show that in order to achieve true scale invariance that such extrema are found as zero crossings of the scale normalised Laplacian of Gaussian (LoG). Lowe [82] proposed the use of the Difference of Gaussian (DoG) as an alternative which can be computed more efficiently. To avoid weak features a post-processing step is applied where only features with high contrast and edge responses in multiple directions are selected. The resulting framework for detecting and matching features across wide baselines is known as the Scale Invariant Feature Transform (SIFT) and has had a dramatic impact on the field of computer vision since its publication (with almost 20,000 citations[1]

[1]Source: Google Scholar (26^{th} July 2013)

since 2004.). Mikolajczyk and Schmid [94] define the Harris-Laplace operator that combines the scale selection provided by the Laplacian with the reliable localisation provided by the Harris detector. They also investigate the use of the determinant of the Hessian matrix, which detects blob-like structures, as an interest point detector. Specifically they detect these interest points at all scales (similar to Brown et al. [11]) but only keep those corners that occur at extrema in the Laplacian.

Bay et al. [5] proposed the Speeded Up Robust Feature (SURF) transform to detect points in both location and scale where the determinant of Hessian matrix is maximum. To speed up the computation the Hessian is approximated through the use of box filters that can be computed efficiently through the use of the integral image. The resulting maxima are reduced and refined through the use of non-maxima suppression and scale-space interpolation, respectively. SURF features possess many of the strengths of SIFT-type features with the advantage that they are significantly cheaper to compute. Given their robustness in matching across wide-baselines they have been used extensively in many visual SLAM systems, including the system described in Chapter 4, for detecting and handling loop closures.

Agrawal et al. [1] identify a drawback of scale space approaches that detect extrema using an image pyramid in that the resulting features can suffer from poor localisation, which can have a significant impact on visual odometry estimation.The reason for this is that features at a larger scale are detected *higher up* in the pyramid and therefore at a lower resolution. To overcome this problem they define the Centre Surround Extrema (CenSurE) detector which attempts to efficiently compute features of all scales at the full input image resolution.They maintain efficiency at all scales the approach uses a set of integral images in conjunction with either a difference of boxes or octagonal approximation of the Laplacian.

Affine Invariance

Although the above detectors provide a greater level of robustness to viewpoint change than the earlier detectors of Moravec, Harris, *etc.* they still only model a sub-group of the geometric transformations induced on an image patch. In a VO setting where we employ a tracking approach (i.e. matching features between consecutive frames) this level of robustness is typically adequate. However in situations where keyframes are employed and therefore we wish to match features over larger changes in viewpoint, we then require detectors (and descriptors) that are invariant to affine transformations.

Mikolajczyk et al. [96] provide a comparative evaluation of a number of popular such detectors. For example, in [97] they extend their earlier work on Harris-Laplace and Hessian-Laplace detectors to affine transformations by using the eigenvalues of the second moment matrix to measure the shape of the affine region for a given interest point. Here an iterative procedure is applied where each iteration warps the region until both eigenvalues become equal. The resulting region can be considered affine *covariant* and should result in the same image patch across viewpoints.

3.6.2 Feature Description, Matching, and Tracking

As mentioned in the introduction to this section, the purpose of detecting features in frames is that it provides the raw material for computing correspondences which in turn provide constraints for the visual SLAM system. This involves either detecting features in both images and then *matching* them, or, detecting features in one of the images and then *tracking* in the next frame. In both cases we require some local model of the appearance of the image around each feature. The key difference here is that the matching of independent features in two images is potentially more computationally intensive than tracking which consists of a local search. The advantage of the matching approach is that it tends to be more robust to large inter-frame motions and can exhibit less drift over long feature tracks. In this section we first address common

approaches to feature tracking and then deal with the issue of feature matching.

A common approach to the tracking problem is to first define a motion model in the image space. For example if we assume a translational image motion model this leads to the following sum of squares (SSD) cost function for a region [86],

$$C(\mathbf{d}) = \sum_{\mathbf{x}} \left(I_1(\mathbf{x}+\mathbf{d}) - I_0(\mathbf{x})\right)^2 \tag{3.30}$$

where the summation is evaluated over the local neighbourhood of the feature point of interest. Lucas and Kanade [86] solve for $\mathbf{d}$ by first approximating $I_1(\mathbf{x}+\mathbf{d})$ by a first-order Taylor expansion around $\mathbf{x}$. Substituting this into C and then differentiating and setting it equal to zero results in an expression which can be used to solve for $\mathbf{d}$. For each point of interest in the image this is achieved by first computing the spatial and temporal derivatives in the neighbourhood of the point, I_x, I_y and I_t, respectively. These are then used to construct the following linear least squares system in $\mathbf{d}$

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \mathbf{d} = \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} \tag{3.31}$$

Shi and Tomasi [124] extended the work of Lucas and Kanade through the use of an affine motion model which is more robust to the warping induced on image patches over long feature tracks. In their algorithm they track from frame to frame using the standard translation model but also use the full affine model to evaluate the change in the neighbourhood between the current frame and the initial frame in which the feature was detected. When this latter change exceeds a threshold the feature is discarded. This algorithm has become known as the Kanade-Lucas-Tomasi (KLT) tracker which, when combined with a pyramidal approach for robustness to large frame-to-frame displacements, has been used in many visual odometry and camera tracking applications. A GPU based KLT tracker due to Zach et al. [153] is used in

the visual SLAM system developed in this thesis as described in Chapter 4.

3.6.3 Feature Matching

The alternative to tracking is to match independently detected features between pairs of frames. Feature matching requires (i) a local description of the image in the region of the detected feature, (ii) a matching metric to evaluate the similarity of two features, and (iii) a strategy for computing a set of consistent matches between the feature sets of a pair of frames.

For example, Nistér et al. [111] use a simple 11×11 pixel window of grayscale values centred at the feature as a descriptor which they match using normalised cross correlation (NCC). They employ a matching strategy where features are first matched only to features in the other image within a region 10% of the resolution of the image. Final they apply a *mutual consistency* filter where the matching procedure is first carried out from both the first image to the second image, and then carried out in reverse. This leads to each feature in the first image having a preferred match in second image, and vice-versa. Only those matches that agree in both directions are accepted as correspondences.

The disadvantage of using correlation windows such as the above is that they are not robust to distortions induced by image motions. A large number of researchers have addressed this problem through the development of feature descriptors that posses varying levels of invariance to such distortions. Perhaps the most famous of these is the SIFT feature descriptor developed by Lowe [82].

SIFT computes its descriptor by first computing an intrinsic scale and orientation independently for each feature. The scale is determined during the feature detection step by the level of the extrema position in the image scale-space (see Section 3.6.1). This level is determined at a sub-pixel accuracy. Next a histogram of gradient directions is computed with the peak. Finally a feature descriptor is computed over a

64×64 pixel window at the computed position, scale, and orientation. The descriptor consists of a set of histograms of gradient orientations over 8 directions, with a separate histogram computed for 16 separate 4×4 pixel sub-windows. This results in a $16 \times 8 = 128$ dimensional vector known as the SIFT-descriptor.

Although a simple Euclidean metric can be applied for matching SIFT vectors, Lowe [82] suggests the use of a *ratio test* which measures the distinctiveness of a match. This works by computing the ratio of the best match for every feature to the next best match. Empirically they show that only accepting matches where the ratio is less than 0.8 eliminates 90% of false matches and whilst maintaining 95% of true matches. Mikolajczyk et al. [96], Mikolajczyk and Schmid [95] compare SIFT with a large number of local region descriptors, and found that SIFT was superior to alternatives in a majority of cases.

The SURF approach [5] defines a feature descriptor similar to, and inspired by, the SIFT descriptor but based on the response of Haar wavelets in the x and y directions. The benefit of the use of Haar wavelets is that they can be computed efficiently through the use of the integral image used in the SURF feature detection step. The resulting 64-dimensional descriptor is more compact than the SIFT descriptor and as a consequence can be more efficient in the matching step. Furthermore, Bay et al. [5] provide experimental evaluation that demonstrates greater robustness.

In Chapter 4 we use SURF descriptors both for place recognition in the KLT-tracking process. Although SURF is considerably faster than SIFT, CPU implementations are not suitable for real-time processing at standard video frame rates. To overcome this issue a public GPU based implementation is available through the OpenCV library [115].

Since their introduction researchers have investigated methods for reducing the matching times and the storage requirements involved in using both SIFT and SURF features descriptors. Typical approaches involve either the application of dimension-

ality reduction techniques or quantisation of the descriptors to reduce the number of bits required for each feature value. A demerit of these techniques is that they first involve the computation of the full size descriptor.

Calonder et al. [15] define the Binary Robust Independent Elementary Features (BRIEF) descriptor that take an alternative approach to computing an efficient descriptor directly from the image data. The approach taken is to use a fixed but random selection of pixel pairs in the neighbourhood, each of which is involved in an ordered binary test. The test simply evaluates whether the first graylevel is less than the second; evaluating to 1 if it is and 0 if it is not. Concatenating the outcomes results in a n-bit binary descriptor vector. The first advantage of this descriptor is that it is compact, with a second advantage that it can be efficiently compared using the Hamming distance. Experimental results show that BRIEF-64 (consisting of 512 bits) out performs both rotationally and non-rotationally invariant versions of SURF on a number of standard benchmarks.

Rublee et al. [121] extend on the work of Calonder et al. by first defining a variant of FAST that computes both the position and orientation of each image keypoint. This allows them to compute an oriented BRIEF feature descriptor. They call this the combined Oriented FAST detector and Rotated BRIEF descriptor ORB. The Binary Robust Invariant Scalable Keypoints (BRISK) of Leutenegger et al. [80] also aims to deal with BRIEF's sensitivity to image distortions by defining a scale space keypoint detector using a variant of the FAST detector in both space and scale dimensions.

The development of feature detector and descriptors shows little sign of abating with new algorithms being published every year. As this research progresses visual odometry systems are benefiting from a wider array of choices in terms of the detector / descriptor combination to employ. The criteria along which this choice is made involves issues such as invariance, speed of computation, storage size of the resulting detector, localisation accuracy, repeatability, to name a few.

3.7 Windowed Bundle Adjustment

The material that has been presented in this chapter has focussed on estimating camera motion between pairs of frames. The first step in that process is to detect a set of keypoints and then using the stereo configuration of the camera estimate their 3D positions. Computing correspondences of these keypoints provides the basis for visual odometry. This is achieved by using the camera pose estimation techniques in conjunction with model fitting techniques such as those presented in sections 3.4 and 3.5 to adjust the estimated extrinsic parameters of the camera so as to minimise the reprojection error (see Eq. 3.20).

The difficulty with the above approach is that it assumes precise knowledge of the 3D positions of the points. Of course given that these positions are computed from uncertain measurements in the image space this assumption is not valid. To cater for the assumption that there are errors in both the camera and scene estimates Eq. 3.20 can instead be optimised over both of these parameter sets. Such an optimisation process is referred to as Bundle Adjustment (BA) as is the process used in computer vision for solving the structure-from-motion (SfM) problem [146].

The advantage of BA is that it provides a ML estimate of the structure of the scene and the motion of the camera, taking all uncertainties into account. Solving the BA problem involves the use of non-linear estimation techniques such as Gauss-Newton or Levenberg-Marquardt optimisation. A particular property of the problem that makes it tractable over large sets of camera and structure variables is its natural sparsity. This is due to the fact that cameras do not interact directly with other cameras and points do not interact directly with other points. As such the optimiser that is employed should directly exploit the resulting sparsity.

Application of BA in visual odometry systems is typically achieved through a windowed approach [34] where as each frame is acquired the techniques from the previous sections are used to provide a first estimate of the camera motion and new

scene structure. A sliding bundle adjustment window due to the previous $n-1$ frames is then augmented with this new camera and the associated structure. Further discussion of the use of windowed bundle adjustment in visual odometry is provided by Scaramuzza and Fraundorfer [123].

3.8 Conclusions

In this chapter each of the steps required for the computation of a stereo visual odometry were reviewed. As can be seen from the material covered, creating a visual odometry involves a considerable number of components, and as a consequence the designer must make a number of important choices at each stage. In Chapters 4 and 5 we draw on the material covered in both this chapter and Chapter 2 in the development and evaluation of a real-time stereo visual SLAM system.

CHAPTER 4

Multi-Session Visual SLAM

4.1 Introduction

In this chapter we describe the main contribution of this thesis : the development of a real-time 6-DOF multi-session visual SLAM system for use in large scale environments. The chapter focusses on the design and implementation of the system with a comprehensive quantitative evaluation presented in Chapter 5. The contents of both these chapters have been published in two separate papers. Initial results were presented in McDonald et al. [91], while the full system implementation with the complete set of experimental results were presented in McDonald et al. [92].

The design and implementation of a visual SLAM system is a significant challenge both from a scientific and engineering perspective. Given that the central focus of this thesis was the extension and application of the anchor nodes formulation to real-time 6-DOF multi-session visual SLAM, the approach taken was to first develop a single session visual SLAM system as a foundation. This system was then extended

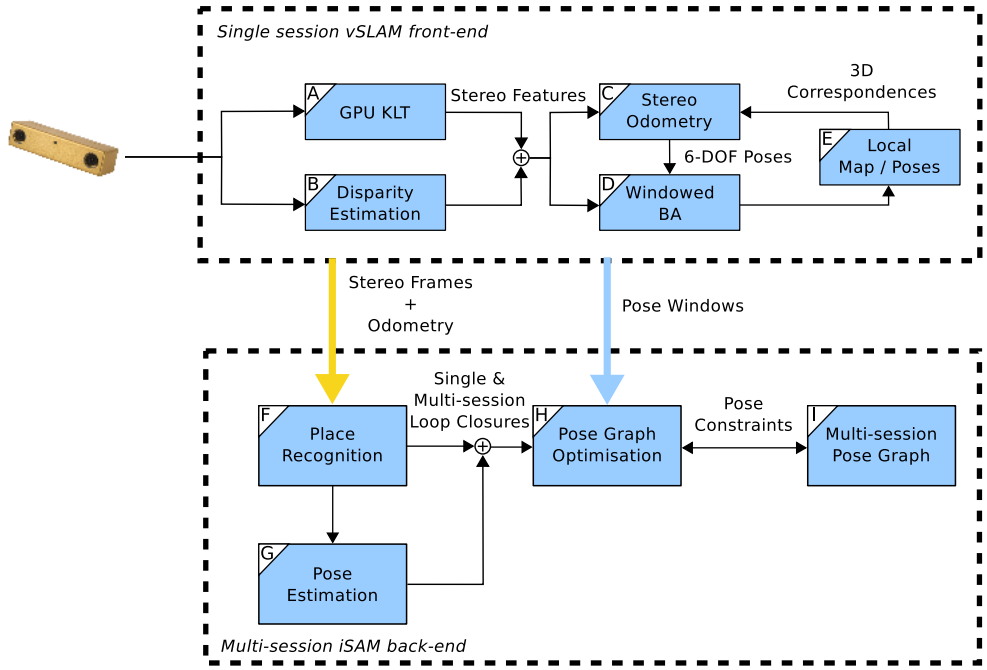


Figure 4.1: Multi-session visual SLAM system architecture. The major components of the system include: (A) 2-D feature tracker [153]; (B) dense stereo disparity estimation [10]; (C) stereo odometry [42, 109]; (D) windowed bundle adjustment using iSAM [61]; (E) local map representation; (F) visual place recognition [13]; (G) loop-closure pose estimation; (H) pose graph optimisation using iSAM; and (I) multi-session map pose graph representation.

to permit multi-session processing.

As such this chapter is divided into two halves. In Section 4.2 a detailed description of the overall system architecture and the operation of each of the individual components is presented from the perspective of single session operation. Section 4.3 then introduces the multi-session and multi-robot mapping problem, reviews the anchor node approach to the problem, and finally, describes the extension of the system to cater for multi-session operation.

4.2 System Overview

A schematic of the system architecture is shown in Fig. 4.1. This includes a stereo visual SLAM front-end, and a multi-session back-end incorporating a place recogni-

tion system for detecting single and multi-session loop closures, and a multi-session state-estimation system. The system uses a sub-mapping approach in conjunction with a global multi-session pose graph representation. Optimisation is performed by applying incremental and batch SAM to the pose graph and the constituent submaps, respectively. Each submap is constructed over consecutive sets of frames, where both the motion of the sensor and a feature based map of the scene is estimated. Once the current submap reaches a user defined maximum number of poses, 15 in our system, the global pose graph is augmented with the resultant poses.

In parallel to the above, as each frame is processed, the visual SLAM front-end communicates with a global place recognition system for intra and inter-session loop closure detection. When a loop closure is detected, pose estimation is performed on the matched frames, with the resultant pose and frame-id's passed to the multi-session pose graph optimisation module.

Algorithm 4.1 provides a detailed specification of the steps involved in the front-end processing. Here the input consists of a set of rectified stereo images, the current bundle adjustment window (*i.e.* the local map and poses), and the current number of poses in the window. For each input image the stereo odometry system computes the motion of the camera, however only if the motion is above a set threshold is it added as a new keyframe to the bundle adjustment window. Each keyframe is forwarded as input to the place recognition system as it arises, with the set of poses from each bundle adjustment window being forwarded when the window reaches it's maximum size. Algorithms describing the operation of the back-end are presented and discussed in Section 4.3.3.

Algorithm 4.1: Single session vSLAM front-end

Input: $\left\langle I_t^L, I_t^R \right\rangle$ Left and right rectified stereo frames
W current bundle adjustment window
w_s current number of poses in W
Output: W updated bundle adjustment window
w_s updated number of poses in W
Data: $\mathbf{P}_t$ current pose, D_t current disparity map, F_t current 2D features
M_t current 3D (*i.e.* $\langle x, y, disparity \rangle$) features
w_{max} maximum window size
L_t list of $\langle x, y, disparity \rangle$ feature and 3D landmark correspondences

```
1  do
2      D_t ← computeDisparity(I_t^L, I_t^R)
3      F_t ← detectFeaturesAndTrack(I_t^L)
4      M_t ← ∅
5      foreach f in F_t do                                  // Compute 3D features
6          if D_t[f.x, f.y] > disparity threshold then
7              M_t ← M_t ∪ ⟨f, D_t[f.x, f.y]⟩
8      P_t ← 0
9      if w_s = 0 then                                  // initialise first window
10         W.keyframe_0 ← ⟨P_t, M_t⟩
11     else
12         if w_s ≥ w_max or loop_detected then   // publish and reset window
13             publish poses from window W          // send poses to back-end
14             keyframe ← W.keyframe_{w_s}
15             W ← ∅
16             W.keyframe_0 ← keyframe
17             w_s ← 1
18         L_t ← ∅
19         foreach measurement x in M_t do      // assemble correspondences
20             if landmarkExists(x) then
21                 L_t ← L_t ∪ ⟨x, findLandmark(x)⟩
22         ⟨P_t, num_inliers⟩ ← computeMotion(L_t)
23         if num_inliers < inlier threshold then      // insufficient inliers
24             P_t ← recoverFromOdometryFailure()
25         else if movementTest(P_t) = false then   // insufficient motion
26             return
27         W.keyframe_{w_s} ← ⟨P_t, M_t⟩                        // add keyframe
28         publish keframe W.keyframe_{w_s}       // send keyframe to back-end
29         optimise(W)                                        // iSAM based WBA
30     w_s ← w_s + 1
31 end
```

4.2.1 Stereo Odometry

Within each submap the inter-frame motion and associated scene structure is estimated via a stereo odometry front-end. The most immediate benefit of the use of stereo vision is that it avoids issues associated with monocular systems, including the inability to estimate scale and indirect depth estimation. The stereo odometry approach we use is similar to that presented by [111].

Our stereo odometry pipeline tracks features using a standard robust approach, followed by a pose refinement step. For each pair of stereo frames we first track a set of Shi-Tomasi corners in the left frame using the KLT tracking algorithm. The resulting tracked feature positions are then used to compute the corresponding feature locations in the right frame. Approximate 6-DOF pose estimation is performed through the use of a RANSAC based 3-point algorithm [42]. The input to the motion estimation algorithm consists of the set of tracked feature positions and disparities within the current frame and the current estimates of the 3D locations of the corresponding landmarks. In our work we have found that ensuring that approximately 50 features are tracked between frames results in a reliable pose estimate through the 3-point RANSAC procedure. Finally, accurate pose estimation is achieved by identifying the inliers from the estimated pose and using them in a Levenberg-Marquardt optimisation that minimises the reprojection error in both the left and right frames.

In our stereo odometry implementation we use a GPU-based KLT tracker [153]. This minimises the load on the CPU (by delegating the feature detection and tracking to the GPU) and exploits the GPU's inherent parallel architecture to permit processing at high frame rates. In parallel to this we compute a disparity map for the frame using the dense block matching algorithm provided by OpenCV [10]. The output of the disparity estimate is then combined with the results of the feature tracker, resulting in a set of stereo features.

In order to maintain an adequate number of features we detect new features in

every frame whilst at the same time setting a minimum inter-feature distance in the KLT tracker. A consequence of this approach is that the system tries to ensure that there is a good distribution of features over the entire frame.

4.2.2 Single Session Visual SLAM

Deriving a pose graph representation from the stereo odometry system involves two levels of processing. The first of these optimises over the poses, features and 3D structure within a local bundle adjustment window. As each new frame is added iSAM can be configured to either perform a full batch optimisation or to use incremental optimisation. The benefit of the full batch approach is that it is more robust to the non-linearities involved, whereas the incremental approach provides a more computationally efficient solution.

The second step transfers optimised pose constraints from the current window to the pose graph once the window reaches a fixed maximum number of frames. Here we compute a set of odometry constraints between consecutive poses within the bundle adjustment window by marginalising out the landmarks between those poses. These constraints are then used to extend the pose graph structure maintained by the back-end, which contains no point features and can be optimised efficiently even for a large number of poses.

As each bundle adjustment window reaches its maximum number of poses, the front-end initialises the next window, W_{n+1}, using the last pose from the current window, W_n. This step corresponds to lines $12 - 17$ of Algorithm 4.1. Specifically the estimates of both the last pose and all the landmarks that are visible from that pose in window W_n, provide the first pose and initially visible landmarks for window W_{n+1}. Since the back-end pose graph is constructed through a chain of odometry constraints, this approach ensures the first constraint from each window provides a pose graph constraint between that window and the last pose in the current pose

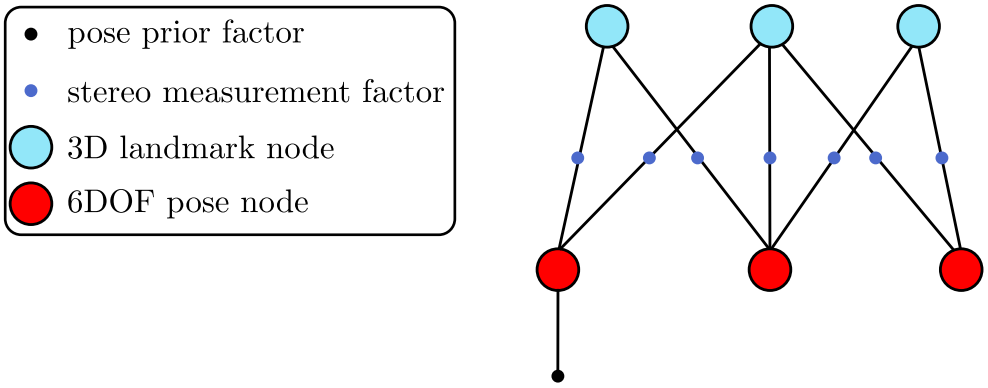


Figure 4.2: Structure of the bundle adjustment factor graph used in modules D & E of Fig. 4.1.

graph. Furthermore this approach has the effect that landmarks can traverse many windows thereby taking wide baseline constraints into account.

We apply smoothing in combination with a homogeneous point representation to the local window to improve the pose estimates obtained from visual odometry. In contrast to visual odometry, smoothing takes longer range constraints into account, which arise from a single point being visible in multiple frames. The homogeneous representation provides a principled mechanism for dealing with points at infinity, see [146] or [54]. Points close to or at infinity cannot be represented correctly by the conventional Euclidean parameterisation.

After removing the over-parameterisation of both rotation and homogeneous point representations (see 4.2.3), the optimisation problem is solved with a standard least-squares solver. We use the iSAM library [62] to perform the smoothing with Powell's Dog-Leg algorithm [118]. As discussed in Chapter 2, iSAM represents the optimisation as a factor graph, a bipartite graph containing variable nodes, factor nodes and links between variables and factors. Factor nodes (or *factors*) represent individual probability densities

$$f_i(\Theta_i) = f_i(\mathbf{x}_{j_i}, \mathbf{p}_{k_i}) \propto \exp\left(-\frac{1}{2}\left\|\Pi(\mathbf{x}_{j_i}, \mathbf{p}_{k_i}) - \mathbf{z}_i\right\|^2_{\Sigma_i}\right) \tag{4.1}$$

where $\Pi(\mathbf{x}, \mathbf{p})$ is the stereo projection of a 3D point $\mathbf{p}$ into a camera of given 3D pose $\mathbf{x}$, yielding the predicted stereo projections (u_L, v) and (u_R, v), $\mathbf{z}_i = (\hat{u}_L, \hat{u}_R, \hat{v})$ is the actual stereo measurement, and Σ_i represents the Gaussian image measurement noise. iSAM then finds the least-squares estimate Θ^* of all variables Θ (camera poses and scene structure combined) as

$$\Theta^* = \arg\max_{\Theta} \prod_i f_i(\Theta_i) \tag{4.2}$$

In order to reduce the computational requirements of the approach we employ a pose decimation scheme, whereby a threshold is applied on the translational and rotational motion of the sensor between poses (lines $25-26$ of Algorithm 4.1). In our current implementation the first pose of each session corresponds to the first frame of the input sequence and is initialised to the 6-DOF origin for that session. Subsequent to this each new pose corresponds to the first frame where the inter-frame motion is at least 0.2m or 0.2rad from the last pose. The effect of this is to reduce the total number of poses within the pose graph, whilst maintaining the accuracy of the final trajectory and map estimates. A secondary but important advantage of this approach is that it also decreases drift, in particular when the sensor is stationary.

When the smoothing window reaches a maximum size or when a loop closure is detected all poses and associated odometry are transferred to the current session's pose graph, and a new local window is initialised. By including all poses from a window, as opposed to just the first or first and last pose (as is the case in other approaches) we ensure that we can represent loop closures between arbitrary frames within the pose graph. Full details of the loop closure handling are provided in Sections 4.2.4 & 4.3.1. As mentioned above, to initialise a new window we use the last pose of the previous window in conjunction with all landmarks that correspond to features that are tracked into the current frame. Note that within the window

the first pose is always set to the origin to avoid optimisation issues with the bundle adjustment when cameras are far from the origin (see Section 4.2.3).

The pose graph is again optimised using the iSAM library [62], where we use the incremental iSAM algorithm [61] to efficiently deal with large pose graphs. In particular we apply the incremental algorithm when dealing with consecutive pose constraints, and apply the batch algorithm for loop closure constraints (i.e. to cater for the fact that such constraints can require more significant updates to the current estimate). In contrast to the stereo projection factors f_i in the smoothing formulation above, we now use factors g_i

$$g_i(\Theta_i) = g_i(\mathbf{x}_{j_i}, \mathbf{x}_{j_i'}) \propto \exp\left(-\frac{1}{2}\left\|(\mathbf{x}_{j_i'} \ominus \mathbf{x}_{j_i}) - \mathbf{c}_i\right\|_{\Xi_i}^2\right) \tag{4.3}$$

that represent constraints $\mathbf{c}_i$ with covariances Ξ_i between pairs of poses as obtained by local smoothing or by loop closure detection. We use the notation $\mathbf{x}_d = \mathbf{x}_b \ominus \mathbf{x}_a$ from Lu and Milios [84] for representing pose $\mathbf{x}_b$ in the local frame of pose $\mathbf{x}_a$ ($\mathbf{x}_b = \mathbf{x}_a \oplus \mathbf{x}_d$).

4.2.3 Quaternions and Homogeneous Points

The projective representation of point features as well as the quaternion representation of rotations that we use here result in over-parameterisations that can be resolved in much the same way.

