

Experience Based Navigation: Theory, Practice and Implementation

Winston Churchill

St John's College



Supervisor:

Professor Paul Newman

Mobile Robotics Group

Department of Engineering Science

University of Oxford

October 2012

Experience Based Navigation: Theory, Practice and Implementation

Abstract

For robotic systems to realise lifelong autonomy they must be able to navigate accurately in changing environments. In this thesis we describe, implement and validate a new approach to the problem of long-term navigation.

To begin, we present our stereo visual odometry system which provides highly accurate pose estimation. Our approach combines several techniques found in existing implementations and a recently published image descriptor that simplifies the solution architecture. The performance and versatility of our system is demonstrated through testing on multiple datasets.

Equipped with our visual odometry system, we describe a new approach to the problem of lifelong navigation. We learn a model whose complexity varies naturally in accordance with the variation of scene appearance. As the robot repeatedly traverses its workspace, it accumulates distinct visual experiences that, in concert, implicitly represent the scene variation - each experience captures a visual mode. When operating in a previously visited area, we continually try to localise in these previous experiences while simultaneously running the visual odometry. Failure to localise in a sufficient number of prior experiences indicates an insufficient model of the workspace and instigates the laying down of the live image sequence as a new distinct experience. In this way, over time we can capture the typical temporally varying appearance of an environment and the number of experiences required tends to a constant. Although we focus on vision as a primary sensor, the ideas we present here are equally applicable to other sensor modalities. We demonstrate our approach working on a road vehicle operating over a three month period at different times of day, in different weather and lighting conditions.

Statement of Authorship

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Winston Churchill, St John's College

Funding

The work described in this thesis was funded by Oxford Technologies Ltd. and the Engineering and Physical Sciences Research Council (EPSRC).

Acknowledgements

First and foremost I would like to thank my supervisor, Professor Paul Newman. His continued support when repeatedly shifting focus and exploring new ideas was greatly appreciated. This, combined with his insight, guidance and humour has made the last four years hugely rewarding.

I also give thanks to the many people I have met along the way for sharing their knowledge, experiences and invaluable suggestions. The majority come from the outstanding Mobile Robotics Group Paul has built. Being a part of this team has been hugely enjoyable. I thank the many members, both past and present, for being great lab mates.

Finally, I would like to thank my family for their continued support throughout my time here.

Contents

1	Introduction	2
1.1	Motivation	2
1.2	Lifelong Navigation	3
1.3	Contributions	4
1.4	Thesis Structure	5
1.4.1	Structure	5
1.4.2	How to read this thesis	6
1.5	Publications and Exploitation	6
2	Stereo Visual Odometry	8
2.1	Introduction	8
2.2	Background	10
2.3	Overview	13
2.3.1	VO, SLAM and Frames of Reference	16
2.3.2	A Complete System	18
2.4	Stereo Cameras	20
2.4.1	Camera Models	20
2.4.2	Image Distortion Correction and Rectification	22
2.5	Front-End Pipeline	24
2.5.1	Feature Extraction and Description	24

2.5.2	Stereo Matching	26
2.5.3	Temporal Matching	27
2.6	Motion Estimation	28
2.6.1	Outlier Rejection	29
2.6.2	Numerical Optimisation	31
2.6.3	Robust Estimation	34
2.6.4	Choice of State	35
2.7	Conclusions	43
3	Systems and Datasets	44
3.1	Introduction	44
3.2	Datasets with Ground Truth	44
3.2.1	Synthetic Dataset	45
3.2.2	Real Datasets	47
3.3	Datasets with Multiple Loops	52
3.4	Published Datasets	54
3.5	Implementation Details	55
3.5.1	Acquiring, Managing and Consuming Data	56
3.6	Conclusions	56
4	Stereo Visual Odometry Analysis	58
4.1	Introduction	58
4.2	Error Metrics	58
4.3	Default Setup	60
4.4	Front-End Analysis	66
4.4.1	Number of Features	66
4.4.2	Feature Distribution	70
4.4.3	Rethreading	73