The unit sphere $S^3 = \{\mathbf{q} \in \mathbb{R}^4 : \|\mathbf{q}\| = 1\}$ can be identified with the set of unit quaternions which form a 3-dimensional Lie group under quaternion multiplication. There is a two-to-one covering map from S^3 onto $SO(3)$ (antipodal points are identified because $\mathbf{q}$ represents the same rotation as $-\mathbf{q}$). The matrix Lie algebra of $SO(3)$ is $\mathfrak{so}(3)$, the set of skew-symmetric matrices, see Hall [51]. Because they have three parameters they provide a minimal local parameterisation of rotations through an exponential map (typically evaluated using Rodrigues' formula). The elements of the

Lie algebra of S^3 can be identified with the tangent space $\mathbb{R}^3$ at the identity. Many mappings exist from $\mathbb{R}^3$ to S^3, here we use the following one from Grassia [47]:

$$\exp\left(\frac{\mathbf{d}}{2}\right) = \begin{pmatrix} \frac{1}{2}\mathrm{sinc}\left(\frac{1}{2}\|\mathbf{d}\|\right)\mathbf{d} \\ \cos\left(\frac{1}{2}\|\mathbf{d}\|\right) \end{pmatrix} \tag{4.4}$$

where $\mathbf{d} \in \mathbb{R}^3$ coincides with the axis/angle representation of a rotation. As for every minimal representation of rotations there are singularities, here at multiples of 2π, though they can be avoided by forcing $\mathbf{d}$ to fall into the range $(-\pi, \pi]$, while still allowing for all possible rotations. An existing quaternion $\mathbf{q}$ is updated by an increment $\mathbf{d}$ using quaternion multiplication $\mathbf{q}\exp\left(\frac{\mathbf{d}}{2}\right)$. This then provides the necessary machinery for incorporating the minimal representation within the least-squares optimisation, whereby the parameter updates are first computed in the minimal representation, and then applied through the use of the exponential map.

We show that the projective parameterisation in 3D is isomorphic to unit quaternions, allowing the use of the same exponential map. The projective parameterisation uses homogeneous four-vectors $\mathbf{p} = (x, y, z, w)^\top \in \mathbb{R}^4\backslash\{0\}$ with the zero vector excluded. A Euclidean point (x, y, z) is written in homogeneous coordinates as $\lambda(x, y, z, 1)$ for $\lambda \in \mathbb{R}\backslash\{0\}$, while points at infinity satisfy $w = 0$. In this real projective space $\mathbb{RP}^3$, points along lines through the origin are equivalent by the relation $\mathbf{p} \sim \lambda\mathbf{p}, \lambda \in \mathbb{R}\backslash\{0\}$. The set of homogeneous points $\mathbf{p} \in \mathbb{R}^4, \|\mathbf{p}\| = 1$ with unit norm spans the 3-sphere S^3 and provides a double cover of this real projective space (antipodal points are identified). Therefore, the same exponential map as for quaternions is applicable to normalised homogeneous points.

An alternative map is presented in Hartley and Zisserman [54, Appendix 6.9.3] that uses Householder transformations instead of quaternion multiplication. Both are valid, and we have not seen a major difference in their convergence. Note that both methods only work well for bundle adjustment as long as the cameras are near the

origin, which is easy to satisfy for our windowed bundle adjustment. Intuitively, the parameterisation becomes more nonlinear for cameras with centre far from the origin.

4.2.4 Place Recognition

Place recognition is an important component in the context of large-scale, multi-robot and multi-session SLAM, where algorithms based on visual appearance are now widely used. Central to the performance of these algorithms has been the developments in invariant interest point detection and feature descriptors as discussed in Section 3.6. Although these approaches detect and describe local patches in the image, in 2003 Sivic and Zisserman [128] developed a global descriptor and matching scheme analogous to techniques previously developed in the text retrieval community. The bag-of-words (BoW) approach first builds a visual vocabulary in the feature descriptor space by clustering descriptors computed over a large training set of images. The resulting clusters become words in the vocabulary. Given this vocabulary a global BoW descriptor can be formed for an image by first computing the feature descriptors which are then mapped to corresponding words in the vocabulary. The BoW descriptor consists of a vector of weighted word frequencies, where the weighting is known as "term-frequency-inverse document frequency" (tf-idf). The effect of the tf-idf weighting is to increase the importance of words that occur frequently in a given image, but at the same time decrease the importance of words that occur frequently across the training set. Fast retrieval is achieved through the use of an inverted file structure which maps each word in the vocabulary to all occurrences of that word in the image dataset.

Nistér and Stewénius [110] built on the work of Sivic and Zisserman [128] through the use of a hierarchical approach leading to vocabulary tree structure. This has the effect of permitting much larger vocabularies with consequent improvement in retrieval rates. Furthermore the use of the vocabulary tree results in far fewer images

from the database being considered during each look-up thereby allowing searches over millions of images.

In this thesis we have used a place recognition software module provided by Cesar Cadena based on his recent work [13, 14], which has been demonstrated to have robust and reliable performance. Although this work does not form part of the contributions of this thesis, the technique is briefly summarised here for completeness and to explain the parameters that were used in the experiments described in Chapter 5. The key to the performance of Cadena's approach is the use of two separate stages. As each new image is presented to the system it is first assessed by a BoW appearance based matcher [128], which is implemented using the hierarchical approach of Nistér and Stewénius [110]. The output of this stage is a normalised similarity score, η_c, that computes the ratio between two matches

$$\eta_c(t, t') = \frac{s(t, t')}{s(t, t-1)} \tag{4.5}$$

where, $s(t, t')$ is the matching score between the input and the best matched candidate, and $s(t, t-1)$ is the match between the current input and the previous input to the system. Matches where $\eta_c \geq 0.8$ are output as loop closure candidates. Matches where $\eta_c \leq 0.15$ are rejected.

Matches where $0.15 < \eta_c < 0.8$ are input to a more computationally intensive second stage evaluation utilising a Conditional Random Field (CRF) based matching strategy. The use of the CRF-based approach permits the combination of both visual and 3D geometric information (derived from the stereo data), thereby permitting more robust matching. The approach taken is a CRF-based algorithm proposed for matching 2D laser scans [116] and for matching image features [117]. In [13] CRF-Matching was extended to reason in 3D space about the association of data provided by a stereo camera system in order to verify loop closures hypothesis. This

verification stage was improved in [14] taking into account the distant information *i.e* the remaining information in the image without 3D information.

They compute the negative log-likelihood $\Lambda^{\mathcal{G}}_{t,t'}$ from the maximum a posteriori (MAP) association between the current scene in time t against the candidate scene in time t'. Loop closure candidates are output from this stage only if the normalised similarity scores assert $\eta_{3D} \leq \beta_{3D} \wedge \eta_{Im} \leq \beta_{Im}$, with:

$$\eta_{\mathcal{G}} = \frac{\Lambda^{\mathcal{G}}_{t,t'}}{\Lambda^{\mathcal{G}}_{t,t-1}} \tag{4.6}$$

where $\mathcal{G}$ indicates the graph, $3D$ or Im. $\beta_{\mathcal{G}}$ is a control parameter that defines the level of similarity we demand for $(t, t-1)$ in terms of close range β_{3D}, and far range β_{Im}. Where smaller values for β means a higher demand. In the system described in this thesis the values of $\beta_{3D} = \beta_{Im} = 2$ were used.

For each session a new image is added to the database whenever the sensor's motion exceeds a threshold of 0.2m or 0.2rad based on the output from the front-end's motion estimation. All loop closure candidates are input to a pose estimation module which also acts as geometric filter to validate the candidates. Full details of this process are provided in Section 4.3.1.

4.3 Multi-session and Multi-Robot Mapping

Operating over larger spaces and timescales raises a number of separate but related challenges for SLAM algorithms and systems. As the timescale of the mapping process increases to weeks or months, carrying it out in a single contiguous traversal through the environment becomes impractical. As a consequence the assumption that the input consists of a single contiguous data stream becomes invalid. Instead such scenarios typically consist of repeated visits or traversals through an environment possibly with the acquisition platform being shutdown and moved in the interven-

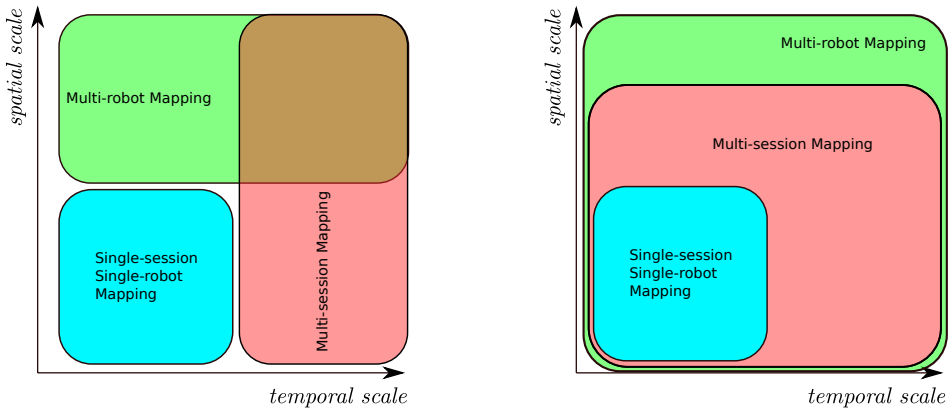


Figure 4.3: (a) Requirements for, and (b) applicability of single-session single-robot, multi-session, and multi-robot mapping over varying spatial and temporal scales. Increasing the spatial and temporal scales of the mapping process makes capturing all of the data in a single contiguous capture session impractical. As a consequence multi-robot and multi-session mapping becomes a more pertinent issue.

ing periods. Hence, multi-session mapping algorithms assume the input consists of a number of possibly unordered distinct sessions with no assumptions being made regarding the relationship between those sessions.

Similarly, as the scale of the domain increases so too does the requirement to use multiple robots in a distributed or collaborative approach [114]. Both of these situations are closely related (*e.g.* many multi-robot approaches deal with the unknown initial relative pose [142, 68]), however there are some subtle differences. In particular multi-session mapping is more amenable to centralised approaches due to the fact that they have less stringent bandwidth requirements. One of the advantages of multi-robot systems is that they can scale to large spaces whilst keeping the computation time bounded through the inherent parallelism of using multiple robots. The cost of running multiple front-ends in parallel is higher communication bandwidth requirements. Furthermore in order to achieve truly scalable multi-robot algorithms requires fully decentralised processing which makes maintaining global consistency more difficult [22, 23, 24]. Conceptually, the relationship between the applicability

of single-session (single-robot), multi-session, and multi-robot mapping systems in terms of spatial and temporal scales is illustrated in Fig. 4.3. Of course any multi-robot algorithm can be used for multi-session mapping which in turn can be used for single-session mapping. Furthermore, given the standard architecture of dividing SLAM systems into separate front-end and back-end optimisation modules, assuming the bandwidth limitations of the infrastructure are not reached, most multi-session SLAM algorithms can be applied in multi-robot situations.

4.3.1 Multi-session mapping with iSAM and anchor nodes

For multi-session mapping we use one pose graph for each robot/camera trajectory, with multiple pose graphs connected to one another with the help of "anchor nodes", as introduced in Kim *et al.* [66] and Ni and Dellaert [108].

In this work we distinguish between intra-session and inter-session loop closures. Processing of loop closures is performed firstly with each frame corresponding to a new pose being input to the place recognition system (see Section 4.2.4). These candidate frames are matched against previously input frames from all sessions. On successful recognition of a loop closure the place recognition system returns the matched frames' session and frame identifiers. The SURF features for both images in the image pair are matched using the Fast Library for Approximate Nearest Neighbour, which computes the matches based on a k-nearest neighbour search resulting in a set of stereo feature correspondences between the two frames. These feature sets consist of lists of SURF feature locations and stereo disparities. Note that since these features are already computed and stored during the place recognition processing, their use here does not place any additional computational load on the system.

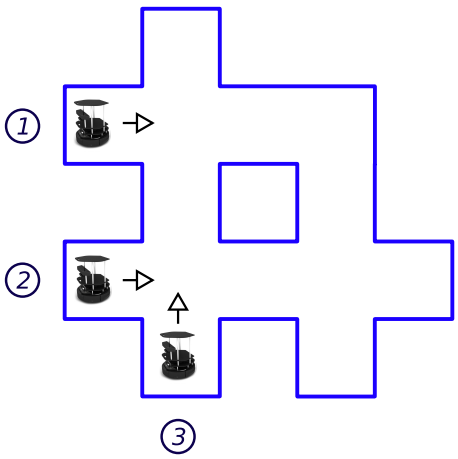
These feature sets serve as input to the same camera orientation estimation system described in Section 4.2.1. Here the disparities for one of the feature sets are used to perform 3D reconstruction of the points in the scene. These 3D points are passed

with their corresponding 2D features from the second image into a 3-point algorithm-based RANSAC procedure. Finally the estimated orientation is iteratively refined through a non-linear optimisation procedure that minimises the reprojection error in conjunction with the disparity.

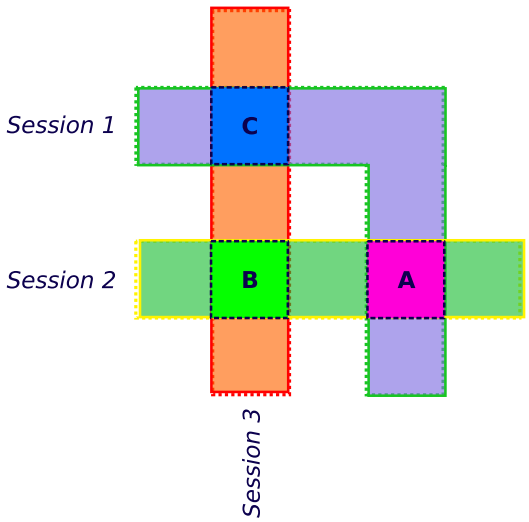
Inter-session loop closures introduce encounters between pose graphs corresponding to different visual SLAM sessions. An encounter between two sessions s and s' is a measurement that connects two robot poses $\mathbf{x}_j^s$ and $\mathbf{x}_{j'}^{s'}$. This is in contrast to measurements between poses of a single trajectory, which are of one of two types: The most frequent type of measurement connects successive poses, and is derived from visual odometry and the subsequent local smoothing. A second type of measurement is provided by intra-session loop closures.

The use of anchor nodes [66] allows at any time the combination of multiple pose graphs that have previously been optimised independently. The anchor node Δ^s for the pose graph of session s specifies the offset of the complete trajectory with respect to a global coordinate frame. That is, we keep the individual pose graphs in their own local frame. Poses are transformed to the global frame by pose composition $\Delta^s \oplus \mathbf{x}_i^s$ with the corresponding anchor node.

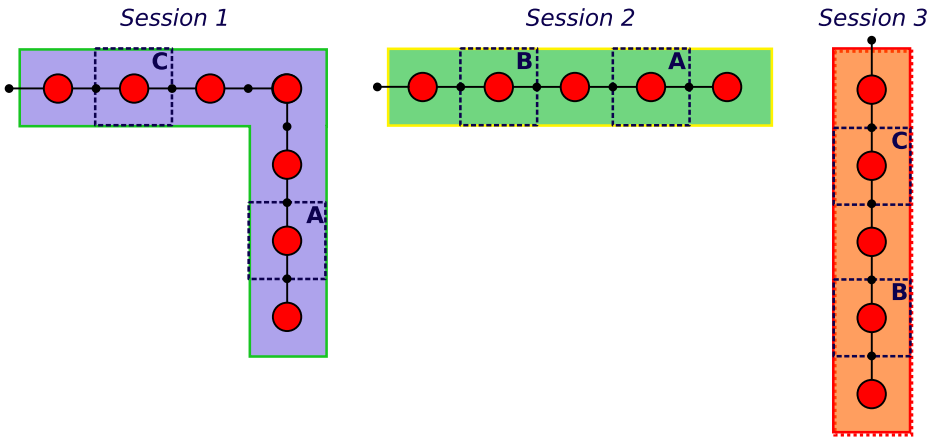
In this relative formulation, the pose graph optimisation remains the same, only the formulation of encounter measurements involves the anchor nodes. The factor describing an encounter between two pose graphs will also involve the anchor nodes associated with each pose graph. The anchor nodes are involved because the encounter is a global measure between the two trajectories, but the pose variables of each trajectory are specified in the session's own local coordinate frame. The anchor nodes are used to transform the respective poses of each pose graph into the global frame, where a comparison with the measurement becomes possible. The factor h



(a) Hypothetical environment including starting point for each robot mapping session.



(b) Areas of environment covered by each session with regions where encounters occur highlighted at A, B, & C.



(c) Pose graphs of each of the individual sessions, represented as factor graphs, registered with the corresponding region of the environment. Here the poses involved in encounters are those that coincide with regions A, B, & C. For example assuming the sessions occur sequentially and in numerical order, the first encounter occurs between Session 1 and Session 2 at region A. From the above pose graphs it can be seen that this encounter involves the fifth and fourth poses from Session 1 and Session 2, respectively.

Figure 4.4: Hypothetical multi-session mapping scenario consisting of three separate sessions, where (a) shows the environment to be mapped, (b) shows the regions mapped by each session, and (c) show the individual pose graphs for each of the sessions (excluding encounter constraints). Subsequent figures show the complete multi-session pose graph. Note that although in this scenario we assume the sessions occur sequentially, there is no inherent requirement for this to be the case when using anchor nodes. On the contrary, the anchor node framework can be used to combine concurrent mapping sessions and makes no assumption regarding inter-session ordering or timing.

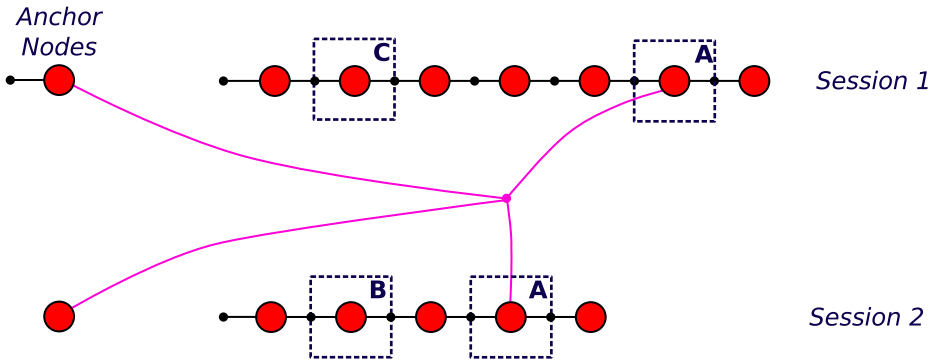


Figure 4.5: Multi-session factor graph incorporating sessions 1 and 2 (see Fig. 4.4(c)), and the encounter at location A (see Fig. 4.4(b)). Note that the factor graph for session 1 has been laid out linearly to simplify the illustration. Here each pose graph is represented using a relative coordinate frame anchored using the priors on the first poses. Although these can be set arbitrarily, typically we set them to the origin. Also shown are the anchor nodes for each session which are used to estimate the pose of the respective session relative to the global reference frame. In this example the encounter between the two poses in the region A induces a constraint between those poses *and* the anchor nodes. This constraint is captured by the encounter factor shown in purple.

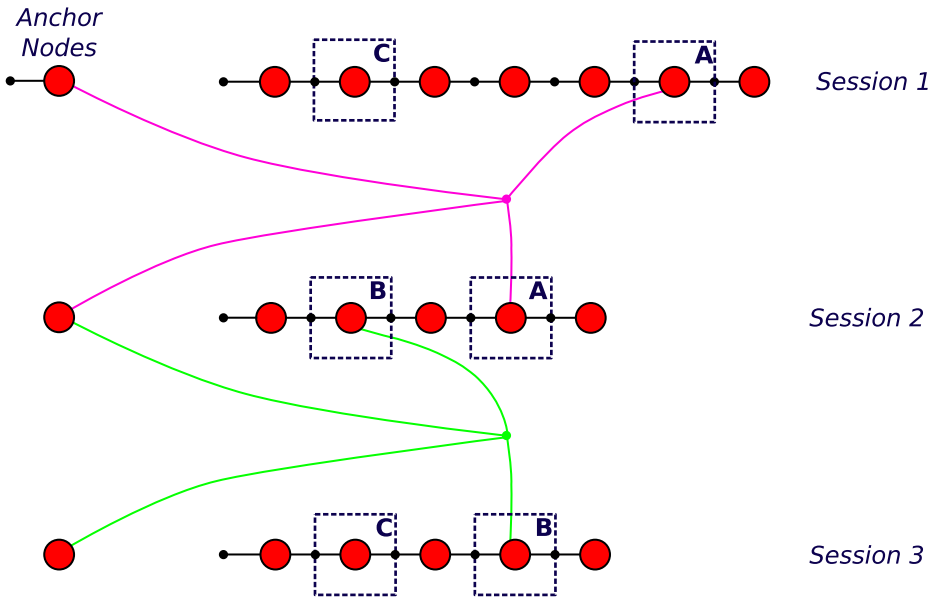


Figure 4.6: Following on from Fig. 4.5, this figure shows the multi-session factor graph incorporating sessions 1, 2, and 3 (see Fig. 4.4(c)), and the encounters at locations A and B (see Fig. 4.4(b)). Again, each pose graph is represented using a relative coordinate frame anchored using the priors on the first poses. Just as with sessions 1 and 2 in Fig. 4.5, session 3 is augmented with an anchor node for estimating its pose in the global reference frame. This figure also adds the encounter between the two poses in region B inducing a constraint between those poses and the associated anchor nodes. This constraint is captured by the encounter factor shown in green.

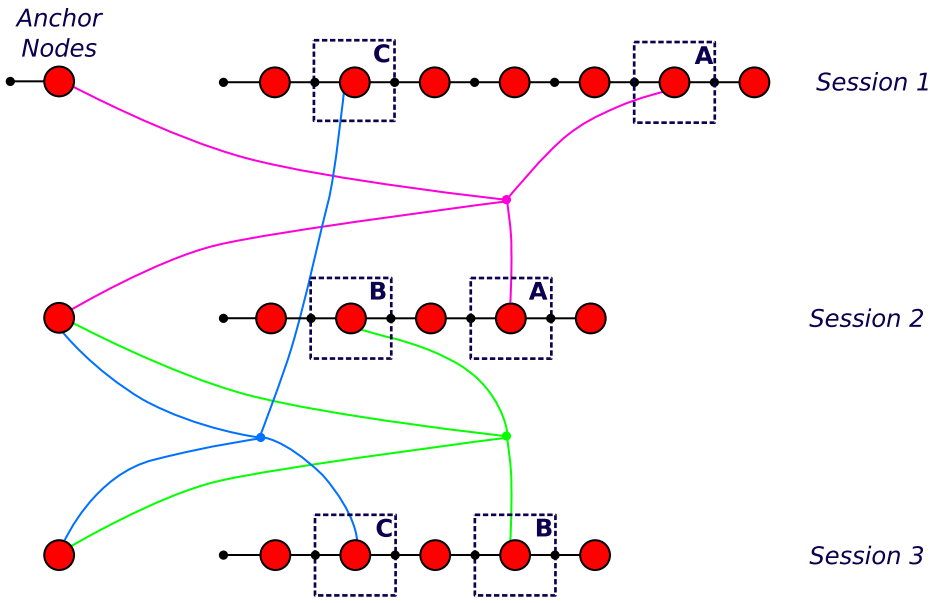


Figure 4.7: Complete factor graph for scenario described in Fig. 4.4, including relative pose graphs for each of the individual sessions, anchor nodes for estimating the global poses of each session, and encounter constraints due to encounters at locations A (purple), B (green), and C (blue) shown in Fig. 4.4(b).

describing an encounter $\mathbf{c}_i$ is given by

$$h(\mathbf{x}_j^s, \mathbf{x}_{j'}^{s'}, \Delta^s, \Delta^{s'}) \propto \exp\left(-\frac{1}{2}\left\|((\Delta^s \oplus \mathbf{x}_j^s) \ominus (\Delta^{s'} \oplus \mathbf{x}_{j'}^{s'})) - \mathbf{c}_i\right\|_\Gamma^2\right) \tag{4.7}$$

These inter-session constraints are incorporated into the cost function defined in Eq. 4.2 in the same way as the other measurements. Figs. 4.4 – 4.7 provides a graphical example of this process. The concept of relative pose graphs generalises well to a larger number of sessions, with the number of anchor nodes depending only on the number of sessions.

4.3.2 A 1-D example of multi-session SLAM

In his PhD thesis, Nico Sünderhauf [137] provides an intuitive example of solving a trivial 1-D SLAM problem in order to provide a comprehensive view of all the steps involved. This section aims to do the same but extends the example to multi-session

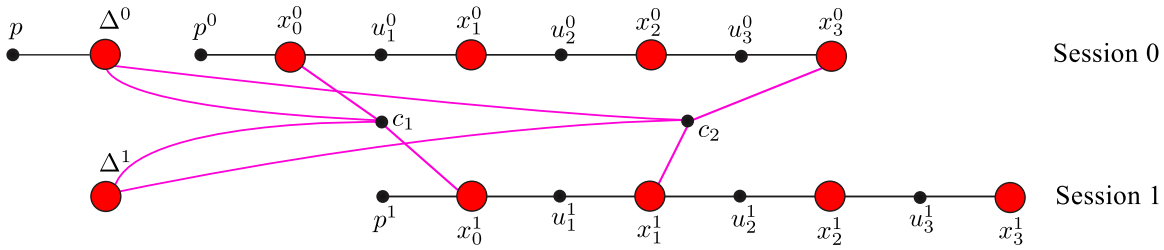


Figure 4.8: Example 1-D multi-session SLAM scenario. Here two pose graphs are built in the same environment where Session 1 is displaced relative to Session 0. Inter-session loop closure constraints, c_1 and c_2, occur between poses x_0^0 and x_0^1, and, x_1^1 and x_3^0, respectively.

SLAM. In particular the example highlights the effect of the use of anchor nodes in terms of estimating the global frames of each of the sessions whilst permitting the use of a relative representation within each of the sessions. Also it shows the effect of the anchor nodes in terms of the variables being estimated, the Jacobian, and the information matrix.

In this example we define two sessions each containing four poses separated by unit increments and joined by two inter-session encounters. A factor graph representation of the example is shown in Fig. 4.8, with the associated data for the example provided in Table 4.1.

To solve the above system we first need to compute the elements of the normal equations as given in Eq. 2.19. In the case of the simple 1D motion model, and assuming unit time steps between pairs of poses, the odometry factors (see Eq. 2.14) are given by,

$$f(x_i, x_j; u_j) = \|x_i + u_j - x_j\|_\Sigma^2 \tag{4.8}$$

Similarly the encounter factors are given by,

$$h(x_j^s, x_{j'}^{s'}, \Delta^s, \Delta^{s'}) = \|((\Delta^s + x_j^s) - (\Delta^{s'} + x_{j'}^{s'})) - c\|_\Gamma^2 \tag{4.9}$$

Note here that these factors actually result in a linear least squares problem, however we still take a non-linear approach to highlight the steps involved when using non-

Session 0

pose	ground truth pose	measured odometry	estimated pose
x_0^0	0.0	0.9	0.0
x_1^0	1.0	1.0	0.9
x_2^0	2.0	1.1	1.9
x_3^0	3.0	-	3.0

(a)

Session 1

pose	ground truth pose	measured odometry	estimated pose
x_0^1	1.0	1.1	0.0
x_1^1	2.0	1.1	1.1
x_2^1	3.0	0.9	2.2
x_3^1	4.0	–	3.1

(b)

Inter-Session Loop Closures

encounter	pose 1	pose 2	ground truth constraint	measured constraint
c_0	x_0^0	x_0^1	1.0	0.9
c_1	x_3^0	x_1^1	-1.0	-0.8

(c)

Table 4.1: Example 1-D multi-session SLAM scenario.

linear motion or measurement models.

Given the above factors the residual vector is constructed by stacking the residuals for each of the individual sessions followed by the residuals for the encounters as follows,

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_0^0 + u_0^0 - x_1^0 \\ x_1^0 + u_1^0 - x_2^0 \\ x_2^0 + u_2^0 - x_3^0 \\ x_0^0 - 0 \\ x_0^1 + u_0^1 - x_1^1 \\ x_1^1 + u_1^1 - x_2^1 \\ x_2^1 + u_2^1 - x_3^1 \\ x_0^1 - 0 \\ ((\Delta^1 + x_0^1) - (\Delta^0 + x_0^0)) - c_0 \\ ((\Delta^1 + x_1^1) - (\Delta^0 + x_3^0)) - c_1 \\ \Delta^0 - 0 \end{pmatrix} \Rightarrow \mathbf{f}(\mathbf{x}_0) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -0.9 \\ -1.1 \\ 0 \end{pmatrix} \tag{4.10}$$

Note here that the fourth, eight and eleventh elements here are due to the priors on the origin poses of the two sessions and on the first session's anchor nodes. Since the system includes encounters between the two sessions, no prior is required on the second session's anchor node. As can be seen in the above, since the estimated positions are initialised from the measured odometry (*i.e.* through dead-reckoning) all of the residuals are zero within the individual sessions. However due to the fact that the anchor nodes have both been initialised to zero when in fact there is a unit translational shift between the sessions, the residuals on each of the inter-session loop constraints are non-zero.

From Eq. 4.10 the Jacobian is given by

$$J = \begin{pmatrix} \mathbf{1} & \mathbf{-1} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{0} & \mathbf{1} & \mathbf{-1} & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{-1} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{-1} & \mathbf{0} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{-1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{-1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{-1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{-1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \end{pmatrix} \tag{4.11}$$

where the blocks due to Session 1, Session2, and the anchor nodes have been highlighted in bold.

Finally, taking the covariance on all odometry and loop closure measurements to be 0.01 and the covariance on the priors to be 0.001 leads to the following information matrix

$$\Omega = \Sigma^{-1} = diag\Big(\ 0.01, 0.01, 0.01, 0.001, 0.01, 0.01, 0.01, 0.001, 0.01, 0.01, 0.001 \ \Big)^{-1} \tag{4.12}$$

$$= diag\Big(\ 100, 100, 100, 100, 1000, 100, 100, 100, 1000, 100, 100, 1000 \ \Big) \tag{4.13}$$

As shown in Eq. 2.19 the normal equations allow us to solve for an update vector,

$$A^T A \boldsymbol{\delta\theta}^* = A^T \mathbf{b} \tag{4.14}$$

$$J^T \Omega^T J \boldsymbol{\delta\theta}^* = J^T \Omega^T \mathbf{f} \tag{4.15}$$

Taking the left hand side, the system information matrix, which is typically taken as

an approximation to the Hessian, $H = J^T \Omega^T J$, is given by

$$H = \begin{pmatrix} 1200 & -100 & 0 & 0 & -100 & 0 & 0 & 0 & 100 & -100 \\ -100 & 200 & -100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -100 & 200 & -100 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -100 & 200 & 0 & -100 & 0 & 0 & 100 & -100 \\ -100 & 0 & 0 & 0 & 1200 & -100 & 0 & 0 & -100 & 100 \\ 0 & 0 & 0 & -100 & -100 & 300 & -100 & 0 & -100 & 100 \\ 0 & 0 & 0 & 0 & 0 & -100 & 200 & -100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -100 & 100 & 0 & 0 \\ 100 & 0 & 0 & 100 & -100 & -100 & 0 & 0 & 1200 & -200 \\ -100 & 0 & 0 & -100 & 100 & 100 & 0 & 0 & -200 & 200 \end{pmatrix} \tag{4.16}$$

while the right hand side, $\mathbf{b} = J^T \Omega^T \mathbf{f}$, is given by

$$\mathbf{b} = \begin{pmatrix} 90 & 0 & 0 & 110 & -90 & -110 & 0 & 0 & 200 & -200 \end{pmatrix}^T \tag{4.17}$$

As discussed in Section 2.4, Eq. 4.14 can be solved using either Cholesky or QR-factorisation which results in an update,

$$\boldsymbol{\delta\theta}^* = \begin{pmatrix} 0 & -0.033 & -0.067 & -0.1 & 0 & 0.033 & 0.033 & 0.033 & 0 & 0.933 \end{pmatrix}^T \tag{4.18}$$

where, as expected, the most significant change is applied to the anchor node for Session 2. The updates to the individual sessions on the other hand are due to the *relative* residuals introduced by the inter-session encounters, as opposed to the global displacement between sessions.

4.3.3 Multi-session implementation

The complete multi-session visual SLAM system follows the architecture shown in Fig. 4.9. Here each input image sequence constitutes a separate session and is processed independently by a dedicated vSLAM front-end. The output of each front-end

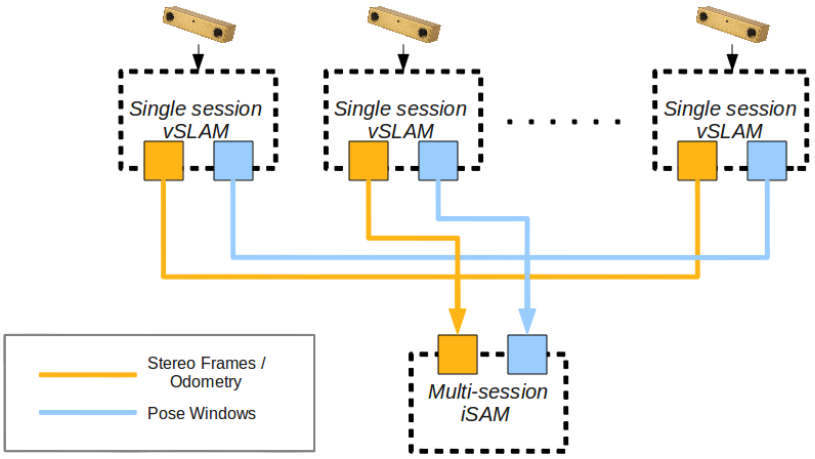


Figure 4.9: Multi-session visual SLAM architecture

is processed by the multi-session back-end which is responsible for multi-session place recognition and pose graph estimation.

The two events that update the multi-session pose graph are (i) when the front-end outputs the latest set of poses from the windowed bundle adjustment process, and, (ii) when the loop closure system outputs a new intra or inter-session loop closure. Algorithms for handling both of these events are shown in Algs. 4.2 and 4.3, respectively.

Algorithm 4.2 takes as input a window from the front-end and a session identifier specifying the session that published the window. If the session is new, the algorithm first creates a new pose node and attaches a prior factor to this node, where the factor causes the pose node to be initialised to the origin. Next an anchor node is created for the session and also attached to a factor that results in an initialisation to the origin.

The algorithm then proceeds to iterate through the input window computing the odometry between each pair of consecutive poses. For each pose, starting with the second pose, a new pose node is created. Next an odometry factor between the new pose node and the previously added pose node is created to encode the odometry measurement. As each pose node or factor is created they are added to the

multi-session pose graph which is then optimised using the multi-session formulation described in the previous sections.

Algorithm 4.3 details the steps involved in incorporating new loop closures into the existing multi-session pose graph. The algorithm takes as input a multi-session loop closure constraint consisting of (i) session identifiers and pose indices within the sessions for the two poses involved in the loop closure, and, (ii) a 6-DOF pose constraint between the two poses. Intra-session constraints are identified when the two session identifiers are equal resulting in a loop closure factor within that session. These factors are parameterised by the two poses and the 6-DOF pose constraint. Inter-session constraints on the other hand result in encounter factors linking the respective sessions. These factors are parameterised by the anchor nodes and poses from each session, and the 6-DOF pose constraint.

One point to note is that although the PR system currently returns only the best match, it is possible to employ other strategies. For example the current architecture could be extended to allow the PR system to return multiple candidates (*e.g.* between more than two sessions simultaneously). Here, each candidate would then be processed independently, *i.e.* by performing the consistency check, *etc.*, prior to being integrated into the pose graph.

The internal components of the front-end and back-end (as shown in Fig. 4.1) are implemented as a set of loosely coupled processes that communicate using the *Lightweight Communications and Marshalling* (LCM) robot middleware system [57]. This permits straightforward parallelism between the components of the system, hence minimising the impact on all modules due to fluctuations in the load of a particular module (e.g. due to place recognition deferring to CRF processing). Furthermore the overall architecture can be transparently reconfigured for different setups (e.g. from single CPU to multi-core or distributed processing).

An important additional benefit of using LCM was the rich toolset developed

Algorithm 4.2: Multi-session back-end: Add pose window

Input: PL Pose List: list of poses $\{\mathbf{P}_0 \dots \mathbf{P}_N\}$from step 13 of Alg. 4.1
S session identifier associated with PL
Data: PG Multi-session pose graph $\{(\Delta_0, PG_0), \dots, (\Delta_N, PG_N), c_0, \dots, c_m\}$
Δ_i Anchor node for session i
PG_i Relative pose graph for session i
c_i Inter-session loop closure constraint

1 **do**
2 **if** ¬**exists**(PG_S) **then** `// init new session`
3 create pose node N_0
4 add N_0 to PG_S
5 create pose $\mathbf{P} \leftarrow \mathbf{0}$
6 create prior pose factor $F_0(N_0, \mathbf{P})$
7 add F_0 to PG_S
8 create anchor node Δ_S
9 add Δ_S to PG
10 create pose factor $D_S(\Delta_S, \mathbf{P})$
11 add D_S to PG
12 $n \leftarrow$ **numPoses**(PG_S)
13 $N_c \leftarrow PG_S.N_{n-1}$ `// set` N_c `to last node of pose graph`
14 **for** *i=1 to N* **do**
15 create new pose node N_n
16 add N_n to PG_S
17 create odometry factor $F_i(N_c, N_n, \mathbf{P}_i \ominus \mathbf{P}_{i-1})$
18 add F_i to PG_S
19 $N_c \leftarrow N_n$
20 **optimise**(PG)
21 **end**

around the library for logging, playback, visualisation, and integration of data acquired from a wide variety of sensors (e.g. lidar). The system developed in this thesis utilises many of these features in particular for the experimental evaluation and for visualisation of the system during operation. A screenshot of the system is shown in Fig. 4.10. The system is built on top of a multimodal visualisation application developed at MIT as part of the DARPA Urban Challenge.

The figure shows the system during operation where two session have been processed, with the map for the first session shown in green, and the map for the second

Algorithm 4.3: Multi-session back-end: Add loop closure

// Add loop closure constraint $\boldsymbol{\delta}$ between pose nodes N_{n_i} and N_{n_j} of sessions S_{s_i} and S_{s_j}, respectively

Input: $c = \{s_i, n_i, s_j, n_j, \delta\}$ Multi-session loop closure constraint

Data: PG Multi-session pose graph $\{(\Delta_0, PG_0), \ldots, (\Delta_N, PG_N), c_0, \ldots, c_m\}$
Δ_i Anchor node for session i
PG_i Relative pose graph for session i
c_i Inter-session loop closure constraint

1 **do**
2 **if** $s_i = s_j$ **then**
3 create odometry factor $F(PG_{s_i}.N_{n_i}, PG_{s_i}.N_{n_j}, \boldsymbol{\delta})$
4 add F to PG_{s_i}
5 **else**
6 create odometry factor $F(PG_{s_i}.N_{n_i}, PG_{s_i}.N_{n_j}, \Delta_{s_i}, \Delta_{s_j}, \boldsymbol{\delta})$
7 add F to PG
8 **optimise**(PG)
9 **end**

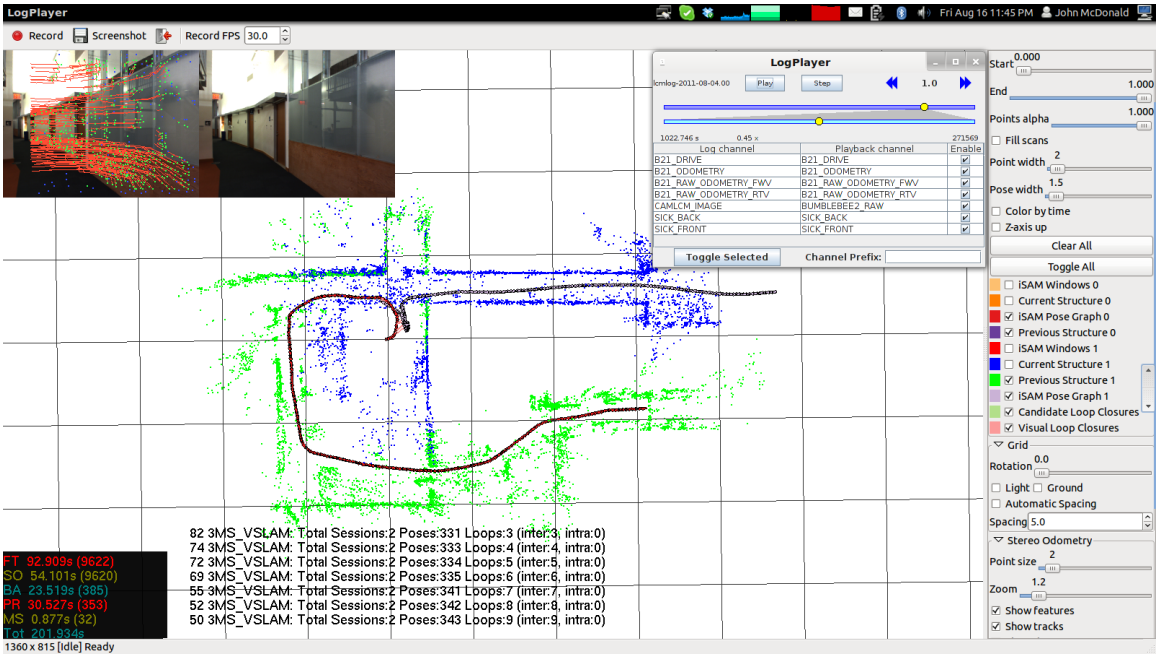


Figure 4.10: Screenshot of the multi-session visual SLAM system developed as part of this thesis.

session shown in blue. The corresponding trajectories are also shown in red and purple, respectively. Loop closures between the two sessions are displayed by the pink lines in the region where the two trajectories meet. The top left hand corner of the window shows the current frame, where the features from the GPU KLT module are overlaid on the left frame. The lower left corner provides timings for each of the components of the system including: the feature tracker (FT), stereo odometry (SO), windowed bundle adjustment (WBA), place recognition (PR), and multi-session pose graph optimisation (MS). The right hand side shown a number of controls provided by the viewer that allow different parts of the visualisation to be reconfigured.

The figure also shows an LCM log player window which is part of the LCM suite of utilities, and allows playback of previously captured LCM log files. The log file shown here is part of the dataset collect during this thesis and described further in the next chapter. As can be seen from the figure, the log file being processed includes a number of separate data channels including odometry information from the B21 robot platform, a raw stereo image channel captured using a Point Grey Bumblebee stereo camera, and two SICK lidar scanners. In this instance the system subscribes to the BUMBLEBEE2_RAW channel to receive the stereo image data feed. Due to the modularity provided by LCM, from the system's perspective such a log playback is indistinguishable from a live camera feed and as such provides an excellent framework for experimental evaluation and system development.

Fig. 4.12 shows a second graphical tool provided by LCM, called the *Sheriff*, which provides an efficient interface for managing and executing multiple cooperating processes. The figure shows the setup used in Chapter 5 for coordinating the multi-session mapping experiments. Finally, Fig. 4.13 shows the stereo processing pipeline developed using the Camunits image acquisition library (`https://code.google.com/p/camunits/`).

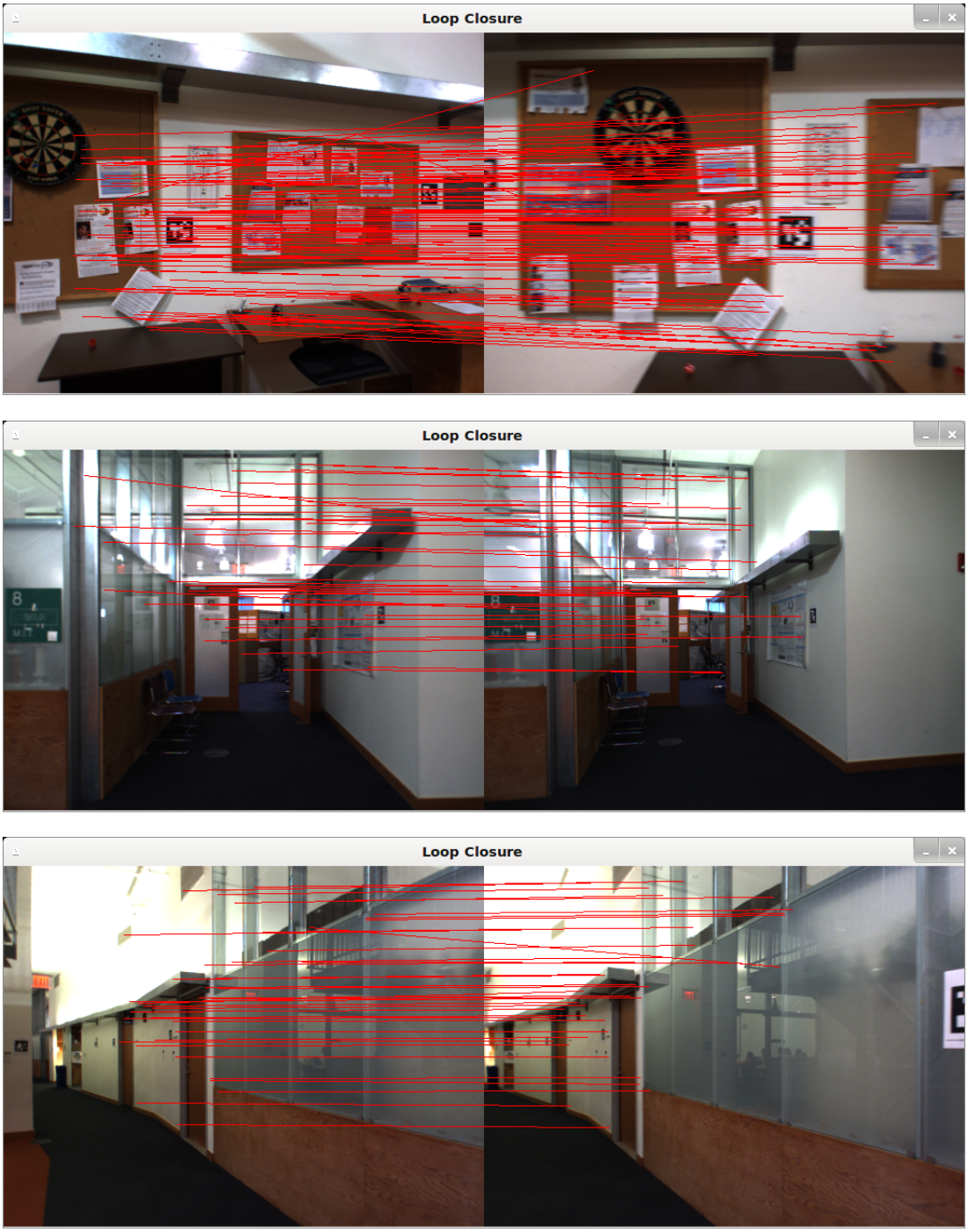


Figure 4.11: Example multi-session loop closures taken from the mapping sessions shown in Fig. 4.10

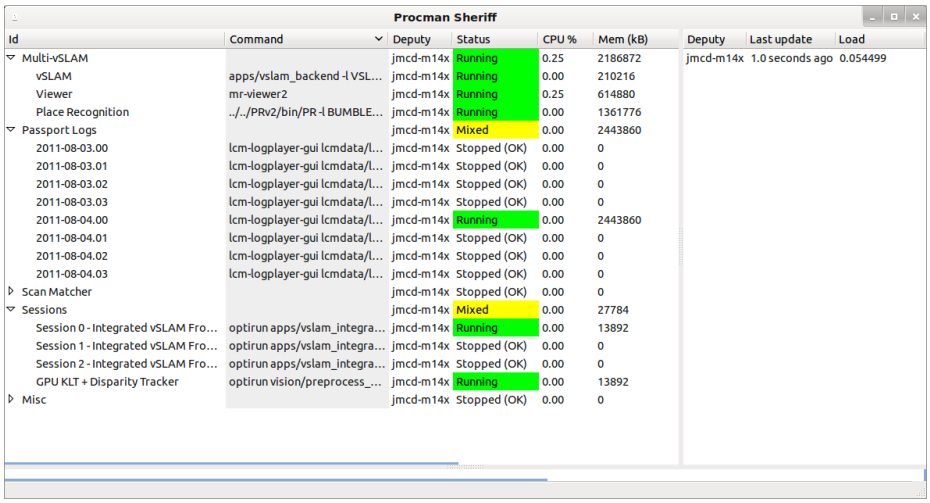


Figure 4.12: Screenshot of the *LCM Sheriff* utility for coordination and execution of system modules

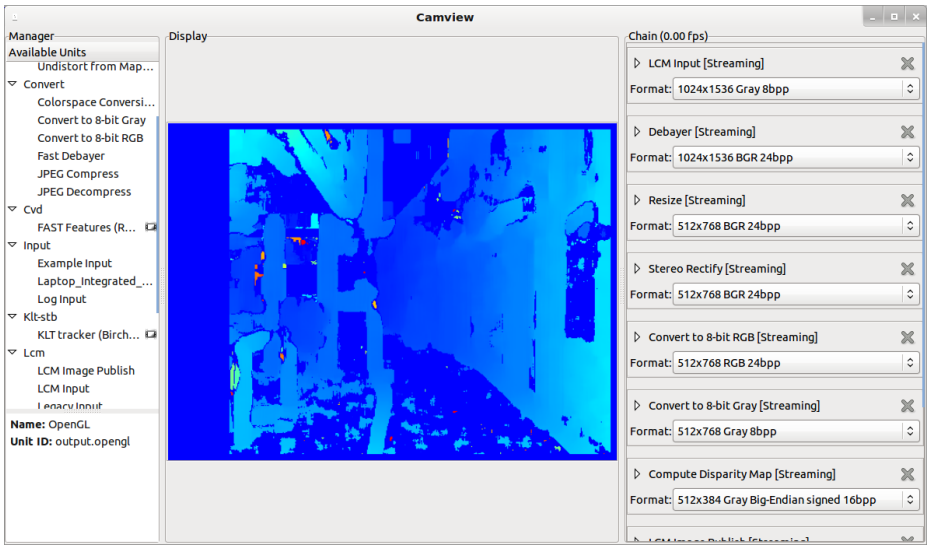


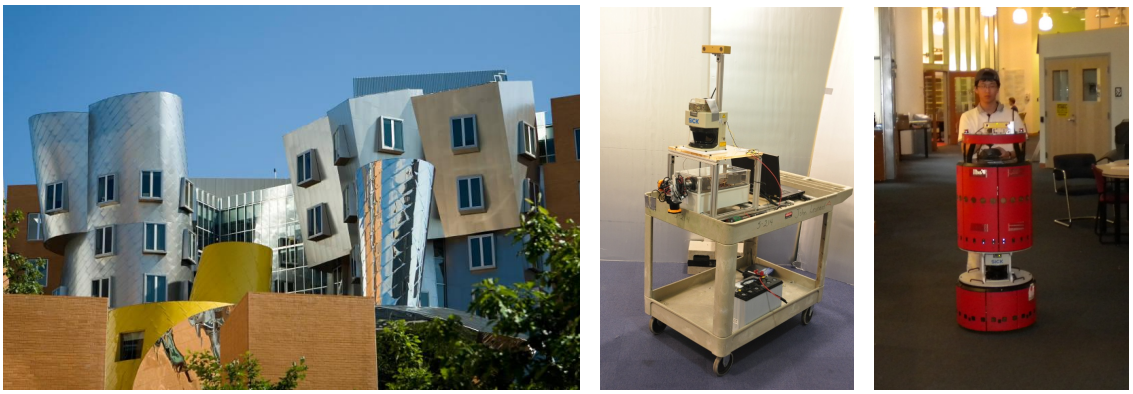
Figure 4.13: Screenshot of the *Camview* based stereo processing pipeline

CHAPTER 5

Experimental Results

In this chapter we provide both a qualitative and quantitative assessment of the performance of the system described in Chapter 4 using a number of different datasets. In Sections 5.1 and 5.2, we assess the performance of our system in a variety of scenarios, using a dataset that was collected at the Ray and Maria Stata Center at MIT over a period of months (see Fig. 5.1). This building is known for its irregular architecture and provides a good testing ground for visual SLAM techniques in general.

The dataset includes indoor, outdoor, and mixed sequences captured using robotic and manually wheeled platforms and a handheld camera with full 6-DOF movement (e.g. ascending and descending stairs, etc.). All image sequences were captured using a Point Grey Bumblebee colour stereo camera with a baseline of 11.9cm and where both lenses had a focal length of 3.8mm. The sensor resolution of each camera was 1024×768 pixels which we subsampled to 512×384 pixels prior to processing. As can be seen in the figure the wheeled platforms also included horizontally mounted 2D lidar scanners. Although we do not use the lidar sensors in our system, the



(a) (b) (c)

Figure 5.1: Experiments carried out in Sections 5.1 and 5.2 are performed using a dataset collected in (a) the Ray and Maria Stata Center, MIT over a period of months. The images in the dataset were acquired using a colour Point Grey Bumblebee stereo camera and were captured using (b) manually wheeled, (c) robot mounted, and handheld configurations.

accompanying laser data allows us to compare the performance of our technique to that of a laser-based scan matcher in restricted wheeled platform scenarios (see Section 5.1 for details). Note all processing is performed in full 6-DOF mode without any assumptions on the structure of the scene *e.g.* using a ground-plane model.

The quantitative assessment is based on three separate experiments; one which assesses the system in a single-session scenario and two which assess the system in multi-session scenarios. Details of the datasets used in each of the experiments are provided in Table 5.1. For each of the experiments processing was carried out on an Intel® Core™ i7 940 2.93GHz based machine with 8GB of RAM and an nVidia® GeForce® 9800 GT graphics card.

To permit a comparison of the performance of the system developed in this thesis with other systems, Sections 5.3 and 5.4 provide experimental results using two publicly available robotics datasets. Section 5.3 provides both single and multi-session results using the Oxford New College Dataset [129]. Section 5.4 provides quantitative results on the accuracy of the system using a separate Stata Center dataset [37] that provides ground truth information for the trajectory of the robot.

	Experiment 1	Experiment 2	Experiment 3
Indoor/Outdoor	Indoor	Indoor	Outdoor
Num. sessions	1	4	3
Seq. length	20.3 min	45 min	4.7 min
Num. poses	883	1562	1406
Intra-session	112	4	0
Inter-session	0	260	13

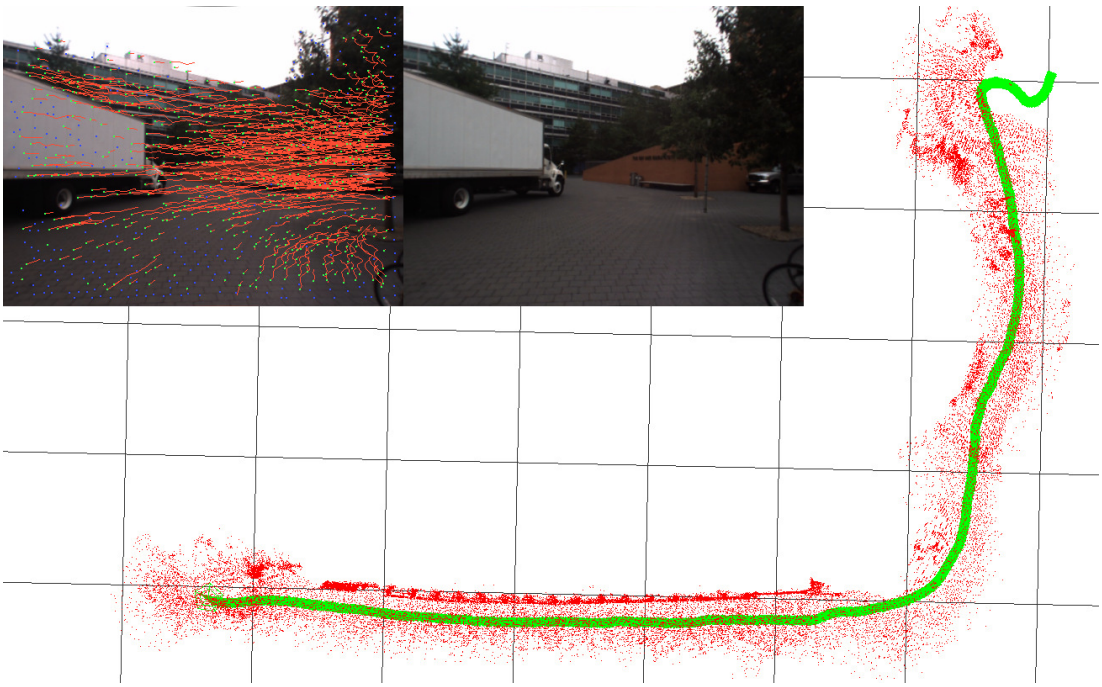
Table 5.1: Description of experimental datasets.