4.4.4	The BRIEF Feature Descriptor	76
4.4.5	Sub-Pixel Precision	80
4.5	Results on the Kitti Vision Benchmark Suite	82
4.6	Results in Other Places	84
4.7	VO Parameters	87
4.8	Practical Considerations	88
4.9	Conclusions	90
5	Lifelong Navigation By Experience	91
5.1	Introduction	91
5.2	Background	93
5.2.1	Long-Term Navigation	94
5.2.2	Teach and Repeat	97
5.3	Experience Based Navigation	98
5.3.1	Overview	98
5.3.2	Experiences	100
5.3.3	Localisers	101
5.3.4	Navigating by Experience	102
5.4	Connecting Experiences	103
5.4.1	The Experience Graph	105
5.4.2	External Loop Closing	106
5.5	Implementation	107
5.6	Results	108
5.6.1	Where were Experiences Saved?	108
5.6.2	When were Experiences Saved?	113
5.6.3	Choices of N	117
5.6.4	Are Multiple Experiences Needed?	120

5.6.5	Long-Term Trends	122
5.6.6	Descriptor Choice	124
5.6.7	Connectivity of the Experience Graph	127
5.6.8	Sensitivity to Lost Threshold	133
5.6.9	Weather Conditions	135
5.6.10	Timing Performance	136
5.7	Discussions	136
5.7.1	Sensor Choice	136
5.7.2	Forgetting Experiences	138
5.8	Conclusions	138
6	Conclusions	140
6.1	Summary of Contributions	140
6.2	Future Work	142
6.3	Concluding Remark	145
A	Vanilla VO Results on the VO Test Datasets	146

List of Figures

1.1	Examples of Changing Places	3
2.1	Visual Odometry Overview	13
2.2	Visual Odometry Pipeline	15
2.3	Planar Relative Reference Frames	16
2.4	Continuous Relative Frames of Reference for Unobservable Ego-Motion	17
2.5	Complete System	19
2.6	Pinhole Camera Model	20
2.7	Example of Distortion Correction and Stereo Rectification	23
2.8	Stereo Camera Model	24
2.9	The FAST Corner Detector	26
2.10	Stereo Matching Example	27
2.11	Temporal Matching Example	28
2.12	Huber Weighting	34
2.13	Coordinate Frame Conventions	39
3.1	Example of DGPS Drift	45
3.2	Screen Shot from Synthetic World	46
3.3	Data Collection Vehicles	47
3.4	The Begbroke VO Test Dataset	49
3.5	The Kitti VO Sequence 06	50

3.6	The Kitti VO Sequence 00	51
3.7	Begbroke Repeated Loops	53
3.8	New College Dataset Route	54
3.9	Screenshot of the VO application	57
4.1	VO Testing: Vanilla Results	60
4.2	VO Testing: Vanilla VO Timings	63
4.3	VO Testing: Number of Features, Synthetic and Kitti06	67
4.4	VO Testing: Number of Features, Begbroke and Kitti00	68
4.5	VO Testing: Number of Features, Timing	70
4.6	VO Testing: Feature Distribution Example	71
4.7	VO Testing: Feature Distribution	72
4.8	VO Testing: Rethreading	74
4.9	VO Testing: BRIEF Pattern	76
4.10	VO Testing: BRIEF	77
4.11	VO Testing: Sub-Pixel Refinement	81
4.12	Results on Kitti Vision Benchmark Suite	83
4.13	Example SCISYS Images	84
4.14	VO Results on SCISYS Quarry Dataset	86
4.15	Constant Time vs Naive Implementation	89
5.1	Variety In Scene Appearance	92
5.2	Overview of Lifelong Navigation Approach	98
5.3	Experiences	100
5.4	Experience Localisers	101
5.5	Experience Graph Structure	106
5.6	Overhead of Begbroke Results	109
5.7	Begbroke Northern Road	110