5.1 Visual SLAM Results

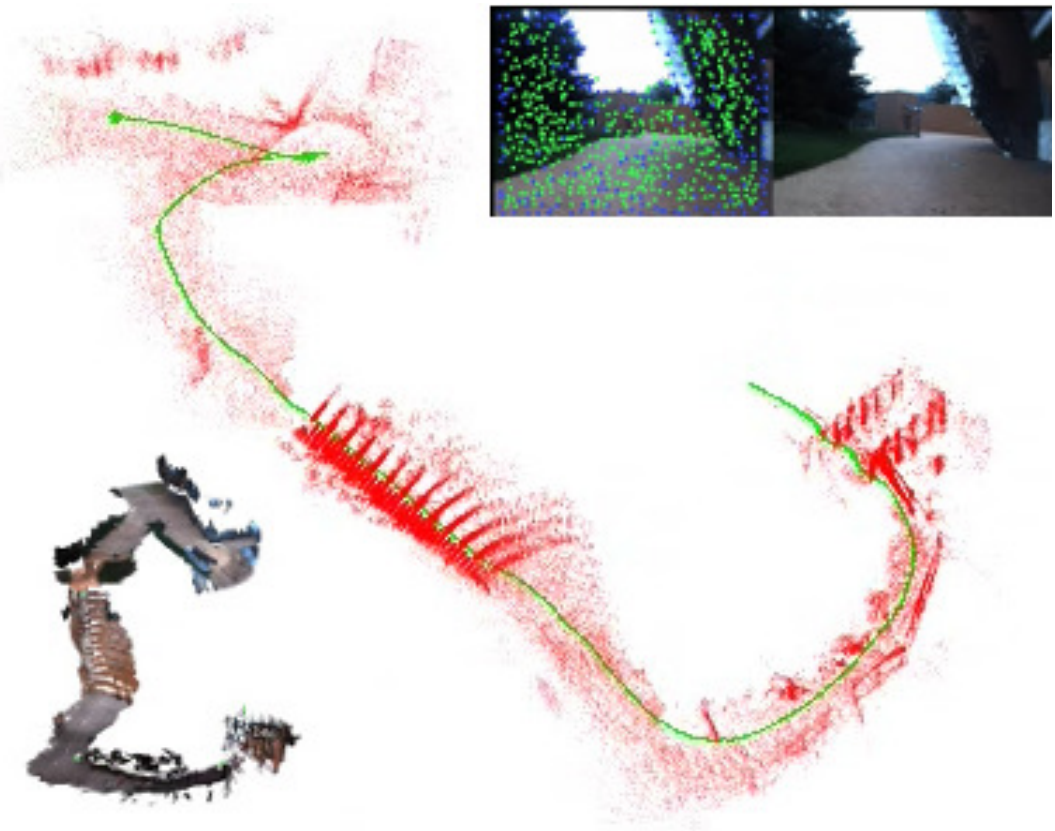
In this section we provide results from a number of single session SLAM experiments. To do this we have applied the system in single session mode (i.e. only running a single front-end) across a variety of sequences from the Stata Center dataset. The results show that the system is capable of operating over extended sequences in both indoor, outdoor and mixed environments with full 6-DOF motion.

For example, two feature-based maps for outdoor sequences are shown in Fig. 5.2. Here, for 5.2(a), the underlying grid is at a scale of 10m, where the trajectory is approximately 100m in length. An example image from the sequence is shown in the inset with the GPU KLT feature tracks overlaid on the left frame. Fig. 5.2(b) shows a similar scale sequence that includes full 6-DOF motion, where the user has carried a handheld camera up a stairs.

To evaluate the accuracy of the front-end we compare its trajectory estimate to that of a scan-matching algorithm applied to the corresponding lidar data. In the absence of loop closures we have found the system to have drift of approximately 1%-3% in position during level motion (i.e. without changes in pitch angle). To demonstrate this, Fig. 5.3 shows two maps with two trajectories, both taken from the same sequence. The black contour shows the 2D lidar scan-matcher based map. The scan-matcher's estimated pose is shown by the blue trajectory, which can be seen more clearly in the lower right-hand inset. The distance between grid lines in the figure is 2m. From the figure the horizontal displacement of the final poses is



(a) Wheeled sequence involving principally 3-DOF motion.



(b) Handheld sequence over two levels including ascending amphitheatre steps

Figure 5.2: Single session visual SLAM processing with full 6-DOF motion.

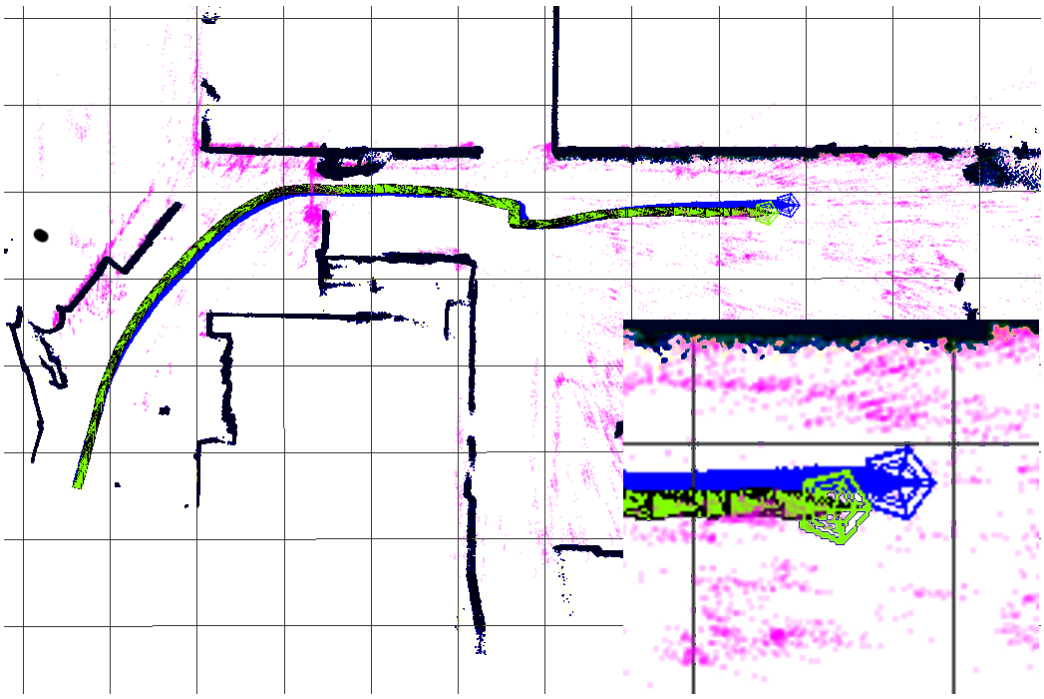
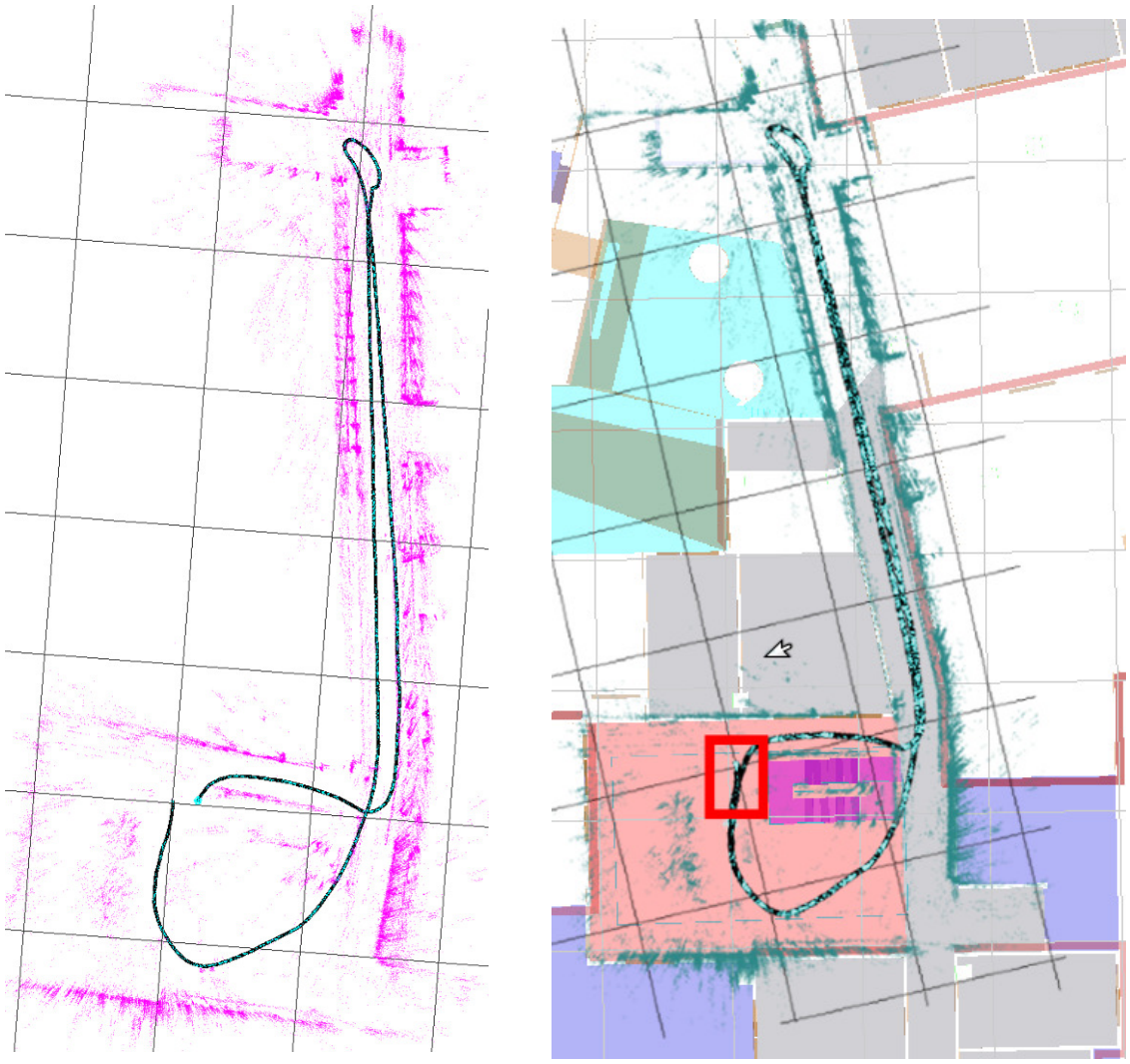


Figure 5.3: Comparison of drift in single session visual SLAM (green poses) against 2D lidar scan-matcher (blue poses) over a 20m trajectory. The grid scale is 2m. The inset in the lower right corner shows a zoomed view of the final poses where the discrepancy is approximately 60cm.

approximately 60cm with a total trajectory of approximately 20m.

An example of the accumulated error in position due to drift is shown in Fig. 5.4. Here the dataset consists of an image sequence taken over an indoor area within in the Stata Center. The grid is at a scale of 5m with the sequence taken by travelling on a large loop over a space of approximately 35m×15m. The image at the top shows the result of the motion estimate in the absence of a loop closure. The majority of the drift in this example is due to the tight turn approximately two-thirds of the way through the sequence, where the divergence between each traversal of the hallway can be seen.

The right-hand figure shows the result of the correction applied to the pose graph due to a sequence of loop closures occurring at the area highlighted by the red box. Here it can seen that the sections of the pose graph showing the traversals of the hallway are much more coincident and that the misalignment in corresponding portions



(a) (b)

Figure 5.4: Single-session dataset containing a large loop. Here the grid scale is at 5m. (a) Map and pose graph prior to loop closure showing drift in position and structure. (b) Map and pose graph showing correction in the position and structure due to a series of loop closures in the area shown by the red square. Background image shows ground truth CAD floor-plans of the environment.

Figure 5.5: Textured version of the single-session dataset containing a large loop shown in Fig. 5.4(b).

of the map is reduced considerably. The figure also shows the accuracy of the map relative to the ground truth CAD floor-plan.

As mentioned in the previous section, in order to measure the computational requirements of the system we evaluated the processing times of each component based on three sets of sequences, each drawn from the Stata dataset. The output map and trajectory for Experiment 1: the single session indoor sequence (see Table 5.1) is shown in Fig. 5.6. In total the input sequence was 20.3 minutes in length (i.e. 24360 frames). Also shown in the table is the final number of poses in the pose graph, and the number of intra-session loop closures. As can be seen from the results, the pose decimation scheme described in Section 4.2.2 significantly reduces the number of poses, in this case to 883. It is important to point out that the ratio of poses to frames in this example is low due to the fact that the camera is mounted on a B21 robot which is moving slowly through the environment. As will be seen in the next section, when the camera moves quickly such as with handheld sequences this ratio is approximately an order of magnitude higher.

Table 5.2 provides summaries of the run-times for the principal modules of the system, where the first two columns provide the mean and variance for Experiment 1. Note that the times for feature tracking and stereo odometry modules correspond to the computational cost for each frame of the input sequence. However given that each of the iterations of the windowed bundle adjustment and place recognition modules only occur on each new keyframe (i.e. pose), the times provided are per keyframe. Finally since the pose graph updates only occur when the front-end window reaches 15 poses or whenever a loop closure occurs, the times shown in this row are per window. Given that each of the modules runs as a separate process, with communication handled via LCM, each task is handled by a separate core of the CPU.

Figures 5.7-5.8 provide plots of the computation time for each iteration of the windowed bundle adjustment, place recognition, and pose graph iSAM, respectively.

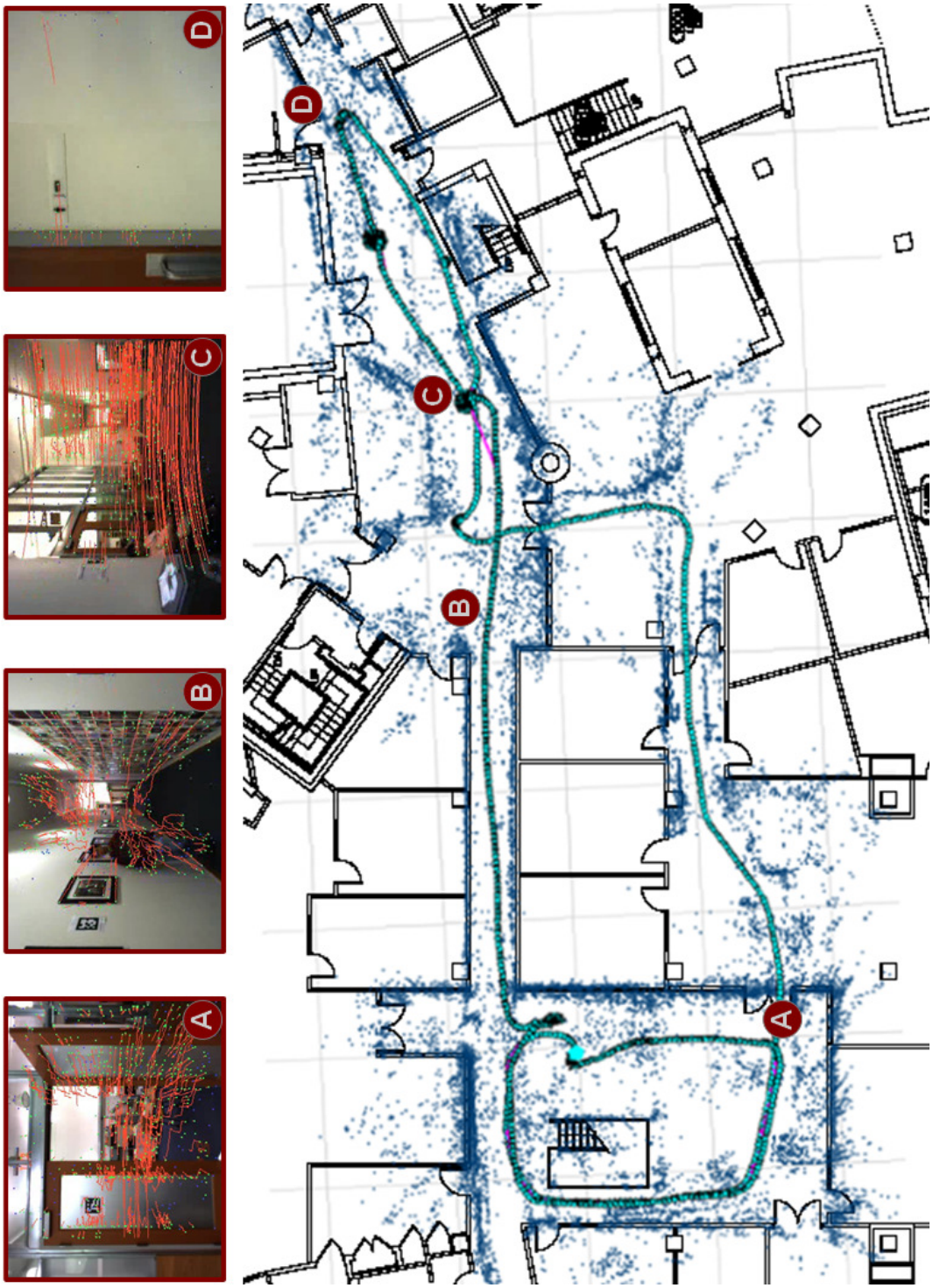


Figure 5.6: Top-down orthographic projection of the results for the experiment 1 single session dataset described in Table 5.1. Here the map is overlaid on an architectural floor-plan for comparison with ground-truth. The reference grid is shown at a scale of 5m. Images A–D displayed above the map show sample frames from the sequence with their approximate location highlighted on the map. Further details can be found in Section 5.1.

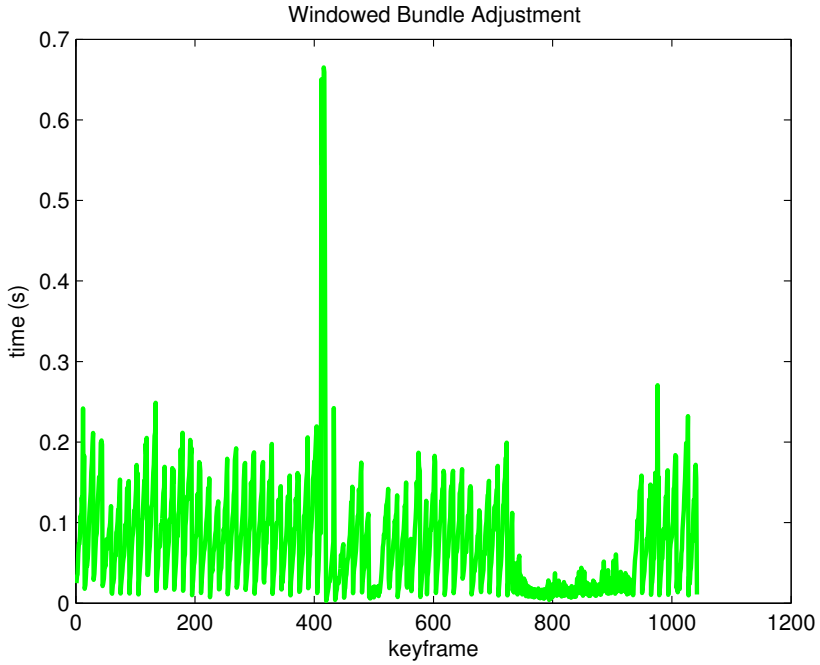
	Experiment 1 Single Session Indoor		Experiment 2 Multi-session Indoor		Experiment 3 Multi-session Outdoor	
	Mean(s)	Variance(s)	Mean(s)	Variance(s)	Mean(s)	Variance(s)
Feat. track.	0.0309	3.29×10^{-5}	0.0315	4.90×10^{-5}	0.0374	4.8×10^{-5}
Stereo odom.	0.0084	1.61×10^{-4}	0.0083	1.62×10^{-4}	0.0021	5×10^{-5}
WBA	0.0755	0.0048	0.0665	0.0029	0.0611	0.0026
Place recog.	0.1703	0.0832	0.1794	0.0792	0.1049	0.0148
iSAM	0.0996	0.0040	0.1960	0.0242	0.1242	0.0170

Table 5.2: Mean runtimes and variances for each iteration of each module of the system for each of the experiments reported in the thesis. Feature tracking and stereo odometry are executed for each frame of the input sequence. Windowed bundle adjustment (WBA) and place recognition are executed for each pose. Incremental updates to the pose-graph are computed via iSAM once for each window.

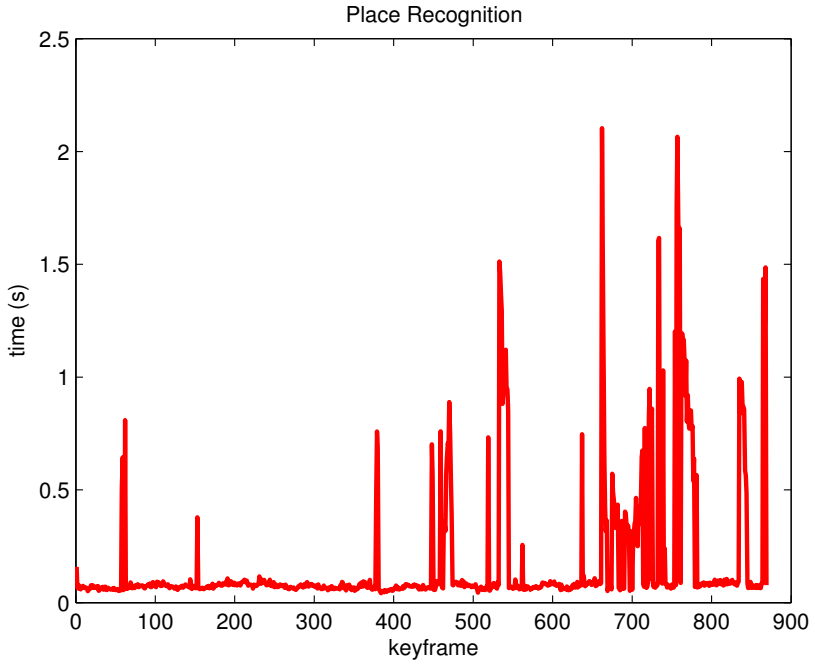
As can be seen from Fig. 5.8 the total number of updates to the pose graph is approximately 160 with a total of 883 poses in the final pose graph. Also, from this graph it is possible to see, (i) the increase in computation time as the size of the pose graph grows, and, (ii) the impact due to loop closures.

Figures 5.9(a) and 5.9(b) provide plots of the cumulative computation time as a function of processed input sequence time for each of the modules in the front-end and back-end respectively. Given that each of these modules runs in parallel, each on a separate core of the CPU, it is clear from this graph that the overall system is capable of running in real-time.

Finally, in order to assess the drift of the system, Figure 5.10 shows a number of visualisations of the variation in the estimated height of the sensor over the single session dataset. Given that there is no ground plane estimation, the original map is output in the frame of reference of the initial camera pose. Hence to create these figures we first computed the plane of best fit to the poses and centered the data around the resulting Z direction (i.e. a mean Z value of 0m). Figure 5.10(a) shows a plot of the resulting data where the X and Y position of the sensor has been manually aligned to the building floorplan. The alignment process only varied the orientation and translation in the plane, and the overall scale. Figure 5.10(b), shows a plot



(a)



(b)

Figure 5.7: Processing times per keyframe for Experiment 1: single session indoor sequence. Per keyframe processing times for (a) iSAM based windowed bundle adjustment and (b) place recognition.

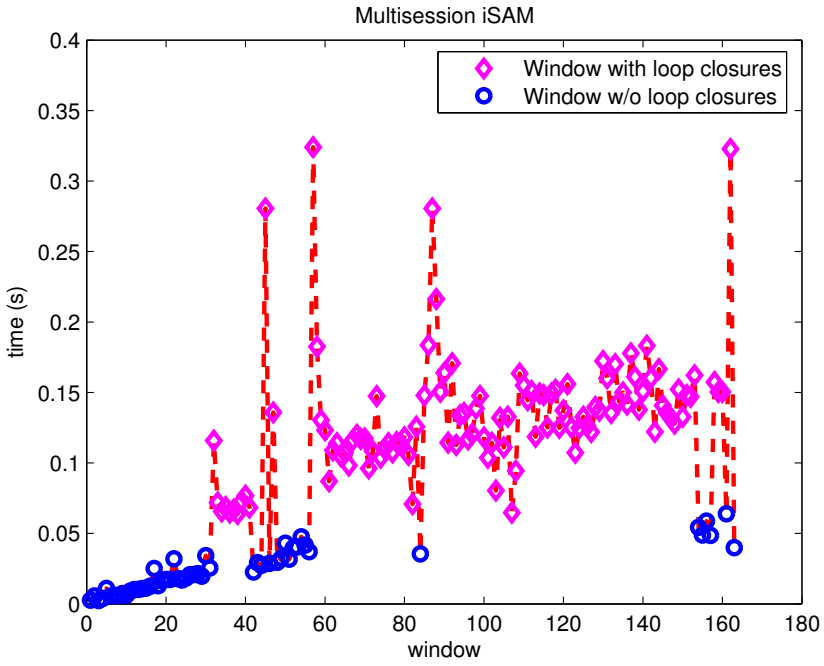
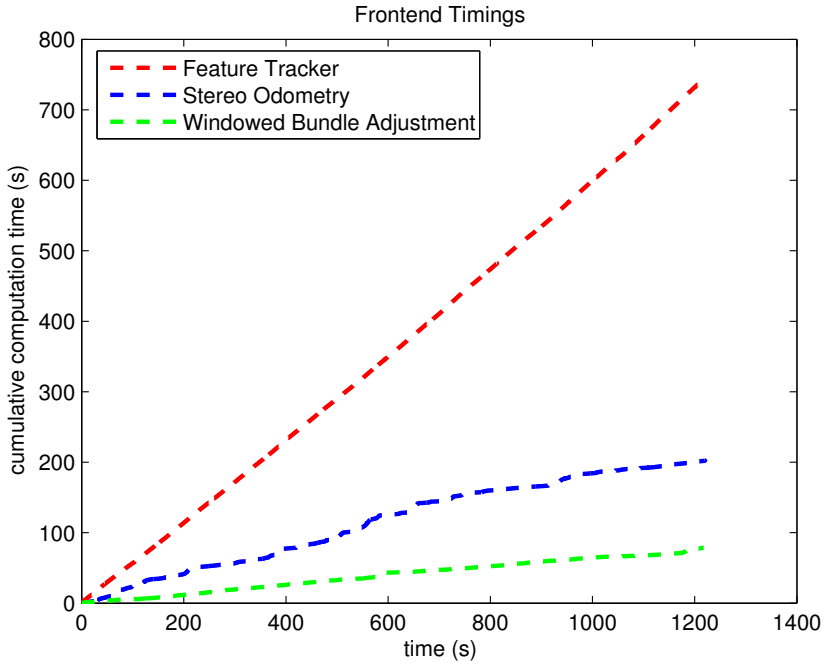
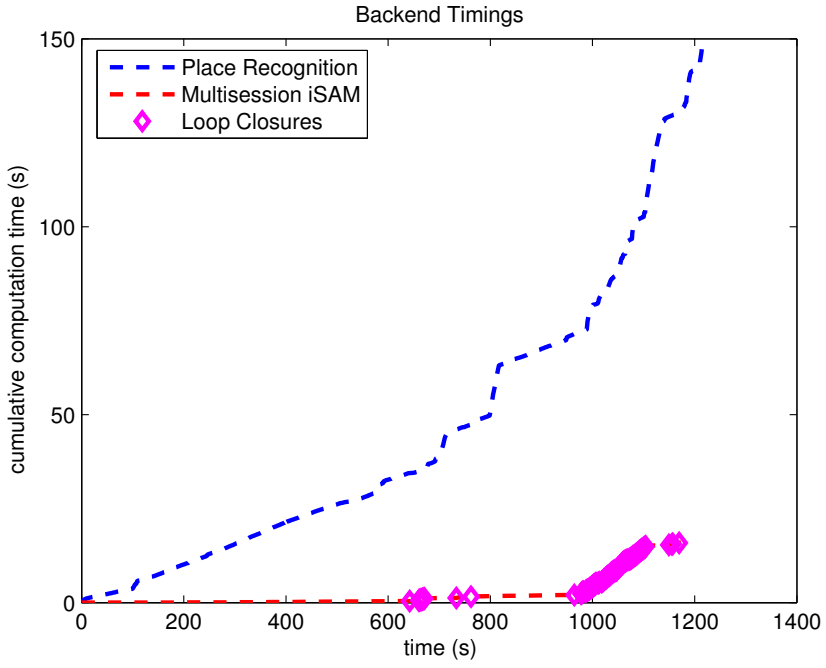


Figure 5.8: Experiment 1 processing times for the iSAM based pose graph optimisation. Note that the timings are per window where the purple diamond data points and blue circle data points distinguish between windows with and without loop closures, respectively.

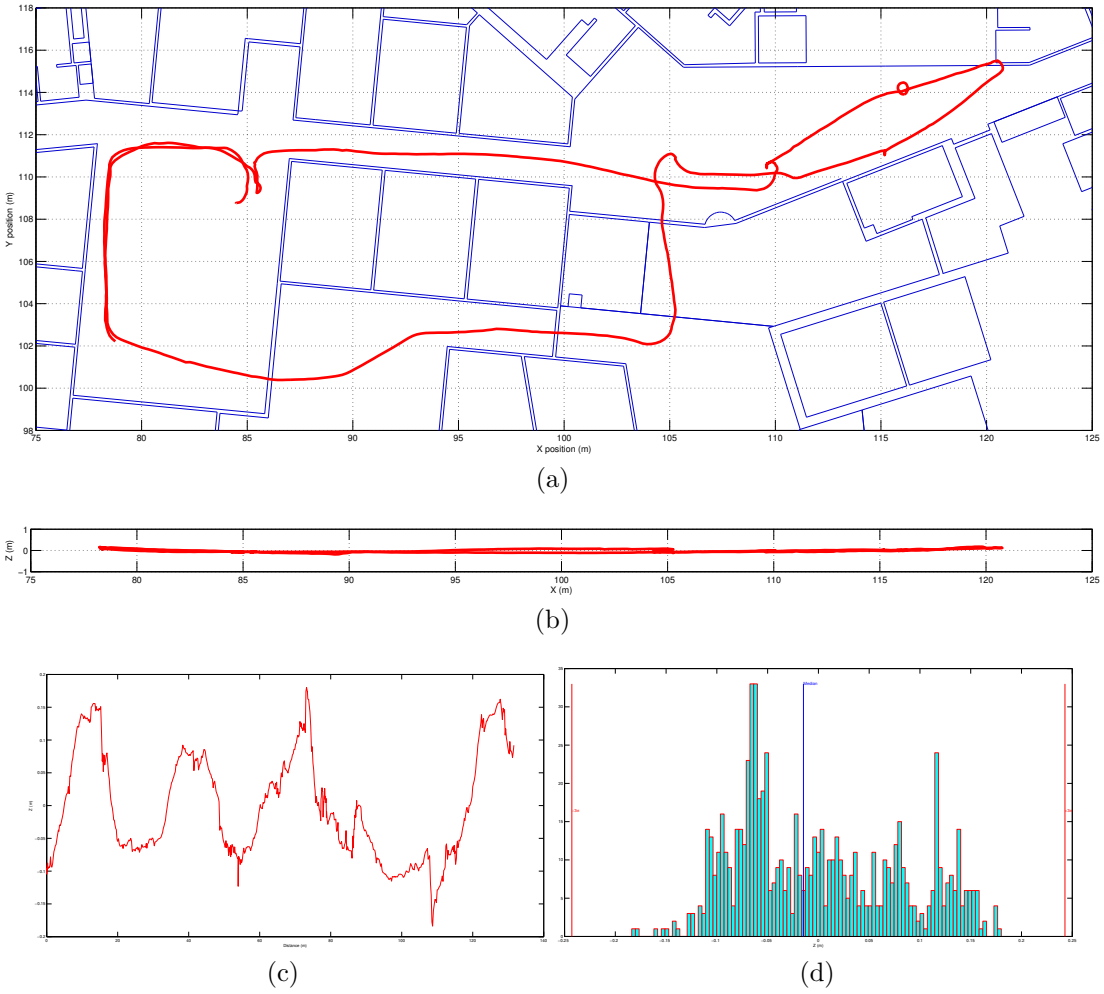


(a)



(b)

Figure 5.9: Cumulative processing times for Experiment 1: single session indoor sequence. These two plots show the total processing time for (a) the front-end, and, (b) the back-end, both as a function of the total input sequence length in seconds.



(a)

(b)

(c) (d)

Figure 5.10: Single session Stata Center elevation estimate. Figure (a) shows a plan view of the resulting trajectory. Figure (b) shows the variation in Z as a function of X position. Figure (c) shows the variation in Z as a function of distance travelled. Finally, Figure (d) shows a histogram of the Z values.

of the X and Z position where the plot is aligned to Figure 5.10(a). Note that in Figure 5.10(b) the scale on the Z-axis is from $-1m$ to $+1m$, and hence as can be seen from the figure there is minimal drift in the Z-direction. Figure 5.10(c) shows a plot of the Z position as a function the distance travelled, whereas Figure 5.10(d) provides a histogram of the distribution of the Z-values. Here the total distance travelled is 131.5m, where the median Z-value is $-0.0147m$, and the standard deviation of the Z-value is 0.0809m.

5.1.1 Odometry Failure

Although the odometry system has shown to be robust over maps of the order of hundreds of meters, two failure modes for the system are due to tracker failure during (i) high-speed motion, and (ii) low-texture or low contrast environments, which can also cause errors whereby the disparity estimation fails over a large set of features. In the current system we address this through reinitialising the tracker and inserting a new pose where the motion relative to the previous pose is set to zero with large covariance. Hence we keep the two sections of the pose graph (i.e. at either side of the failure) topologically connected whilst capturing the high degree of uncertainty between them. This is done with the intention that future loop closures between the sections will provide adequate constraints to correct for this uncertainty.

A frame where such an odometry failure occurred in Experiment 1 is shown in the right-most frame at the top of Fig. 5.6 (i.e. image D). Here lack of texture in the environment results in a low feature count and hence an inability to estimate the camera's motion. Fig. 5.6 also highlights the location of the frame in the map. At the point in the sequence where this failure occurs the map diverges, however a subsequent loop closure close to point C in the map corrects for this drift. As can be seen from comparison, the final estimated structure is in close agreement with the ground truth floor plan.

We note that the above approach can be avoided for short term tracking failures by incorporating inertial sensors [58]. For longer tracking failures, an alternative approach that we are currently investigating is the possibility of using multi-session SLAM as a solution to this problem, whereby odometry failure results in the creation of a new session with a weak prior on the initial position. This disjoint session is treated the same as any other session. When a new encounter does occur, the session can be reconnected to the global pose graph.

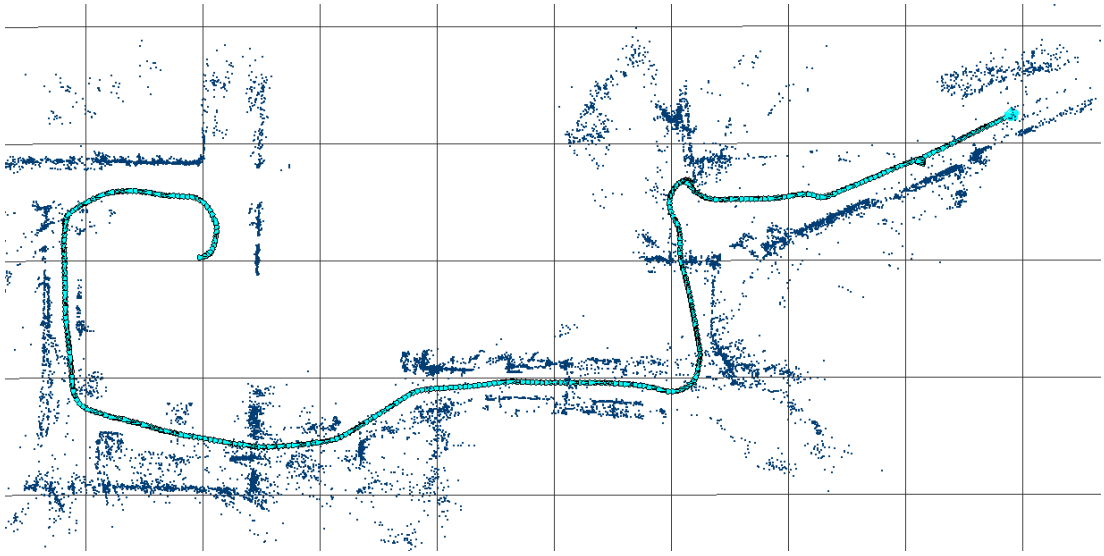
5.2 Multi-session Visual SLAM Results

To evaluate the multi-session performance of the system we tested it on the Experiments 2 and 3 datasets detailed in Table 5.1. The rationale for choosing the datasets was that the Experiment 2 dataset contains, as a subset, the single session dataset from Experiment 1, and therefore allows a direct comparison to be made between the system's single- and multi-session operation. The sequences used in Experiment 3 differ from the other datasets in that they were captured using a handheld Bumblebee camera in an outdoor environment and contain sub-sequences where the user ascends and descends stairs. Hence Experiment 3 involves a much larger and more erratic range of motions in all 6-DOF.

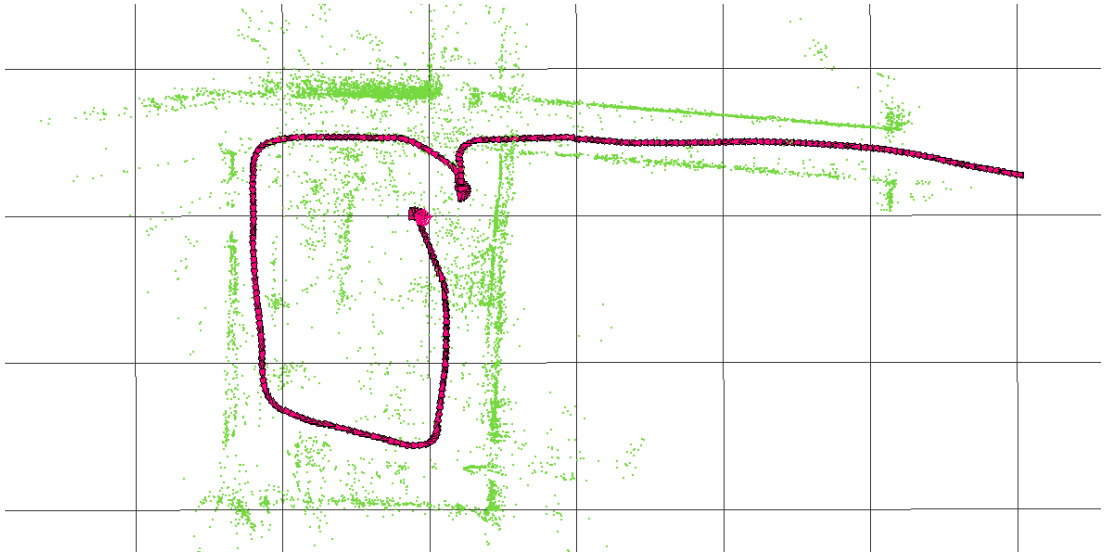
Figs. 5.11–5.14 provide a number of different views of the output of the system for the Experiment 2 dataset. Figs. 5.11 – 5.12 show the results for the individual sessions including trajectories and maps. Fig. 5.13 shows the trajectories where the elevation increases with time and the purple links between the trajectories correspond to the intra and inter-session loop closures. Finally, Fig. 5.14 shows the complete multi-session pose-graph and map including all four sessions aligned to the building floorplan.

As reported in Table 5.1, in this experiment the input dataset consisted of 4 separate sessions, which, when combined, corresponded to a total of 45 minutes of video at 20 fps. The final multi-session pose graph contains 1562 poses and spans an area of approximately 75m $\times$ 25m. Comparing the output to that of Experiment 1, the ratio of the number of poses in the final trajectory to the length of the input video sequence is of the same order. Timings for each of the modules of the system for this dataset are given in columns 3 and 4 of Table 5.2.

Figs. 5.15–5.17 provides a set of plots for the module execution times for Experiment 2 equivalent to those shown for Experiment 1 in the previous section (see Figs. 5.7–5.9). As can be seen from the multi-session results, each of the modules

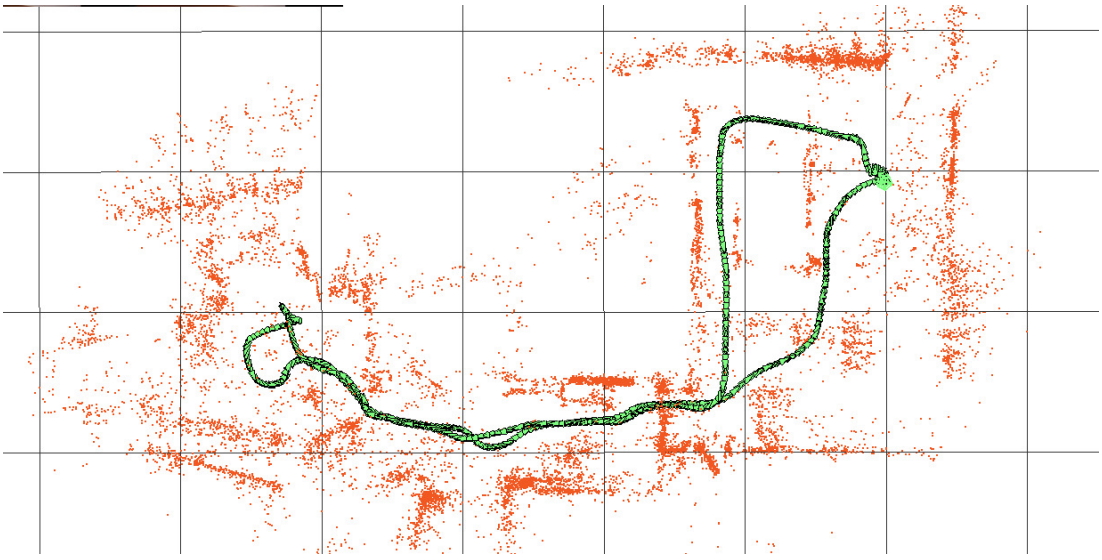


(a) Session 1

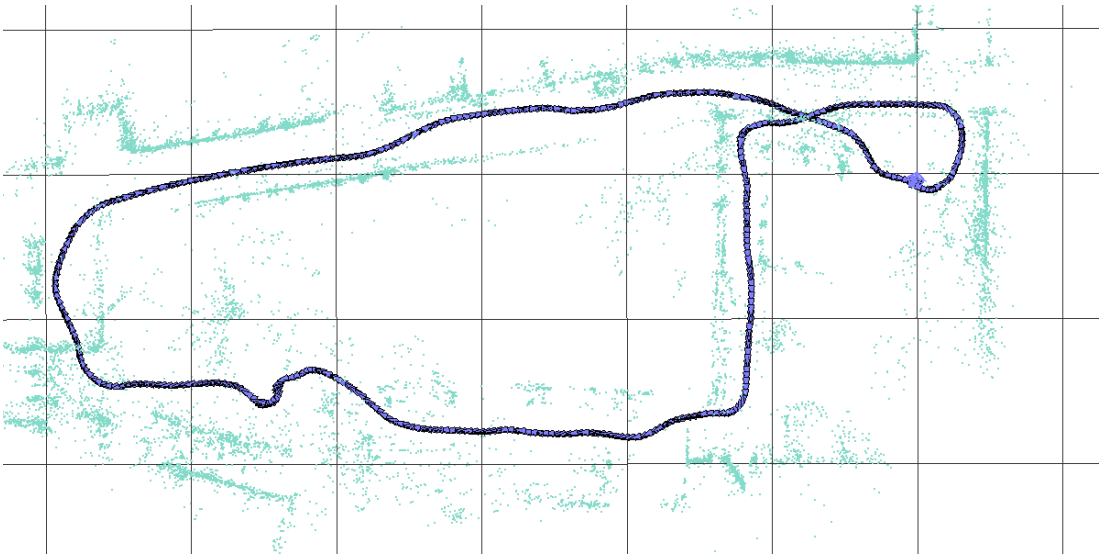


(b) Session 2

Figure 5.11: Experiment 2: Sessions 1 & 2 of Stata Center multi-session second floor dataset. The underlying grid is set at a 5m scale in all figures. See Section 5.2 for further details.



(a) Session 3



(b) Session 4

Figure 5.12: Experiment 2: Sessions 3 & 4 of Stata Center multi-session second floor dataset. The underlying grid is set at a 5m scale in all figures. See Section 5.2 for further details.

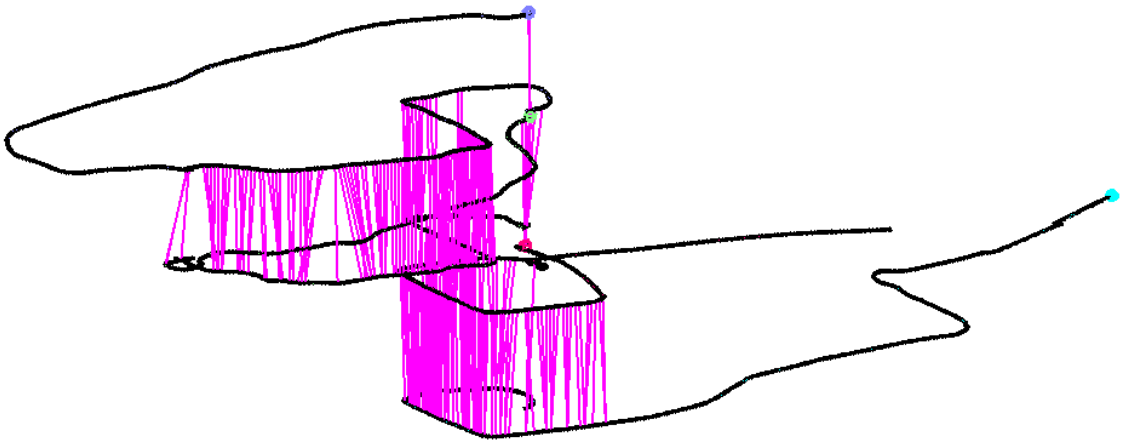


Figure 5.13: Experiment 2: Detected loops within and between pose graphs in the Stata Center multi-session second floor dataset. Here the Z-axis increases with time with loop closures shown by the purple links between the pose graphs. See Section 5.2 for further details.

except for the multi-session iSAM module have similar mean execution times to the single-session operation. The reason for the difference in iSAM is due to the fact that the complexity of the optimisation is a function of the number of poses in the pose graph.

Fig. 5.18 shows the estimated map and trajectory for the Experiment 3 dataset, where again the grid is at a scale of 5m. Fig. 5.18(a) shows a plan view of the map, where it can be seen that the total area covered by the 3 sessions is approximately 110m $\times$ 80m. A side view of the map where the 6-DOF motion of the camera is apparent is shown in Fig. 5.18(b). Given that the sequences in this dataset are taken from a handheld camera the motion of the sensor is at much higher velocities and as a consequence the ratio of the number of poses to the number of frames processed is an order of magnitude higher than in the two previous experiments. In particular, although the total length of the image sequences is 4.7 minutes compared to the 45 minutes of the indoor multi-session experiment, the number of poses are within 10% of each other.

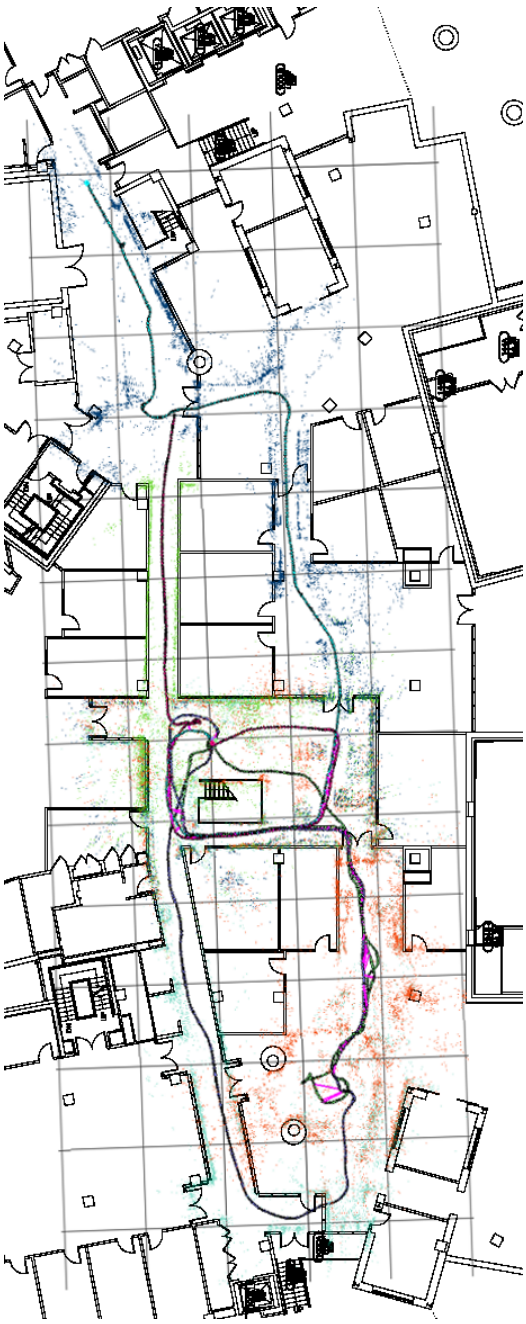
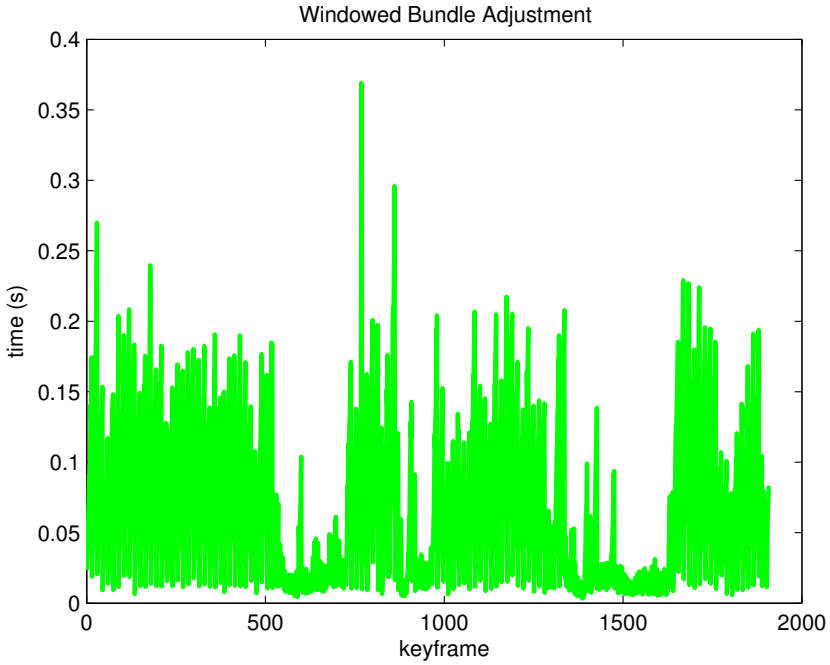
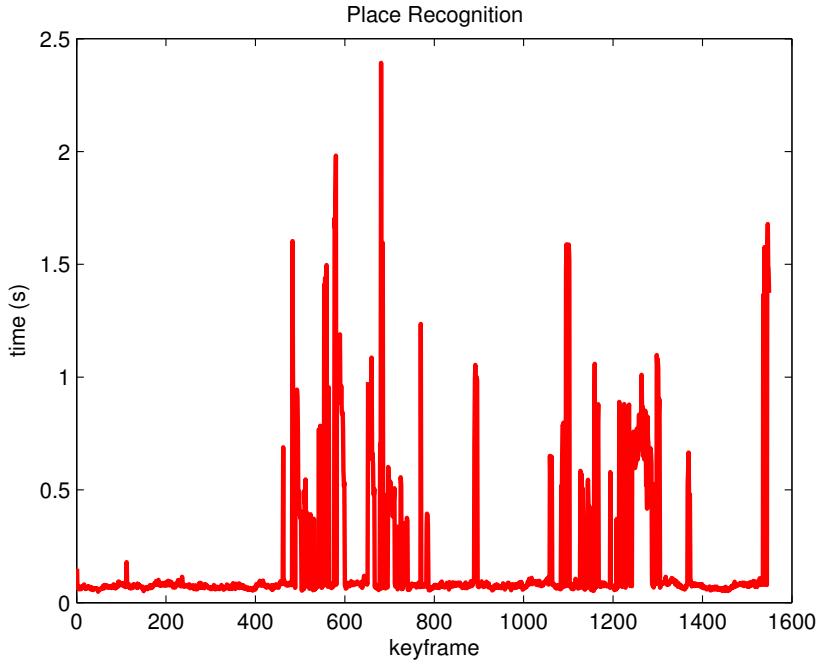


Figure 5.14: Experiment 2: Combined multi-session map and pose graph for Stata Center multi-session second floor dataset including four separate sessions captured over a 75m × 25m area. The underlying grid is set at a 5m scale in all figures. See Section 5.2 for further details.



(a)



(b)

Figure 5.15: Processing times per keyframe for Experiment 2: multi-session indoor dataset. Per keyframe processing times for (a) iSAM based windowed bundle adjustment and (b) place recognition.