5.8	Begbroke Eastern Road	110
5.9	Begbroke Car Park	111
5.10	Begbroke Overhanging Trees	112
5.11	Time Ordered Results for Begbroke	113
5.12	Weather Conditions	115
5.13	Examples of Localisation Failure on Run 4	115
5.14	Examples of Localisation Failure on Run 38	116
5.15	Time Ordered Results for Begbroke, $N = 2$	118
5.16	Time Ordered Results for Begbroke, $N = 3$	119
5.17	Single Maps Performance	121
5.18	Greedy Reordered Results	123
5.19	Impact of Different BRIEF Descriptor Lengths	125
5.20	Varying BRIEF Descriptors	126
5.21	No Discovery Graph Structure	127
5.22	Comparison of Graph Connectivity, $N = 1$	129
5.23	Comparison of Graph Connectivity, $N = 3$	130
5.24	Experience Utilisation	131
5.25	Sensitivity to Lost Threshold	134
5.26	Sunny vs. Overcast Conditions	135
5.27	Timing Performance	137

List of Tables

3.1	VO Test Datasets - Summary	47
4.1	VO Testing: Other Vanilla Statistics	65
4.2	VO Testing: Number of Features	69
4.3	VO Testing: Feature Distribution	73
4.4	VO Testing: Rethreading	75
4.5	VO Testing: BRIEF	79
4.6	VO Testing: Sub-Pixel Refinement	81
4.7	Kitti Vision Benchmark Suite Ranking Table	82
5.1	Experience Based Navigation Results for choices of N	119
5.2	Graph Connectivity Results	131
5.3	Sensitivity to Lost Threshold	133
A.1	VO Testing: Vanilla Performance, Synthetic	146
A.2	VO Testing: Vanilla Performance, Kitti06	147
A.3	VO Testing: Vanilla Performance, Begbroke	147
A.4	VO Testing: Vanilla Performance, Kitti00	147

Chapter 1

Introduction

1.1 Motivation

To achieve lifelong autonomy, robotic systems require the ability to navigate accurately and reliably for long periods of time. When operating over such time scales they will experience changes in the environment, and must be able to deal with this seamlessly - we see this as a big challenge. Current state of the art solutions can perform tasks such as autonomous retro-traverse of a robot against a previously taught route [36], however these are only feasible if the prior map captures the current state of the world. In reality we find many commonly occurring events that humans deal with naturally can disrupt these autonomous systems, as the appearance of the world changes sufficiently for current algorithms to fail. Such change can come from many sources: dynamic object/scenes, lighting conditions, time of day, weather and seasonal shifts. An autonomous robot must be equipped with the ability to deal with these regular events. Self-driving cars that can only be used on overcast autumn afternoons, or autonomous agricultural vehicles that can only operate before noon on sunny days are underwhelming. If we expect true autonomy from our robots this problem must be addressed.



Figure 1.1: Long-term autonomy requires the ability to navigate in changing environments. Attempting to build a single monolithic map from all encounters of the world is difficult. We propose a new approach that models the world as a set of independent experiences. Given this set of images, our approach would likely build three experiences covering the overcast, sunny and evening conditions.

1.2 Lifelong Navigation

Concretely, this thesis is concerned with improving the navigation performance of a robotic system over long periods of time in changing environments.

Before the issue of longevity is discussed, we must first address the issue of accurate navigation. In this thesis we choose an approach called stereo visual odometry [81] which, using a sensor with two cameras separated by a fixed baseline (stereo), estimates the ego-motion of said sensor by analysing its images. This technique has been well studied recently with several variations proposed. We begin by describing

our implementation and provide detailed testing, results and analysis. The choice of sensor is unimportant for the lifelong navigation approach described next, although camera-based systems are attractive as they are passive sensors that capture a rich, dense representation of the world while being low weight, low power and relatively small.