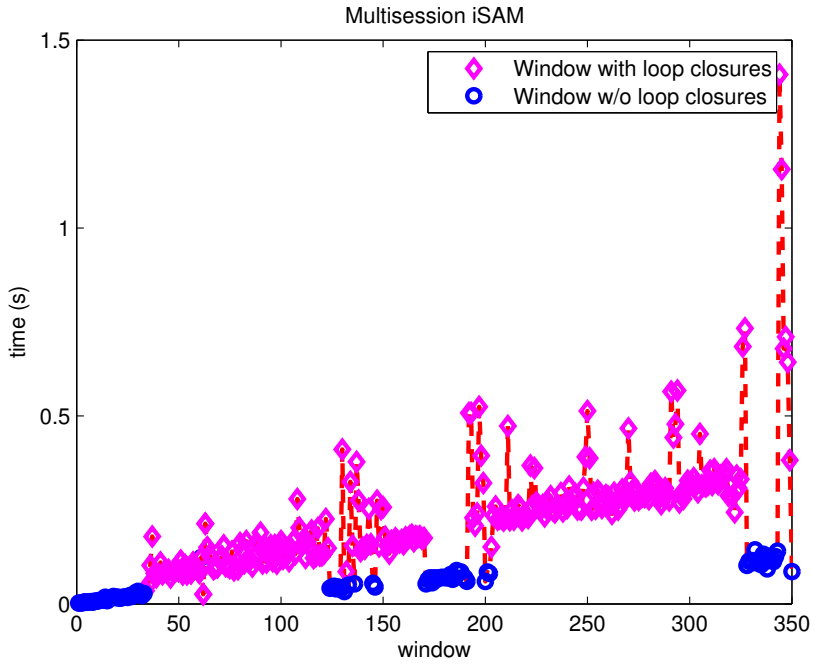
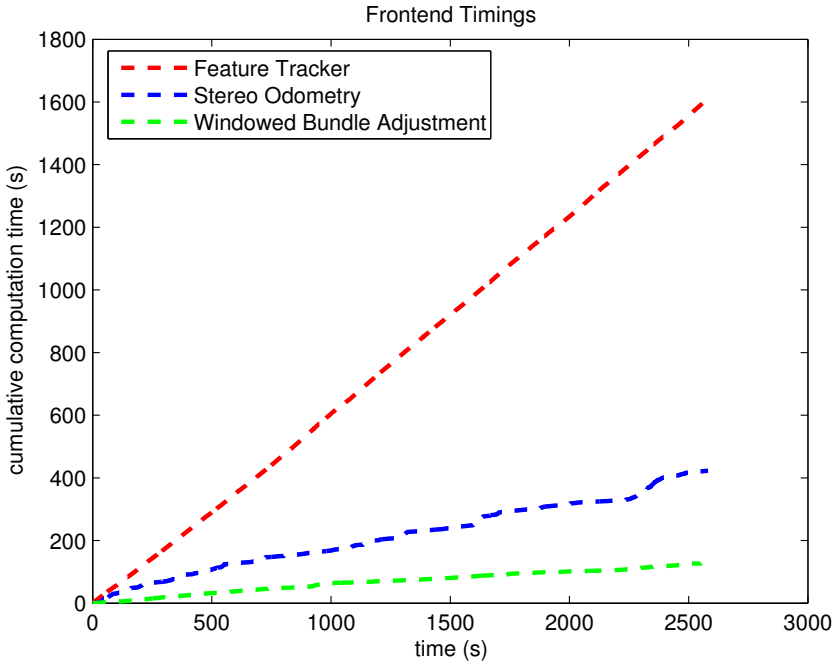
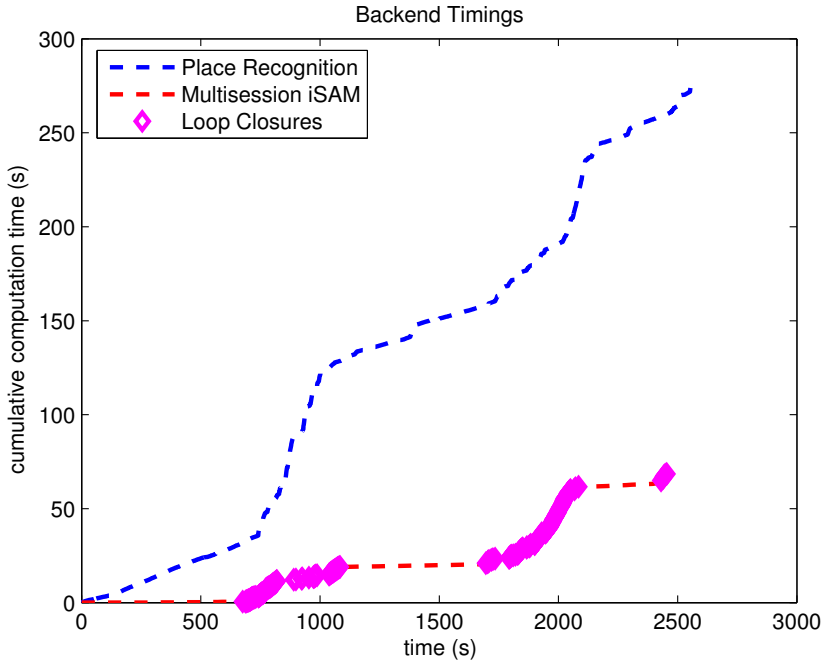


Figure 5.16: Experiment 2 processing times for the iSAM based pose graph optimisation. Note that the timings are per window where the purple diamond data points and blue circle data points distinguish between windows with and without loop closures, respectively.

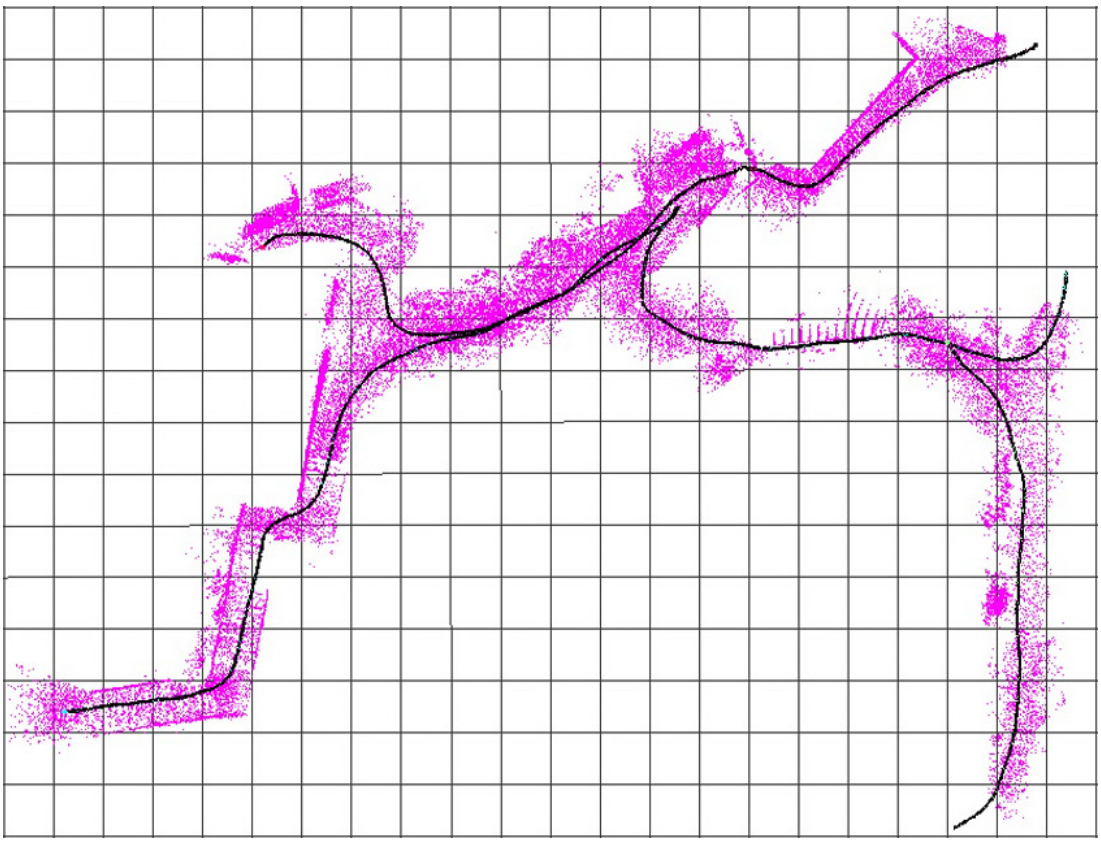


(a)

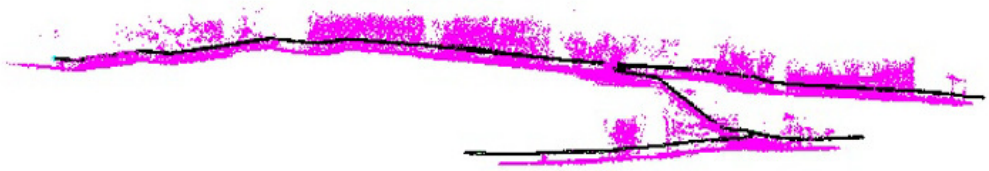


(b)

Figure 5.17: Cumulative processing times for Experiment 2: multi-session indoor dataset. These two plots show the total processing time for (a) the front-end, and, (b) the back-end, both as a function of the total input sequence length in seconds.

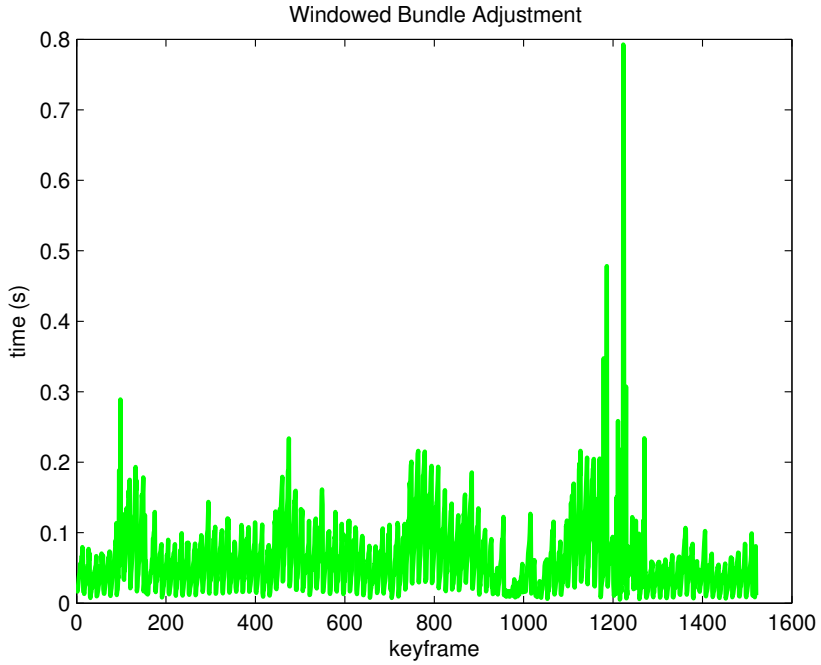


(a)

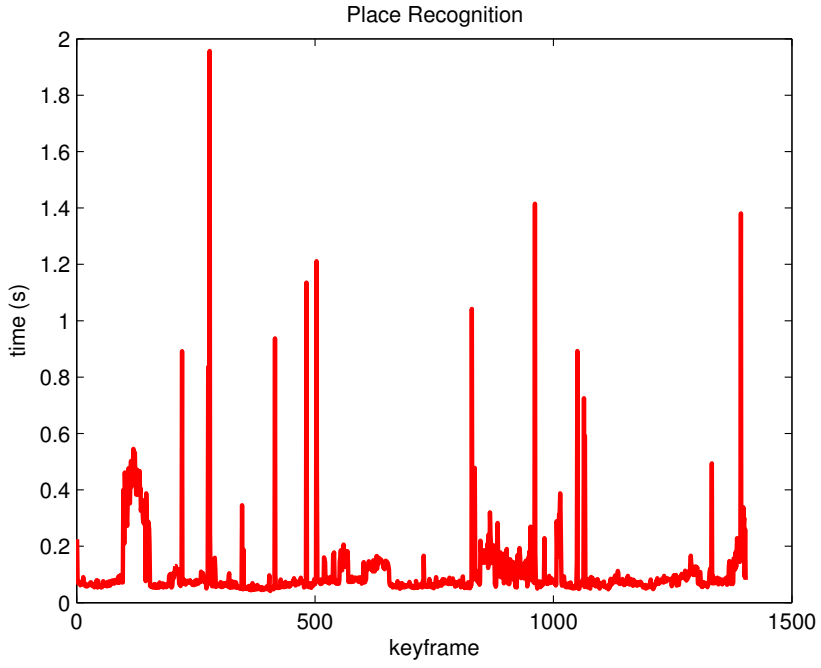


(b)

Figure 5.18: Multi-session results for outdoor datasets from Experiment 3 including 3 sessions. (a) shows a plan view where the grid is at a scale of 5m. (b) shows an orthographic side view where the full 3D structure is apparent, in particular the stairs. See Section 5.2 for further details.



(a)



(b)

Figure 5.19: Processing times per keyframe for Experiment 3: multi-session outdoor handheld dataset. Per keyframe processing times for (a) iSAM based windowed bundle adjustment and (b) place recognition.

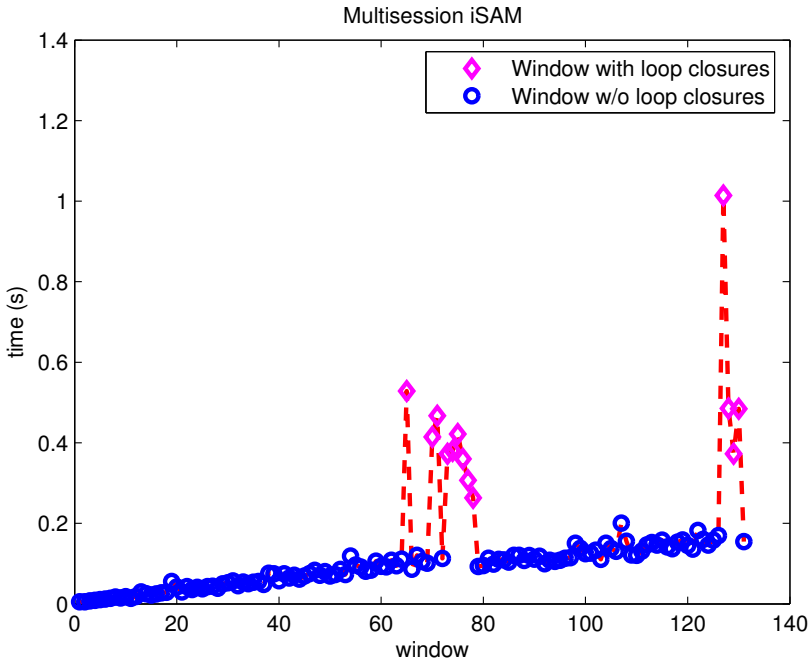
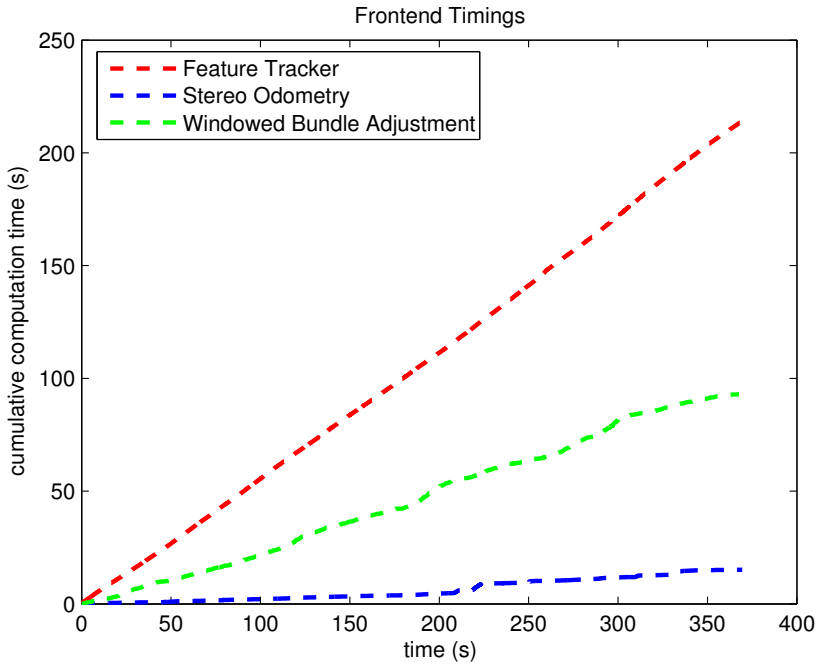
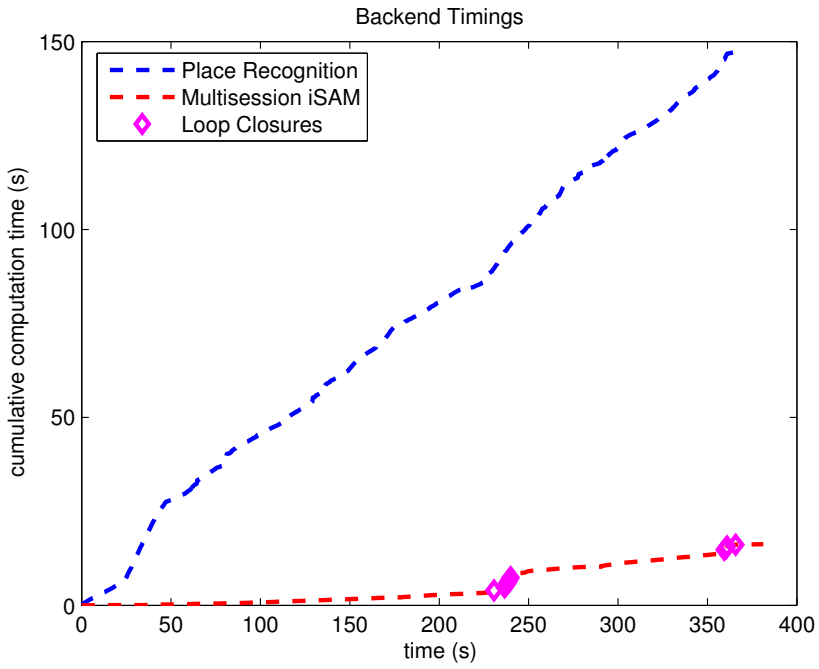


Figure 5.20: Experiment 3 processing times for the iSAM based pose graph optimisation. Note that the timings are per window where the purple diamond data points and blue circle data points distinguish between windows with and without loop closures, respectively.



(a)



(b)

Figure 5.21: Cumulative processing times for Experiment 3: multi-session outdoor handheld dataset. These two plots show the total processing time for (a) the front-end, and, (b) the back-end, both as a function of the total input sequence length in seconds.

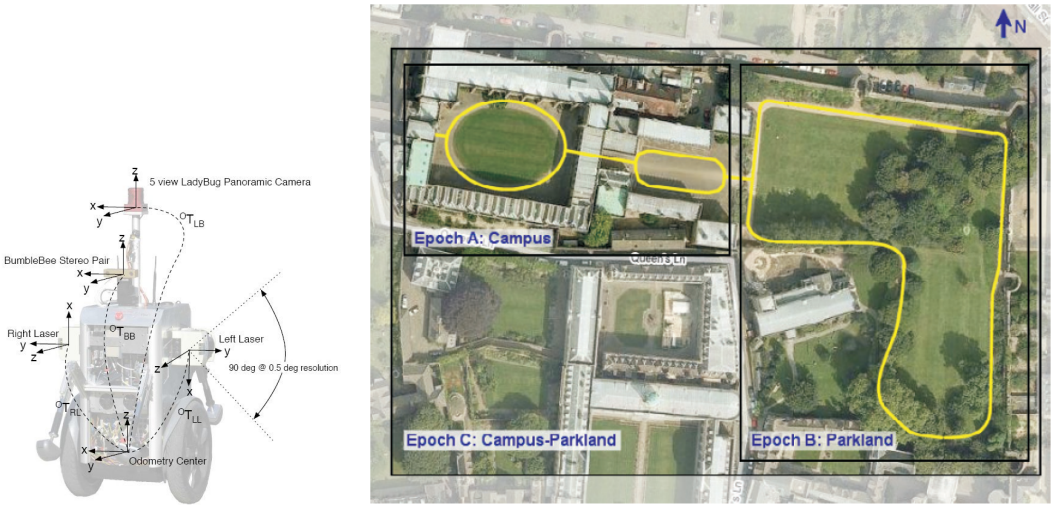
Columns 5 and 6 of Table 5.2 provide details of the timings for Experiment 3. Two important differences with the outdoor dataset were that the appearance of the scene was far more textured and as such the disparity estimation and 3D feature tracking was more reliable. The effect of this can be seen in the speed up of the stereo odometry computation which was principally due to significantly less iterations of the RANSAC procedure. Detailed plots of the timings for Experiment 3 are shown in Figs. 5.19– 5.21.

One issue encountered during the outdoor sequences was intra-frame aliasing of SURF features (e.g. due to the repeated red bricks). This resulted in a number of true positive loop closures from the place recognition system being rejected due to a failure of the geometric consistency test, which was in turn due to a failure of the SURF correspondence estimation. For example in experiment 3 the total number of loop closures was 13. This is the reason why, although the total number of poses is similar to Experiment 2, the iSAM processing time is lower.

5.3 New College Dataset Experiments

In this section we provide results for a similar set of experiments to the previous sections but using the Oxford New College Dataset [129]. This dataset is a public dataset widely used in benchmarking of SLAM systems. The dataset consists of data collected from multiple sensors mounted on a Segway based robot (see Figure 5.22(a)), including lidar, stereo and omnidirectional imagery. As described in Smith et al. [129], the dataset is divided into three epoch's, A (Campus), B (Parklands), and C (Campus & Parklands) as shown in Figure 5.22(b).

To evaluate the performance of the system over this dataset both single session and multi-session experiments were performed. For the single session experiment the system was run on the Epoch A component of the dataset. The results of this exper-



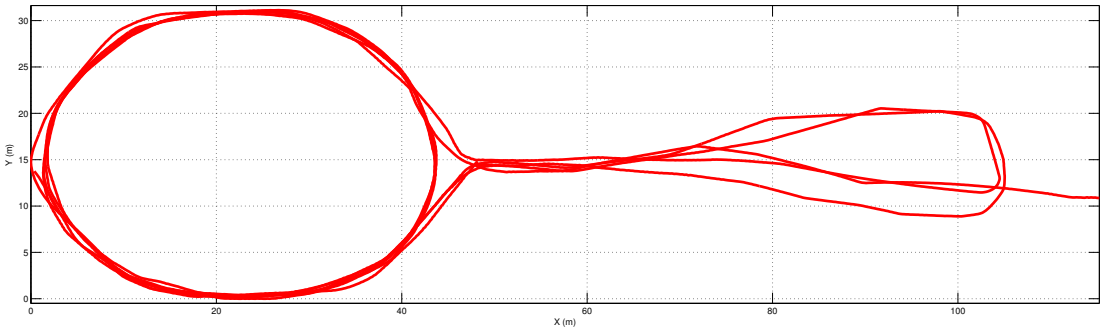
(a) (b)

Figure 5.22: Overview of the New College Dataset from Smith et al. [129]. (a) Segway robotic sensor platform. (b) Aerial view of New College showing the trajectory of the robot for epochs A-C.

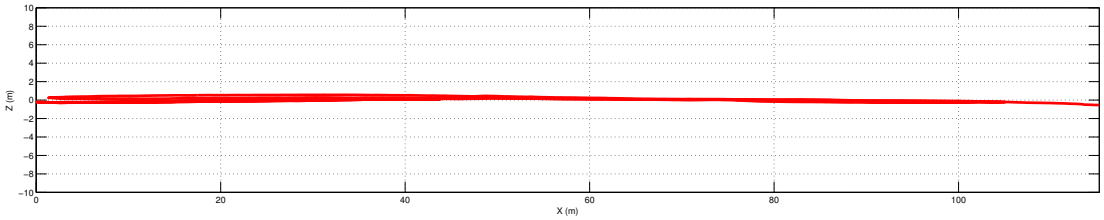
iment are shown in Figure 5.23, where a set of plots similar to those in Figure 5.10 are provided. Here, however the platform moves at a much higher velocity which results in a significant increase in the average number of keyframes per second. As a consequence over the ~1000s of log data the result pose graph contains 4334 poses.

Figure 5.23(a) shows an overhead view of the map where the same plane fitting process that was used in Figure 5.10 has been applied to the trajectory. Figures 5.23(b) and 5.23(c) provide two plots of the X and Z coordinates of the trajectory. Here both axes are at the same scale in Figure 5.23(b), showing that the variation in Z is low compared to the scale of the trajectory. In Figure 5.23(c) the scale on Z-axis is zoomed to highlight the structure of the variation in Z. Finally Figures 5.23(d) and 5.23(e) show the variation in Z as a function of distance travelled and a histogram of the Z values, respectively. Here the total distance travelled is ~967m with a median variation in Z of 0.0261m and a standard deviation of 0.2056m.

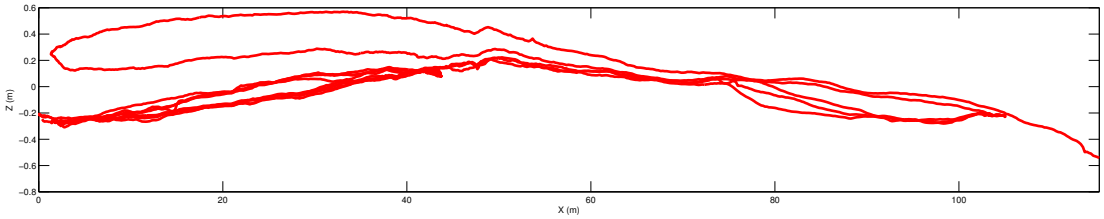
To evaluate the multi-session operation of the system using the New College dataset, the complete dataset, including all three epochs, was divided into 5 sep-



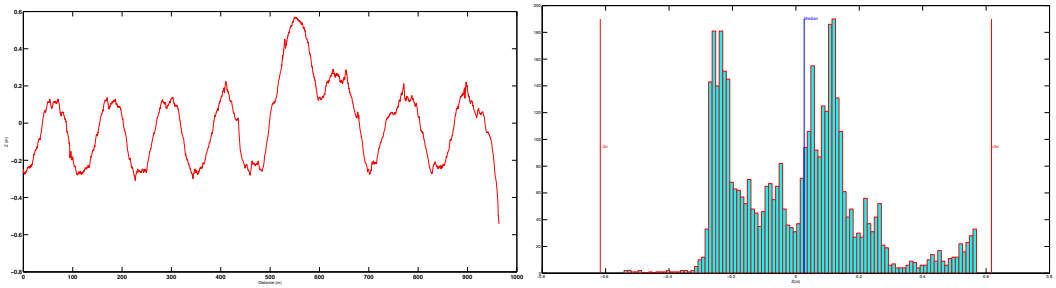
(a)



(b)



(c)



(d) (e)

Figure 5.23: Single session Oxford New College experimental results. Figure (a) shows a plan view of the resulting trajectory. Figure (b) shows the variation in Z as a function of X position. Figure (c) shows the same data as Figure (b) with the scale on the Z axis scaled to highlight the variation in Z. Finally, Figure (d) shows the variation in Z as a function of distance travelled, whilst Figure (e) shows a histogram of the Z values.

arate sessions in a similar manner to the experiments in Section 5.2. The outputted maps from each of the sessions are shown in Figure 5.24, with the full multi-session map shown in Figure 5.25. This map consists of 11032 poses, 177 intra-session loop closures, and 15 inter-session loop closures, and hence contains an order of magnitude more poses than in the experiments from the previous section.

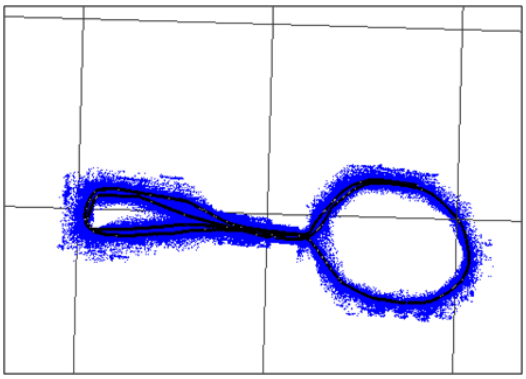
One area of obvious drift in the map is towards the end of the Session 4 trajectory. This was caused by an odometry failure at the top of the map (relative to its orientation in Figure 5.24(d)) resulting in a local angular error which in turn resulted in an accumulating translational error. Failure of the place recognition system to identify a loop closure in this region of the pose graph meant that no constraints were available to correct for this error.

Figures 5.26 - 5.28 provide plots for the same set of processing times as the multi-session experiments from the previous section. Note that given the larger number of keyframes, in this case Figures 5.26(a) and 5.26(b) show histograms of the running times for the windowed bundle adjustment and place recognition modules, respectively.

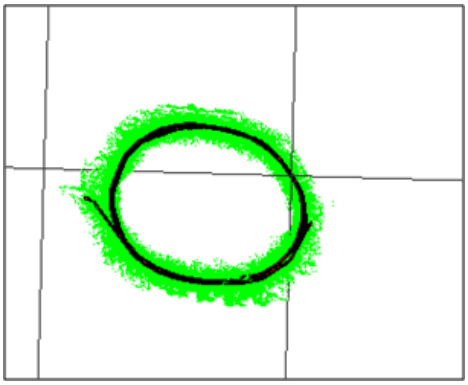
Figure 5.28 shows the cumulative processing times for the front-end and back-end processing. Given the speed of motion and the fact that the system is operating in an outdoor setting (i.e. with highly textured imagery), as would be expected, the plots show that the system exhibits a performance profile similar to the handheld 3-session experimental results from Section 5.2.

5.4 Stata Ground Truth Dataset Experiments

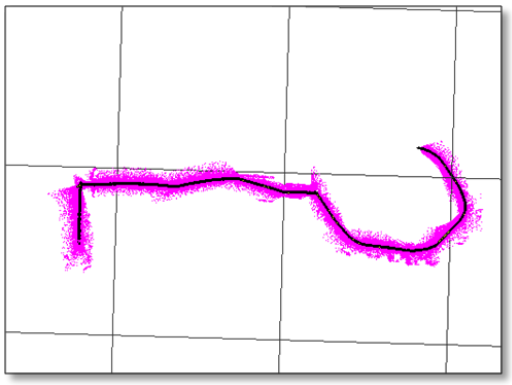
A number of the experiments reported throughout this chapter have addressed the issue of accuracy by quantifying the variation in the height of the sensor in the wheeled datasets, of by visually comparing the resulting maps to building floor plans. In fact



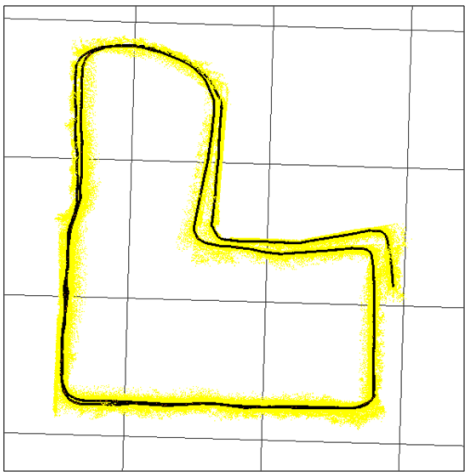
(a) Session 1



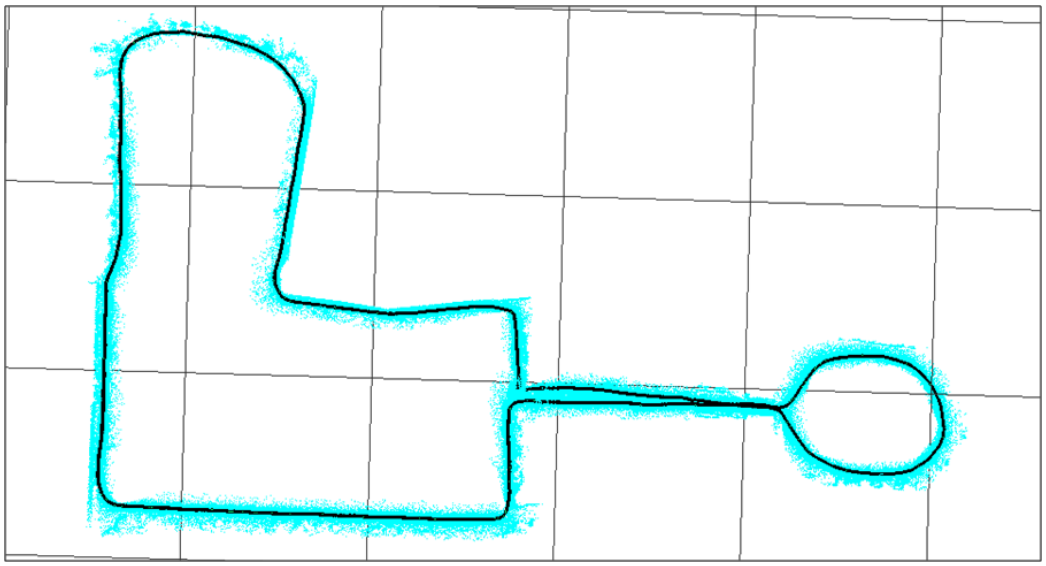
(b) Session 2



(c) Session 3



(d) Session 4



(e) Session 5

Figure 5.24: Individual sessions of New College multi-session experiment.

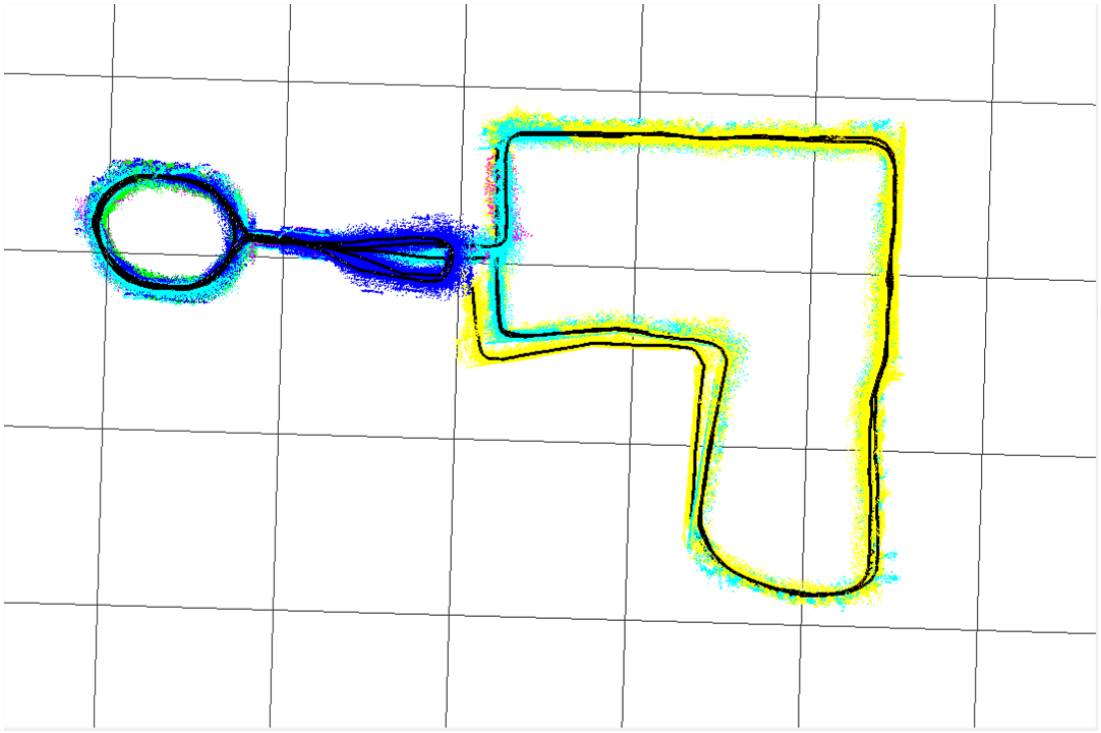
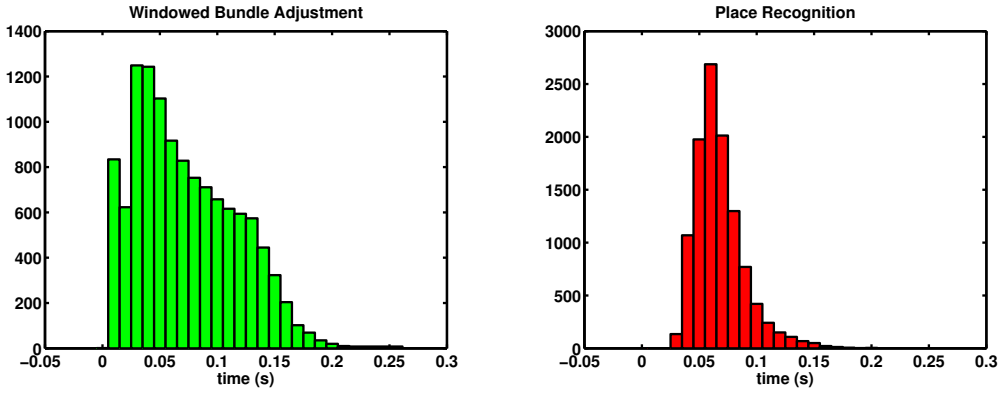


Figure 5.25: New College multi-session map including five separate sessions



(a) (b)

Figure 5.26: Distribution of processing times per keyframe for New College multi-session indoor dataset. Per keyframe processing times for (a) iSAM based windowed bundle adjustment and (b) place recognition.

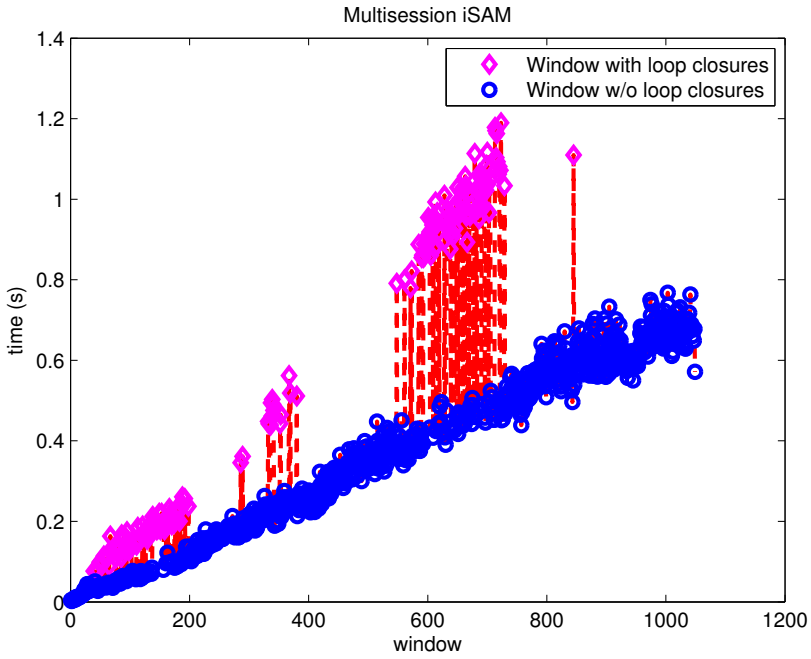
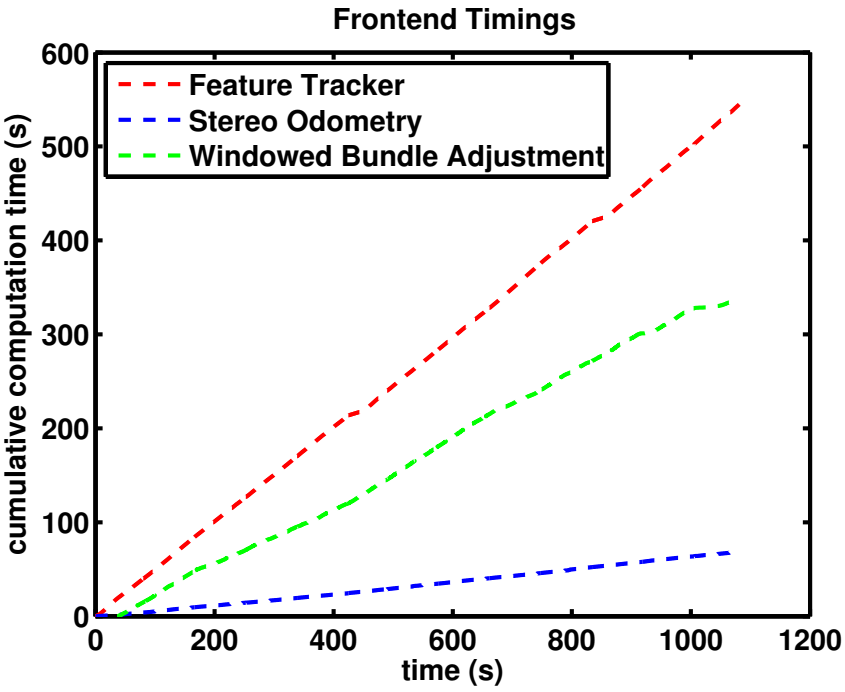
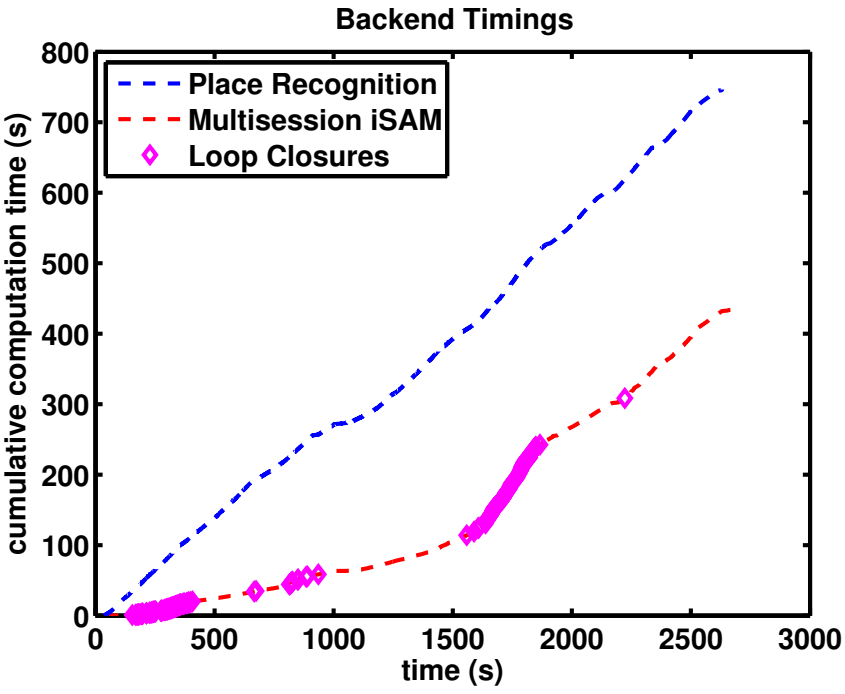


Figure 5.27: Multi-session New College processing times for the iSAM based pose graph optimisation. Note that the timings are per window where the purple diamond data points and blue circle data points distinguish between windows with and without loop closures, respectively.



(a)



(b)

Figure 5.28: Cumulative processing times for the multi-session New College experiment. These two plots show the total processing time for (a) the front-end, and, (b) the back-end, both as a function of the total input sequence length in seconds.

this has been the approach used throughout the research community in the assessment of the majority of large scale visual SLAM systems. Although such an approach is useful in terms of providing a qualitative (or limited quantitative) assessment of the accuracy of SLAM systems, direct comparison of the performance of different approaches requires the availability of datasets that provide ground truth pose information for the platform. A number of groups have published such datasets, however they typically employ expensive motion tracking systems that only operate over a limited space. As a consequence capturing benchmark datasets for large scale SLAM systems cannot exploit such systems.

Recent work by Fallon et al. [37] takes an alternative approach whereby they captured a multi-sensor dataset using a PR2 robot over the 10 floors of the Stata Center at MIT. The dataset was captured over a number of years and currently consists of 38 hours of data with a total trajectory length of 42 kilometres. An important aspect of this dataset is that a large component of it contains ground truth pose information for each frame of the data. This information was computed by manually aligning batches of consecutive lidar scans from the dataset with as-built floor plans of the building. The authors report that the ground truth has a typical accuracy of 2 cm. The entire dataset is available at `http://projects.csail.mit.edu/stata/`.

In this section we present a comparison of the output of the visual SLAM system developed in this thesis with the ground truth for log 2012-01-25-12-14-25. This log consists of 18.6 mins of data covering a trajectory of ~$964m$ from the second floor of the Stata Center. The results use only the visual data from the dataset and make no use of the wheel odometry, etc.

Figure 5.29 shows the ground truth trajectory and the trajectory estimated using the visual SLAM system, which are overlaid on the building floor plan. As can be seen from the figure, both trajectories are in close agreement, except for a segment at

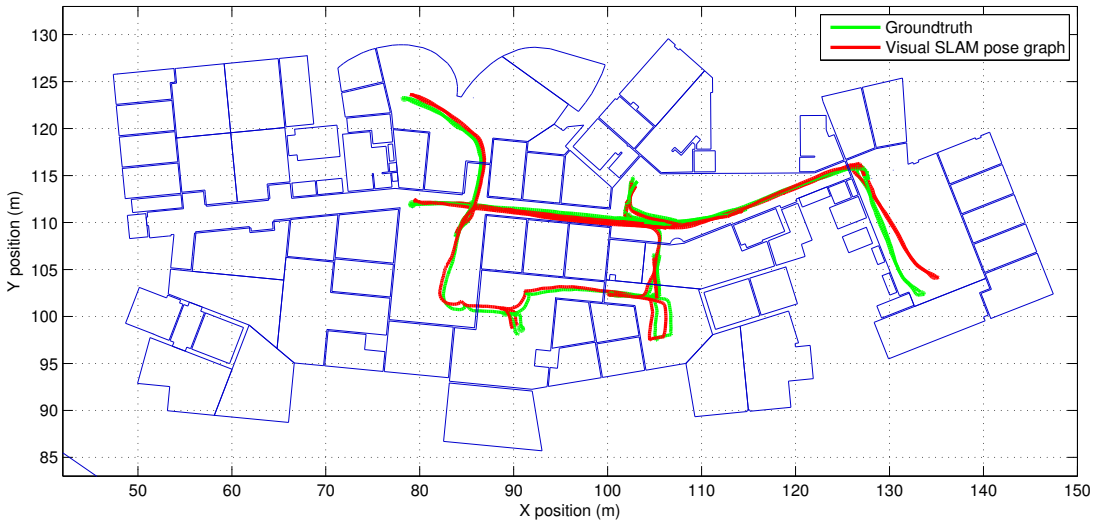
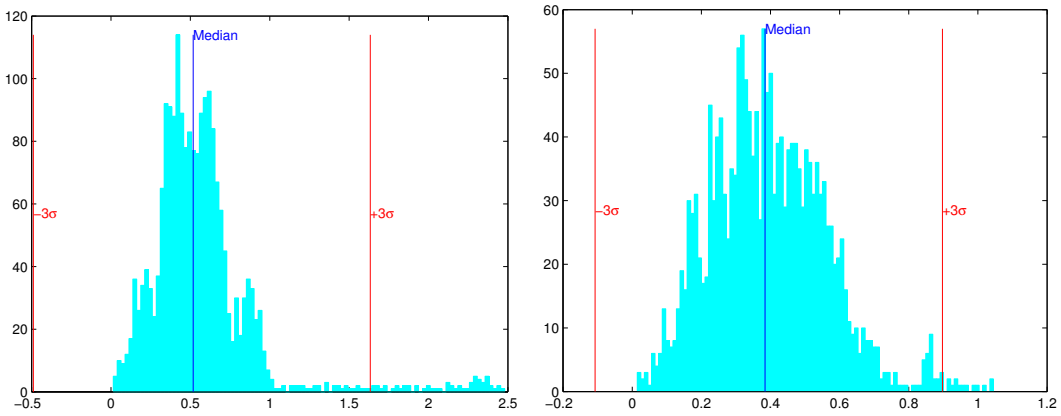


Figure 5.29: Comparison of system to ground truth for log 2012-01-25-12-14-25 of the Stata Ground Truth Dataset.

the right hand end of the map. This drift in the trajectory was due to an odometry failure at the right angled turn at the beginning of the segment resulting in a local angular error, which in turn results in an accumulating translational error.

The distribution of absolute position errors between each pose of the estimated trajectory and the corresponding ground truth pose is shown in Figure 5.30. Figure 5.30(a) shows the distribution for the full trajectory. In order to assess the impact of the of the above drift, Figure 5.30(b) shows the absolute error distribution between the ground truth and a cropped version of the estimated trajectory where the poses due to the spur at the right hand end of the map have been removed.

Finally Table 5.3 provides summary statistics for both the full and cropped trajectories. As can be seen from the table the full trajectory has a median error of ~52cm, whereas the mean is ~57cm due to the outliers from the drift. The statistics for the cropped trajectory, however, show a much lower error and fewer outliers with the result that there is less disparity between the median (~38cm) and mean (~39cm).



(a) (b)

Figure 5.30: Error distributions for Stata Ground Truth experiment shown in Figure 5.29. Here (a) shows the distribution for the full trajectory where the errors due to the drift caused by an odometry failure at the right hand end of the map are apparent. To show the effect of this error on the distribution, (b) shows the distribution for a cropped version of the red trajectory, where this component of the trajectory has been removed.

Trajectory	Median	Mean	Std. Dev.	Num. Poses
Full	0.5180m	0.5721m	0.3536m	2024
Cropped	0.3841m	0.3949m	0.1675m	1793

Table 5.3: Stata Ground Truth Dataset error statistics.

CHAPTER 6

Conclusions

6.1 Thesis Contributions

This thesis presented a 6-DOF multi-session visual SLAM system. The principal contribution of the thesis is the integration of all the components required for a multi-session visual SLAM system using iSAM with the anchor node formulation [66]. In particular this is the first example of an anchor node based SLAM system that (i) uses vision as the primary sensor, (ii) operates in general 6-DOF motion, (iii) includes a visual place recognition module for identifying encounters in general environments, and (iv) derives 6-DOF pose constraints from those loop closures within these general environments (i.e. removing the need for fiducial targets, as were used in [66]).

We have demonstrated this system in indoor and outdoor environments using both wheeled and handheld sensors as input. We have presented examples of single and multi-session pose graph optimisation and map construction, and provided a comprehensive quantitative assessment of the system's performance in a number of

different scenarios.

At the outset of this thesis the goal was to extend the previous work of Kim et al. [66] to a full multi-session visual SLAM framework. At that time, although the field of visual SLAM was being pursued by many researchers [103], no visual SLAM system designed for operation in large scale environments was publicly available; systems such as PTAM [67], which had been released as open source were targeted at small scale environments. Hence, for this thesis investigation, it was first necessary to develop a complete single-session visual SLAM system capable of operating over extended spatial scales. This aspect of the thesis proved to be one of the most challenging requiring the development of a complete modular visual SLAM front-end and back-end. For the front-end these developments included (i) the integration of many components from a diverse set of libraries, (ii) the extension of iSAM for windowed bundle adjustment, (iii) the implementation of a complete stereo visual odometry module, and (iv) the development of a number of utilities for simplifying the stereo processing, visualisation, message passing, etc. The back-end required (i) integration of a visual place recognition, (ii) computing pose constraints for the loop closure candidates, and (iii) management and optimisation of the resultant multi-session pose graph using iSAM.

An early design decision was to utilise a modular framework in order to minimise the dependencies between the components of the system. Central to achieving this was the use of the Lightweight Communications and Marshalling (LCM) library [57] which provided an efficient messaging framework where modules operate using a *publish-and-subscribe* paradigm. Using LCM means that the only explicit dependencies between the modules are the message formats for encoding their inputs and outputs with each module implemented as a stand-alone process. Such a decoupled architecture is very suitable for modern multi-core CPU's where each module can execute quasi-independently on a separate core. This was a key element in achieving real-time

Package Name	Link
camunits	`https://code.google.com/p/camunits/`
cholmod	`http://www.cise.ufl.edu/research/sparse/cholmod/`
DBOW	`http://webdiis.unizar.es/~dorian/index.php?p=31`
eigen	`http://eigen.tuxfamily.org/index.php?title=Main_Page`
iSAM	`http://people.csail.mit.edu/kaess/isam/`
GPU-KLT	`http://www.inf.ethz.ch/personal/chzach/opensource.html`
lcm	`https://code.google.com/p/lcm/`
levmar	`http://users.ics.forth.gr/~lourakis/levmar/`
libbot2	`https://code.google.com/p/libbot/`
OpenCV	`http://opencv.org/`
pods	`http://sourceforge.net/p/pods/home/Home/`
viewer	`http://grandchallenge.mit.edu`

Table 6.1: Open source software packages used in the multi-session visual SLAM system.

operation for the system.

In total the system involved approximately 20K lines of code (excluding external libraries). A list of the external libraries used in the system is provided in Table 6.1. One of the goals for the future work after this thesis is to provide an open source release of the multi-session visual SLAM system to permit use by other researchers. An ultimate goal is to enable future roboticists to simply issue the command `apt-get install vslam` and to be able to download and operate a robust and computationally efficient visual SLAM system that provides persistent multi-session operation for a wide range of application domains.

6.2 Future Directions

Although multi-session visual mapping can provide part of the solution to long-term persistent localisation and mapping other issues must also be addressed. The most pertinent of these issues were introduced in Section 1.2 including a description of work published by a number of researchers for addressing them. In this section we revisit these issues and highlight opportunities that exist for combining the work of

this thesis with these advances.

An assumption of the current system is that only true positive loop closures are introduced into the pose graph. Although this assumption holds true for the experiments carried out in Chapter 5, in general, and especially in situations involving long-term autonomy, false positive loop closures will occur. Avoiding this situation can only be achieved by introducing more and more conservative thresholds on the loop closure detection process resulting in increasing rejections rates for valid loop closures, discarding valuable information in the estimation process.

An alternative approach is to start from the assumption that false positive loop closures do occur and to develop methods that are robust to such outliers. Recently significant progress has been made in this area by Sunderhauf and Protzel [138], Olson and Agrawal [112] and Latif et al. [78, 77]. Of particular relevance here is the work of Latif et al. [78, 77] who directly address the multi-session mapping problem in the context of outliers pose constraints.

A second issue with long term autonomy is the scalability problem. Using the anchor node formulation as the number and length of the sessions increases so too will the size of the pose graph in an unbounded fashion. As mentioned in the introduction the anchor node formulation is directly related to the base node formulation used in T-SAM due to Ni et al. [108]. The principal difference here is that T-SAM is a batch approach focussed on enabling out-of-core estimation where only a subset of the factor graph needs to be in memory at any one time. We consider the anchor node formulation to be complementary to this approach where a T-SAM type sub-division could be applied for scalability whilst the anchor node formulation could be employed for combining multiple sessions within each of the "plates".

An alternative approach to achieving scalability is presented by Johannsson's reduced pose graph representation to achieve temporal scalability [58]. Here although new measurements are continually used to improve the pose graph, a new pose node

is only introduced if it is located outside of what is referred to as the "active region" of existing pose nodes. In this way the number of nodes in the pose graph is a function of the scale of the space mapped.

Independently of this thesis the author has been involved in the development of the Kintinuous mapping algorithm for dense mapping methods over extended scale environments [149, 150, 151]. Initially this focussed on development of a shifting scheme based on a cyclic buffer framework. This allowed continuous KinectFusion type mapping beyond the original confines of the technique imposed by the limits of the GPU memory [149]. Further research improved the robustness of the odometry system by incorporating dense visual methods [150]. An example map of our earlier work is shown in Fig. 6.1(a).

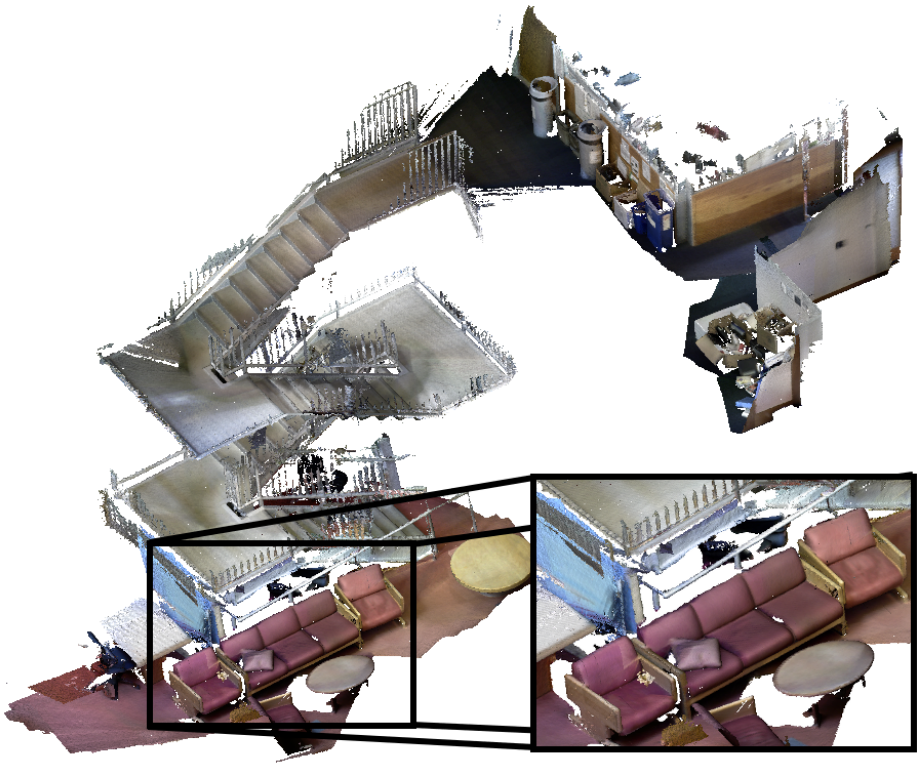
More recently we have also developed a solution to the global consistency problem for dense mapping systems [151]. Here the underlying camera trajectory is modelled using a pose-graph and is optimised using iSAM. The solution provides a method for smoothly mapping corrections from the pose graph to the dense map. An example of a globally aligned map is shown in Fig. 6.1(b). Here the insets show regions of the map where loop closures have occurred. These areas consist of regions computed during separate passes of the environment, showing the effectiveness of the alignment process.