Armed with a navigation system that can perform exploration and localisation, we wish to deploy it for long-term use. As the robot continuously navigates, the traditional approach would continually match and average previous observations to new measurements and forever accrue new landmarks into a single map. It would also force all encounters of a place to live in the same frame of reference. But is this really the best way forward? Is it really best to average observations as morning turns to afternoon, as winter drifts to summer? Should we be attempting to build a single monolithic map that includes everything we have ever seen? Do we really need globally consistent metric maps? We argue that we do not need these things, and so should not attempt them.

We describe a possible solution to the problem of lifelong navigation through the use of experiences. Different locally metric representations of the world are stored independently and only linked topologically. This neatly side steps the issue of data fusion and consistency. We will show that by using this concert of distinct experiences together can achieve long-term navigation.

1.3 Contributions

This thesis makes the following contributions:

- We lay out a new framework for the problem of lifelong navigation which is agnostic to sensor choice. The approach is based on *experiences* and incrementally learns a model whose complexity varies naturally with that of the

environment.

- We describe in detail a state of the art stereo visual odometry system. The system is tested on several datasets to enable analysis of key components and discussion.
- We carry out extensive testing of the new lifelong navigation approach using visual odometry. Testing is performed on a new dataset, which to the best of our knowledge contains the most repeated traverses of an environment over an extended period of time.
- We present The New College Dataset - a public benchmark dataset containing vision and laser data.

1.4 Thesis Structure

1.4.1 Structure

In Chapter 2 we begin with a high level overview of stereo visual odometry. At the beginning we give a brief history of previous systems and then describe our approach, with some background on the various sub-components used.

Chapter 3 sets out several datasets we require in future chapters, detailing why we have chosen these datasets. We also briefly describe some of the engineering undertaken to produce the complete systems used throughout this thesis.

To ensure we are getting the most from our visual odometry system we perform an extensive analysis of several of its key components in Chapter 4. We adjust the algorithm and test on a set of datasets with different characteristics and ground truth data to justify our design choices and select appropriate parameters.

Chapter 5 presents a new lifelong navigation approach which leverages the previously described visual odometry system. We break away from the typical approach

to mapping in the long-term and do not attempt to perform data fusion, instead allowing representations to exist independently of one another. Extensive results are presented on data collected regularly, over multiple months at different times of day and in different weather conditions.

Closing comments and possible future research directions are given in Chapter 6.

1.4.2 How to read this thesis

When read in its entirety this thesis tells a story about an accurate navigation system (visual odometry), its thorough testing and finally a new lifelong navigation paradigm. However, for readers who are familiar with visual odometry systems Chapter 2 could be skipped on first reading. Equally the design and parameter choices discussed in Chapter 4 are specific to the visual odometry system, and not required for the navigation approach in Chapter 5. The majority of the new work is presented in Chapter 5.

1.5 Publications and Exploitation

The Dataset described in Chapter 3 was published as “The New College Vision and Laser Data Set” in the International Journal of Robotics Research (IJRR) in May 2009 [99].

The stereo visual odometry system described in Chapter 2 and analysed in Chapter 4 was field tested by SCISYS PLC¹ and is currently being evaluated for use by the European Space Agency (ESA).

The lifelong navigation approach described in Chapter 5 was presented as “Practice Makes Perfect? Managing and Leveraging Visual Experiences for Lifelong Navigation” at IEEE International Conference on Robotics and Automation (ICRA)

¹<http://www.scisys.co.uk>

in St Paul, Minnesota in May 2012 [16]. Extensions to this work were published as “Continually Improving Large Scale Long Term Visual Navigation of a Vehicle in Dynamic Urban Environments” at IEEE Intelligent Transportation Systems Conference (ITSC) in Anchorage, Alaska in September 2012 [15].

Chapter 2

Stereo Visual Odometry

2.1 Introduction

Visual Odometry (VO) is the process of computing the ego-motion of a camera from consecutive images. VO is the subset of a wider problem: estimating the transformation between camera poses and the underlying three-dimensional scene structure. This problem is referred to as Structure from Motion (SfM) and was largely developed, first by the photogrammetry and then the computer vision communities [105, 42].

In the field of robotics, when discussing VO, we normally have a few assumptions and expectations. Firstly, we are typically interested in an image sequence from a single calibrated camera unit. This means we do not need to include camera parameters into the estimation problem. Secondly, we are often more interested in the camera trajectory, rather than the underlying scene structure - although as we will discuss later, taking the time to improve the structure estimate leads to an improved motion estimate. Thirdly, VO implies a system that runs fast enough to be useful to a moving robot. Depending on the application this can be anything from 1-2Hz, all the way up to the frame rate of the camera, typically 20-30Hz. 4

The first VO systems began to appear in the 1980s, but only since the early 2000s has its usage become widespread in the area of robotics and beyond. Perhaps the most famous use case of VO is on NASA’s Mars Exploration Rovers, Spirit and Opportunity [64, 13], and the more recent Mars Science Laboratory - Curiosity [74] - where it is used as a supplement to the dead reckoning estimates. It has also been used in unmanned aerial [50, 40, 108] and underwater vehicles [26]. Its popularity can be attributed to several factors. Recent algorithm and hardware improvements have enabled systems to run on standard consumer-grade computer hardware found on robots. Cameras can be relatively cheap, lightweight, and low powered, which are desirable properties for robotic sensors, while passively capturing a rich representation of the world. The quality of the motion estimation from VO is better than that of wheel encoders as wheels tend to slip [90], and is comparable to laser based systems. It can also be combined with other sensors such as Inertial Measurement Units (IMU) [49, 85, 109], sun sensors and inclinometers [58] to further improve performance.

Approaches to VO and localisation can be divided into two categories: (i) dense approaches which use the whole image, and (ii) sparse methods which extract a set of repeatable interest points. The first approach perform matching using all of the image data, these techniques have been explored but they are computationally expensive [17]. Furthermore, dense approaches are memory intensive, so the size of the map area is restricted. In [66] localisation is performed against a textured occupancy grid using full image matching and requires 500MB of memory for a map covering a hallway environment. A recent real time dense reconstruction approach using both cameras [75] and Microsoft’s Kinect [76] has been shown to work in real-time at modest scales - up to $7m^3$ using a graphics processing unit (GPU). Dense optical flow has been used previously but assumptions or additional information is required, for example in [18] it is assumed the rover only observes a planar ground

structure. As a result most modern VO systems use sparse points as they are computationally cheaper and do not require prior knowledge of the scene structure - this is the approach we take. Memory and storage requirements for the output are also lower - our VO system requires less than 40MB/km.

VO can be performed with a camera that only has one sensor (monocular camera) [23, 28]. The difficulty with using monocular cameras is that the scale of the scene cannot be recovered from the images alone. This means the camera motion is only known up to a scale factor, or another sensor such as an IMU is needed to estimate the motion to scale [82]. If a camera with two sensors (stereo camera) is used, and the distance between the two sensors is known and fixed, the scene structure and camera motion can be recovered with scale, resulting in metric motion estimation. Due to this simplification, we will focus solely on a stereo visual odometry approach, although many of the steps would also be found in a monocular based approach. When discussing stereo based systems, we often refer to the pair of images as a frame.

This chapter aims to give a high level overview of the stereo VO pipeline used in this thesis and introduce some of the previous work in this area. Analysis and discussion for key parts of the process are presented in Chapter 4. For a recent review on the wider field of VO we refer the reader to [90, 34].