As can be seen from the figure, the density and quality (both geometric and photometric) of the resultant maps are far richer than those produced by the previous generation of feature based visual SLAM systems. Such systems are opening up new possibilities in SLAM where the maps are providing the foundations for reasoning about the environment at a semantic level [41, 122].

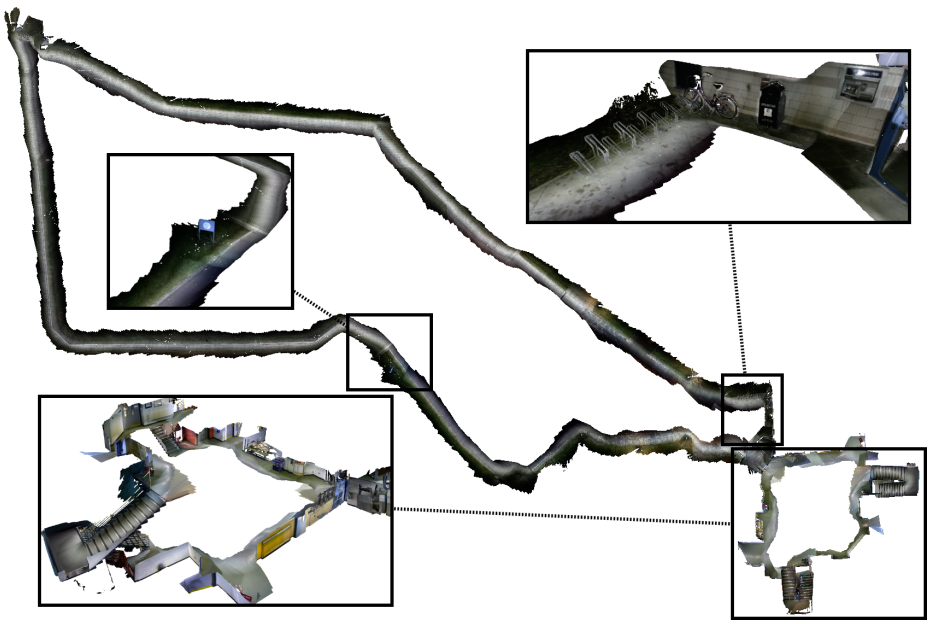
The capability of combining multiple dense mapping sessions is equally relevant in the context of long term autonomous operation. Furthermore given their reliance on RGBD sensors, which have limited applicability in outdoor settings, multi-session

mapping provides a suitable paradigm for multi-sensor approaches. For example creating metrically aligned dense maps of multiple indoor spaces could be achieved by first utilising the stereo mapping approach presented in this thesis to map the adjoining outdoor spaces. Fusing the resulting dense and sparse sessions could then be achieved using the presented multi-session back-end.

In summary, returning to the key themes identified in Chapter 1: (1) Scalable Representations, (2) Robustness, and (3) Dense Mapping & Semantics – the hope is that the anchor node multi-session framework described in the thesis can be extended in all three directions. The anchor node formulation has the potential to dovetail nicely with Johannsson's reduced pose graph representation to achieve temporal scalability [58]. Multi-session operations can also operate in concert with new robust pose graph optimisation approaches, such as Sunderhauf and Protzel [138], Olson and Agrawal [112] and Latif et al. [78, 77]. Finally, the multi-session approach should apply equally well to dense mapping methods such as Kintinuous [150].



(a) Textured dense map computed using the Kintinuous mapping algorithm incorporating both geometric and dense visual odometry. For further details see Whelan et al. [150]



(b) Globally consistent extended scale dense map computed using Kintinuous. For further details see Whelan et al. [151]

Figure 6.1: Examples of dense mapping with Kintinuous. Whelan et al. [149, 150, 151].

References

[1] M. Agrawal, K. Konolige, and M. Blas. Censure: Center surround extremas for realtime feature detection and matching. In *Computer Vision - ECCV 2008*, volume 5305 of *Lecture Notes in Computer Science*, pages 102–115. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-88692-1. URL `http://dx.doi.org/10.1007/978-3-540-88693-8_8`.

[2] M. Angermann and P. Robertson. Footslam: Pedestrian simultaneous localization and mapping without exteroceptive sensors - hitchhiking on human perception and cognition. *Proceedings of the IEEE*, 100(Special Centennial Issue): 1840–1848, 2012.

[3] N. Ayache and O. Faugeras. Building, registrating and fusing noisy visual maps. *Intl. J. of Robotics Research*, 7(6):45–65, 1988.

[4] A. Bachrach, S. Prentice, R. He, and N. Roy. RANGE - robust autonomous navigation in GPS-denied environments. *J. of Field Robotics*, 28(5):644–666, September 2011.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features

(SURF). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. ISSN 1077-3142. doi: 10.1016/j.cviu.2007.09.014.

[6] Charles Bibby and Ian Reid. Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association. In *Robotics: Science and Systems (RSS)*, June 2007.

[7] S. Birchfield. Derivation of the Kanade-Lucas-Tomasi tracking equation, January 1997. URL `http://www.ces.clemson.edu/~stb/klt/birchfield-klt-derivation.pdf`.

[8] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[9] M. Bosse, P. Newman, J. Leonard, and S. Teller. Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework. *Intl. J. of Robotics Research*, 23(12):1113–1139, December 2004.

[10] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA, 2008.

[11] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 510–517, June 2005. doi: 10.1109/CVPR.2005.235.

[12] P. Burt and E. Adelson. The Laplacian pyramid as a compact image code. *Communications, IEEE Transactions on*, 31(4):532–540, April 1983. ISSN 0090-6778. doi: 10.1109/TCOM.1983.1095851.

[13] C. Cadena, D. Gálvez, F. Ramos, J.D. Tardós, and J. Neira. Robust place recognition with stereo cameras. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010.

[14] C. Cadena, J. McDonald, J. Leonard, and J. Neira. Place recognition using near and far visual information. In *Proceedings of the 18th IFAC World Congress*, August 2011.

[15] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *Proceedings of the 11th European conference on Computer vision: Part IV*, Eur. Conf. on Computer Vision (ECCV), pages 778–792, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15560-X, 978-3-642-15560-4. URL `http://dl.acm.org/citation.cfm?id=1888089.1888148`.

[16] R.O. Castle, G. Klein, and D.W. Murray. Wide-area augmented reality using camera tracking and mapping in multiple regions. *Computer Vision and Image Understanding*, 115(6):854 – 867, 2011. ISSN 1077-3142. doi: 10.1016/j.cviu.2011.02.007. URL `http://www.sciencedirect.com/science/article/pii/S1077314211000701`.

[17] S. Choi, T. Kim, and W. Yu. Performance evaluation of RANSAC family. In *Proc. BMVC*, pages 81.1–81.12, 2009. ISBN 1-901725-39-1. doi:10.5244/C.23.81.

[18] M. Csorba. *Simultaneous Localisation and Map Building*. PhD thesis, University of Oxford, 1997.

[19] M. Cummins. *Probabilistic Localization and Mapping in Appearance Space*. PhD thesis, University of Oxford, 2009.

[20] M. Cummins and P. Newman. Probabilistic appearance based navigation and loop closing. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2042–2048, April 2007.

[21] M. Cummins and R. Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *Intl. J. of Robotics Research*, 2010.

[22] A. Cunningham, M. Paluri, and F. Dellaert. DDF-SAM: Fully distributed SLAM using constrained factor graphs. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3025 –3030, October 2010. doi: 10.1109/IROS.2010.5652875.

[23] A. Cunningham, K. Wurm, W. Burgard, and F. Dellaert. Fully distributed scalable smoothing and mapping with robust multi-robot data association. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2012.

[24] A. Cunningham, V. Indelman, and F. Dellaert. Ddf-sam 2.0: Consistent distributed smoothing and mapping. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 05/2013 2013.

[25] T. A. Davis. *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2006. doi: 10.1137/1.9780898718881.

[26] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410, 2003.

[27] Andrew Davison. *Mobile Robot Navigation Using Active Vision*. PhD thesis, University of Oxford, 1998.

[28] F. Dellaert. Square Root SAM: Simultaneous location and mapping via square root information smoothing. In *Robotics: Science and Systems (RSS)*, Cambridge, MA, 2005.

[29] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research*, 25(12):1181–1203, December 2006.

[30] J. Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. Technical report, Stanford University, 2006.

[31] G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localisation and map building (SLAM) problem. *IEEE Transactions of Robotics and Automation*, 17(3):229–241, 2001.

[32] H.F. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (SLAM): Part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110, June 2006. ISSN 1070-9932. doi: 10.1109/MRA.2006.1638022.

[33] E. Eade and T. Drummond. Unified loop closing and recovery for real time monocular SLAM. In *British Machine Vision Conference*, 2008.

[34] Chris Engels, Henrik StewÃ©nius, and David NistÃ©r. Bundle adjustment rules. In *In Photogrammetric Computer Vision*, 2006.

[35] C. Estrada, J. Neira, and J.D. Tardós. Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Trans. Robotics*, 21(4):588–596, 2005.

[36] R.M. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters for view-based SLAM. *IEEE Trans. Robotics*, 22(6):1100–1114, December 2006. ISSN 1552-3098. doi: 10.1109/TRO.2006.886264.

[37] M. Fallon, H. Johannsson, M. Kaess, and J.J. Leonard. The MIT Stata Center dataset. *Intl. J. of Robotics Research*, 32(14):1695–1699, Dec 2013.

[38] M. F. Fallon, H. Johannsson, J. Brookshire, S. Teller, and J. J. Leonard. Sensor fusion for flexible human-portable building-scale mapping. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Algarve, Portugal, 2012.

[39] M.F. Fallon, M. Kaess, H. Johannsson, and J.J. Leonard. Efficient AUV navigation fusing acoustic ranging and side-scan sonar. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[40] M.F. Fallon, H. Johannsson, and J.J. Leonard. Efficient scene simulation for robust Monte Carlo localization using an RGB-D camera. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1663–1670, St. Paul, MN, May 2012.

[41] R. Finman, T. Whelan, M. Kaess, and J. Leonard. Toward lifelong object segmentation from change detection in dense RGB-D maps. In *European Conference on Mobile Robotics*, Barcelona, Spain, September 2013. Accepted. To appear.

[42] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/358669.358692.

[43] C. Forster, L. Kneip, S. Lynen, and R. Siegwart. Centralized multi-robot monocular SLAM. In *Workshop on Integration of perception with control and navigation for resource-limited, highly dynamic, autonomous systems, In Robotics: Science and Systems (RSS)*, 2012.

[44] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct point, corners and centres of circular features. In *Proceedings of the ISPRS Conference on Fast Processing of Photogrammetric Data*, pages 281–305, 1987.

[45] D. Fox, J. Ko, B. Stewart, K. Konolige, and B. Limetkai. Distributed multi-

robot mapping. *Multi-Robot Systems: From Swamrs to Intelligent Automata*, 2, 2003.

[46] U. Frese. Interview: Is SLAM solved? *KI-Künstliche Intelligenz*, 24(3):255–257, 2010.

[47] F.S. Grassia. Practical parameterization of rotations using the exponential map. *J. Graph. Tools*, 3:29–48, March 1998. ISSN 1086-7651.

[48] J. A. Grunert. Das Pothenotische Problem in erweiterter Gestalt nebst Über seine Anwendungen in der G eodäsie. In *Grunerts Archiv für Mathematik und Physik, Band 1*, pages 238–248. 1841.

[49] J. Guivant and E. Nebot. Compressed filter for real time implementation of simultaneous localization and map building. In *Field and Service Robot 2001*, pages 309–314, Finland, June 2001. ISBN 952-5183-13-0.

[50] J. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, volume 1, pages 318–325, 1999.

[51] B.C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. Springer, 2000.

[52] B. M. Haralick, C. N. Lee, K. Ottenberg, and M. Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13:331–356, 1994. URL `http://dx.doi.org/10.1007/BF02028352`.

[53] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of The Fourth Alvey Vision Conference*, volume 15, pages 147–151. Manch-

ester, UK, 1988. URL http://www.cis.rit.edu/~cnspci/references/dip/harris1988.pdf.

[54] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. Second Edition.

[55] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, April 1987. doi: 10.1364/JOSAA.4.000629. URL http://josaa.osa.org/abstract.cfm?URI=josaa-4-4-629.

[56] A. Howard. Multi-robot simultaneous localization and mapping using particle filters. In *Robotics: Science and Systems (RSS)*, June 2005.

[57] A. Huang, E. Olson, and D. Moore. LCM: Lightweight communications and marshalling. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010.

[58] H. Johannsson, M. Kaess, M.F. Fallon, and J.J. Leonard. Temporally scalable visual SLAM using a reduced pose graph. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[59] S.J. Julier and J.K. Uhlmann. A counter example to the theory of simultaneous localization and map building. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 4, pages 4238–4243, 2001. doi: 10.1109/ROBOT.2001.933280.

[60] M. Kaess. *Incremental Smoothing and Mapping*. PhD thesis, Georgia Institute of Technology, December 2008.

[61] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. Robotics*, 24(6):1365–1378, December 2008.

[62] M. Kaess, H. Johannsson, and J.J. Leonard. Incremental smoothing and mapping (iSAM) library. http://people.csail.mit.edu/kaess/isam, 2010–2012.

[63] 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*, 31:217–236, February 2012.

[64] A. Kim and R.M. Eustice. Real-time visual SLAM for autonomous underwater hull inspection using visual saliency. *IEEE Trans. Robotics*, 2013. To appear.

[65] Ayoung Kim and Ryan M. Eustice. Combined visually and geometrically informative link hypothesis for pose-graph visual SLAM using bag-of-words. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1647–1654, San Francisco, CA, September 2011.

[66] B. Kim, M. Kaess, L. Fletcher, J.J. Leonard, A. Bachrach, N. Roy, and S. Teller. Multiple relative pose graphs for robust cooperative mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 3185–3192, Anchorage, Alaska, May 2010.

[67] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, pages 225–234, Nara, Japan, November 2007.

[68] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3232–3238, 2003.

[69] K. Konolige and M. Agrawal. FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Trans. Robotics*, 24(5):1066–1077, October 2008.

[70] K. Konolige and J. Bowman. Towards lifelong visual maps. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1156–1163, October 2009.

[71] K. Konolige, D. Fox, C. Ortiz, A. Agno, M. Eriksen, B. Limketkai, J. Ko, B. Morisset, D. Schulz, B. Stewart, and R. Vincent. Centibots: Very large scale distributed robotic teams. *Experimental Robotics IX*, 21:131–140, 2006.

[72] K. Konolige, J. Bowman, J.D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-based maps. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[73] K. Konolige, J. Bowman, J.D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-based maps. *Intl. J. of Robotics Research*, 29(8):941–957, July 2010.

[74] J. B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality.* Princeton University Press, Princeton, New Jersey, 1999. ISBN 0-691-05872-5.

[75] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[76] C.G. Kunz. *Autonomous underwater vehicle navigation and mapping in dynamic, unstructured environments.* PhD thesis, Massachusetts Institute of Technology, 2012.

[77] Yasir Latif, César Cadena, and José Neira. Realizing, reversing, recovering: Incremental robust loop closing over time using the iRRR algorithm. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4211–4217. IEEE, 2012.

[78] Yasir Latif, Cesar Cadena Lerma, and Jose Neira. Robust loop closing over time. In *Robotics: Science and Systems (RSS)*, Sydney, Australia, July 2012.

[79] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate $O(n)$ solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.

[80] S. Leutenegger, M. Chli, and R.Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555, November 2011. doi: 10.1109/ICCV.2011.6126542.

[81] T. Lindeberg. Scale-space theory: a basic tool for analyzing structures at different scales. *Journal of Applied Statistics*, 21(1-2):225–270, 1994. doi: 10.1080/757582976.

[82] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL `http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94`.

[83] C.-P. Lu, G.D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(6):610 –622, June 2000. ISSN 0162-8828. doi: 10.1109/34.862199.

[84] F. Lu and E. Milios. Globally consistent range scan alignment for environmental mapping. *Autonomous Robots*, 4:333–349, April 1997.

[85] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2D range scans. *Journal of Intelligent and Robotic Systems*, pages 249–275, April 1997.

[86] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international*

joint conference on Artificial intelligence - Volume 2, IJCAI'81, pages 674–679, 1981. URL http://dl.acm.org/citation.cfm?id=1623264.1623280.

[87] Y. Ma, S. Soatto, J. Košecká, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer, 2006.

[88] K. Madsen, H.B. Nielsen, and O. Tingleff. *Methods for Non-Linear Least Squares Problems*. Informatics and Mathematical Modeling, Technical University of Denmark, 2nd edition, 2004. Accessed through http://www2.imm.dtu.dk/~km/.

[89] K. Madsen, H.B. Nielsen, and O. Tingleff. Methods for non-linear least squares problems. Lecture notes, 2004. URL http://www.imm.dtu.dk/courses/02611/nllsq.pdf.

[90] L. Matthies. *Dynamic stereo vision*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1989.

[91] J. McDonald, M. Kaess, C. Cadena, J. Neira, and J.J. Leonard. 6-DOF multi-session visual SLAM using anchor nodes. In *European Conference on Mobile Robotics (ECMR)*, Orbero, Sweden, September 2011.

[92] J. McDonald, M. Kaess, C. Cadena, J. Neira, and J.J. Leonard. Real-time 6-DOF multi-session visual SLAM over large scale environments. *J. of Robotics and Autonomous Systems*, 61(10):1144–1158, October 2013.

[93] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. RSLAM: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision*, pages 1–17, 2010. ISSN 0920-5691. URL http://dx.doi.org/10.1007/s11263-010-0361-7. 10.1007/s11263-010-0361-7.

[94] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 525–531, 2001. doi: 10.1109/ICCV.2001.937561.

[95] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Machine Intell.*, 27(10):1615–1630, 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.188.

[96] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65:43–72, 2005. ISSN 0920-5691. doi: 10.1007/s11263-005-3848-x.

[97] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60:63–86, 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000027790.02288.f2.

[98] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI Conf. on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[99] J.M.M. Montiel, J. Civera, and A.J. Davison. Unified inverse depth parametrization for monocular SLAM. In *Robotics: Science and Systems (RSS)*, 2006.

[100] H. Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover*. PhD thesis, Stanford Univ., Stanford, CA, 1980.

[101] F. Moreno-Noguer, V. Lepetit, and P. Fua. Accurate non-iterative $o(n)$ solution to the PnP problem. In *Intl. Conf. on Computer Vision (ICCV)*, Rio de Janeiro, Brazil, October 2007.

[102] J. Neira and J. D. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. Robotics and Automation*, 17(6):890–897, December 2001.

[103] J. Neira, A.J. Davison, and J.J. Leonard. Guest editorial special issue on visual SLAM. *IEEE Trans. Robotics*, 24(5):929–931, October 2008. ISSN 1552-3098. doi: 10.1109/TRO.2008.2004620.

[104] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, pages 127–136, Basel, Switzerland, October 2011.

[105] R.A. Newcombe and A.J. Davison. Live dense reconstruction with a single moving camera. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 1498 –1505, June 2010. doi: 10.1109/CVPR.2010.5539794.

[106] R.A. Newcombe, S.J. Lovegrove, and A.J. Davison. DTAM: Dense tracking and mapping in real-time. In *Intl. Conf. on Computer Vision (ICCV)*, pages 2320–2327, Barcelona, Spain, November 2011.

[107] K. Ni and F. Dellaert. Multi-level submap based SLAM using nested dissection. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[108] K. Ni, D. Steedly, and F. Dellaert. Tectonic SAM: Exact, out-of-core, submap-based SLAM. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1678–1685, April 2007.

[109] D. Nistér. Preemptive RANSAC for live structure and motion estimation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 199–206 vol.1, 2003. doi: 10.1109/ICCV.2003.1238341.

[110] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168, 2006. doi: 10.1109/CVPR.2006.264.

[111] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *J. of Field Robotics*, 23(1):3–20, 2006. ISSN 1556-4967. doi: 10.1002/rob.20103. URL `http://dx.doi.org/10.1002/rob.20103`.

[112] E. Olson and P. Agrawal. Inference on networks of mixtures for robust robot mapping. In *Robotics: Science and Systems (RSS)*, July 2012.

[113] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2262–2269, May 2006.

[114] E. Olson, J. Strom, R. Goeddel, R. Morton, P. Ranganathan, and A. Richardson. Exploration and mapping with autonomous robot teams. *Communications of the ACM*, 56(3), March 2013. URL `http://doi.acm.org/10.1145/2428556.2428574`.

[115] Kari Pulli, Anatoly Baksheev, Kirill Kornyakov, and Victor Eruhimov. Real-time computer vision with opencv. *Commun. ACM*, 55(6):61–69, June 2012. ISSN 0001-0782. doi: 10.1145/2184319.2184337. URL `http://doi.acm.org/10.1145/2184319.2184337`.

[116] F. Ramos, D. Fox, and H. Durrant-Whyte. CRF-Matching: Conditional Random Fields for feature-based scan matching. In *Robotics: Science and Systems (RSS)*, 2007.

[117] F. Ramos, M.W. Kadous, and D. Fox. Learning to associate image features with CRF-Matching. In *Intl. Sym. on Experimental Robotics (ISER)*, pages 505–514, 2008.

[118] D.M. Rosen, M. Kaess, and J.J. Leonard. An incremental trust-region method for robust online sparse least-squares estimation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1262–1269, St. Paul, MN, May 2012.

[119] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Eur. Conf. on Computer Vision (ECCV)*, pages 430–443, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-33832-2, 978-3-540-33832-1. doi: 10.1007/11744023_34. URL `http://dx.doi.org/10.1007/11744023_34`.

[120] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Anal. Machine Intell.*, 32 (1):105 –119, January 2010. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.275.

[121] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Intl. Conf. on Computer Vision (ICCV)*, pages 2564–2571, Los Alamitos, CA, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-1101-5. doi: http://doi.ieeecomputersociety.org/10.1109/ICCV.2011.6126544.

[122] R.F. Salas-Moreno, R.A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. SLAM++: Simultaneous localisation and mapping at the level of objects. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, Portland, Oregon, June 2013.

[123] D. Scaramuzza and F. Fraundorfer. Visual odometry. Part I: The first 30 years and fundamentals. *IEEE Robotics & Automation Magazine*, 18(4):80 –92, December 2011. ISSN 1070-9932. doi: 10.1109/MRA.2011.943233.

[124] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593 –600, June 1994. doi: 10.1109/CVPR.1994.323794.

[125] G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Robotics: Science and Systems (RSS)*, Seattle, USA, June 2009.

[126] G. Sibley, C. Mei, I. Reid, and P. Newman. Planes, trains and automobiles – autonomy for the modern robot. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 285–292. IEEE, 2010.

[127] G. Sibley, C. Mei, I. Reid, and P. Newman. Vast-scale outdoor navigation using adaptive relative bundle adjustment. *Intl. J. of Robotics Research*, 2010. doi: 10.1177/0278364910369268. URL `http://ijr.sagepub.com/cgi/content/abstract/0278364910369268v1`.

[128] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Intl. Conf. on Computer Vision (ICCV)*, volume 2, page 1470, Los Alamitos, CA, USA, 2003. IEEE Computer Society. ISBN 0-7695-1950-4. doi: http://doi.ieeecomputersociety.org/10.1109/ICCV.2003.1238663.

[129] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman. The new college vision and laser data set. *The International Journal of Robotics Research*, 28 (5):595–599, 2009.

[130] R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *Intl. J. of Robotics Research*, 5:56–68, December 1986.

[131] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

[132] S. M. Smith and J. M. Brady. SUSAN-A new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997. ISSN 0920-5691. doi: 10.1023/A:1007963824710.

[133] J. Sola, A. Monin, and M. Devy. BiCamSLAM : Two times mono is more than stereo. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4795 –4800, April 2007. doi: 10.1109/ROBOT.2007.364218.

[134] C. V. Stewart. Robust parameter estimation in computer vision. *SIAM Review*, 41(3):513–537, 1999.

[135] H. Strasdat, J.M.M. Montiel, and A.J. Davison. Real-time monocular SLAM: Why filter? In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2010.

[136] H. Strasdat, A.J. Davison, J.M.M. Montiel, and K. Konolige. Double window optimisation for constant time visual SLAM. In *Intl. Conf. on Computer Vision (ICCV)*, 2011.

[137] N. Sünderhauf. *Robust Optimization for Simultaneous Localization and Mapping*. PhD thesis, Technischen Universitat Chemnitz, 2012.

[138] N. Sünderhauf and P. Protzel. Towards a robust back-end for pose graph SLAM. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1254 –1261, May 2012. doi: 10.1109/ICRA.2012.6224709.

[139] N. Sünderhauf and P. Protzel. Switchable constraints for robust pose graph SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, October 2012.

[140] J.D. Tardós, J. Neira, P.M. Newman, and J.J. Leonard. Robust mapping and localization in indoor environments using sonar data. *Intl. J. of Robotics Research*, 21(4):311–330, April 2002.

[141] C. J. Taylor and D. J. Kriegman. Minimization on the lie group $so(3)$ and related manifolds. Technical Report 9405, Yale University, April 1994.

[142] S. Thrun and Y. Liu. Multi-robot SLAM with sparse extended information filers. In *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, Sienna, Italy, 2003. Springer.

[143] S. Thrun, D. Koller, Z. Ghahramani, H.F. Durrant-Whyte, and A.Y. Ng. Simultaneous mapping and localization with sparse extended information filters. In J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, editors, *Proc. of the Fifth Intl. Workshop on Algorithmic Foundations of Robotics*, Nice, France, 2002.

[144] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, MA, 2005.

[145] P.H.S. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78 (1):138–156, 2000. ISSN 1077-3142. doi: http://dx.doi.org/10.1006/cviu.1999.0832.

[146] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *LNCS*, pages 298–372. Springer Verlag, 2000.

[147] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J.J. Leonard. Dynamic pose graph SLAM: Long-term mapping in low dynamic environments. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October 2012.

[148] M.R. Walter, R.M. Eustice, and J.J. Leonard. Exactly sparse extended information filters for feature-based SLAM. *Intl. J. of Robotics Research*, 26(4): 335–359, 2007.

[149] T. Whelan, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard, and J.B. McDonald. Kintinuous: Spatially extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, July 2012. Available as MIT CSAIL Technical Report MIT-CSAIL-TR-2012-020.

[150] T. Whelan, H. Johannsson, M. Kaess, J.J. Leonard, and J.B. McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[151] T. Whelan, M. Kaess, J.J. Leonard, and J.B. McDonald. Deformation-based loop closure for large scale dense RGB-D SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, November 2013.

[152] S. B. Williams. *Efficient Solutions to Autonomous Mapping and Navigation Problems*. PhD thesis, ACFR, Univ. of Sydney, Australia, September 2001.

[153] C. Zach, D. Gallup, and J.-M. Frahm. Fast gain-adaptive KLT tracking on the GPU. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–7, June 2008. doi: 10.1109/CVPRW.2008.4563089.

[154] D. Zou and P. Tan. CoSLAM: Collaborative visual SLAM in dynamic environments. *IEEE Trans. Pattern Anal. Machine Intell.*, 35(2):354–366, 2013. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2012.104.