2.2 Background

Although it was Nistér’s 2004 paper [81] which coined the term Visual Odometry, the idea of motion estimation from images was first described by Moravec in 1980 [73]. Many of the pieces found in today’s modern VO pipelines were outlined by Moravec, who presented a cart equipped with a TV camera and a transmitter. Images were broadcast to a computer which undistorted them, extracted interest

points, estimated depths, tracked points over several frames and computed a motion estimate. The cart could navigate a simple obstacle course, stopping every meter and capturing multiple images with a single camera on a sliding track before processing and moving on. Later Matthies and Shafer [67, 68] used improved uncertainty handling in the motion estimation step to achieve better performance. This approach was later used on NASA’s Mars Rovers, Spirit and Opportunity.

In 2004 Nistér *et al.* [81] presented an approach they called Visual Odometry. It contained several significant differences from previous work and was the first real-time system, being able to run at 13Hz by taking advantage of the MMX instruction set to speed up image processing. Unlike earlier work which computed the ego-motion by matching the triangulated point clouds of two stereo frames - a 3D to 3D registration problem - they solved the 3D to 2D problem, matching the previous triangulated 3D point cloud to the current 2D image measurements. This greatly improved the estimate as the biggest ambiguity in the triangulated position is depth, which can severely harm a 3D to 3D estimate, but has relatively little impact in a 3D to 2D scheme. They also used the now ubiquitous RANSAC [33] for outlier rejection. The complete system was the first to perform in real-time over several hundred meters in outdoor environments.

Engels *et al.* showed that bundle adjustment was a viable option for real-time systems [29] and this was incorporated by Konolige *et al.* [53, 1] who used the addition of sparse bundle adjustment to improve the accuracy of their large scale VO system. The idea of optimising a recent local window was also demonstrated by Sibley *et al.* [98] which later evolved into the idea of *relative* bundle adjustment [97]. The relationship between VO and Simultaneous Localisation and Mapping (SLAM) is discussed in Section 2.3.1.

More recently several deviations and improvements have been presented. Mei *et al.* [71] describe various improvements to the image processing pipeline such

as feature distribution and sub-pixel refinement for improved measurement quality. Strasdat *et al.* [100] perform a double window optimisation with the inner region solving the full bundle adjustment problem and the outer solving the computationally simpler pose graph problem. Furgale *et al.* [36] used a GPU implementation of SURF in their VO system to enable the teach and repeat ability of their rover. This work is discussed further in Chapter 5. Very recently efforts have been made to further improve the performance of VO by modelling the bias of distant landmarks [86] or by assuming the remaining error in VO is due to a bias caused by insufficient system modelling and learning a correction factor [25].

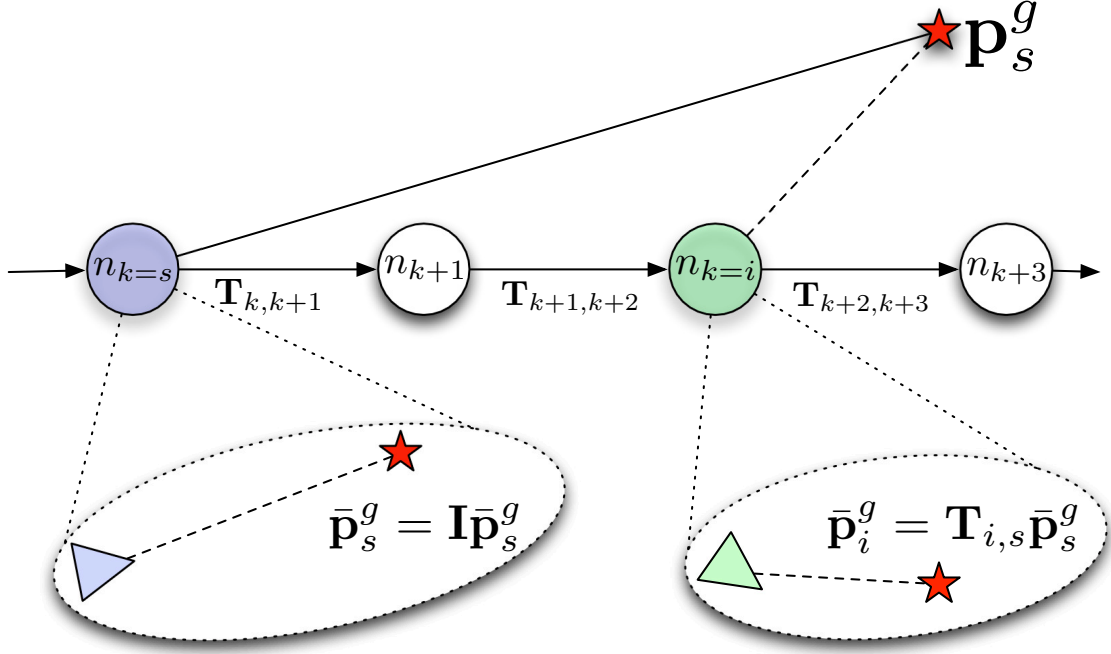


Figure 2.1: A sketch of the VO system output. Images are represented by nodes n_k . The 3D position of features $\mathbf{p}_s^g$ are called landmarks, with global index g and stored relative to frame s . Each node stores a set of landmark observations $\mathbf{Z}_i$. Landmarks are projected into node n_i via the transform $\bar{\mathbf{p}}_i^g = \mathbf{T}_{i,s} \bar{\mathbf{p}}_s^g$. $\bar{\mathbf{p}}$ indicates the homogenous form of $\mathbf{p}$. By applying the transform between nodes to landmarks, we are able to project them into different images. In this example we see landmark relative to camera frame $n_{k=s}$ and $n_{k=i}$. Each ellipse captures how landmarks appear to that node.

2.3 Overview

We now provide a general overview of our stereo VO approach, which is fairly typical of current state of the art approaches, and introduce some notation. A graphical view of our approach can be seen in Figure 2.1. We process a sequence of stereo frames $\mathcal{F}^k = \{\mathcal{F}_0, \dots, \mathcal{F}_k\}$ indexed by k . For the first frame $\mathcal{F}_0$ we extract a sparse set of points from each image. We denote pixel locations in the left image by $\mathbf{z}_L = [u_l, v_l]^T$ and in the right image by $\mathbf{z}_R = [u_r, v_r]^T$, where the subscript on u and v indicate the camera. $\mathbf{z}_L$ and $\mathbf{z}_R$ are referred to as “features” and each are additionally characterised by a descriptor which is a function of the local image

region.

Pairs of features from the left and right image are matched to produce a stereo measurement, which takes the form

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_L \\ \mathbf{z}_R \end{bmatrix} \quad (2.1)$$

and by using the known properties of the stereo camera is triangulated to produce a 3D position of the features called a landmark, $\mathbf{p}^g = [x, y, z]^T$, where g is a globally unique id.

For all subsequent frames, we attempt to match features in the latest frame to landmarks that were successfully matched in the previous image. Once we have these data associations, we can compute the 6 degree of freedom (DoF) motion of the camera between k and $k - 1$. This is stored as a 4×4 transformation matrix with the form

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \boldsymbol{\rho} \\ \mathbf{0} & 1 \end{bmatrix} \quad (2.2)$$

where $\mathbf{R}$ is a 3×3 rotation matrix ($\mathbf{R}^T = \mathbf{R}^{-1}, \det(\mathbf{R}) = 1$) and $\boldsymbol{\rho}$ is a 3×1 translation vector. $\mathbf{T}$ can be parameterised by a 6×1 vector $\mathbf{t} = [x, y, z, \theta_r, \theta_p, \theta_q]^T$ where x, y and z are the translation and θ_r, θ_p and θ_q are roll, pitch and yaw respectively. We detail how transformation matrices are built in Section 2.6.4.

We store the camera motion in a graph, $G_{vo} = (N_{vo}, T_{vo})$. Nodes, $n_k \in N_{vo}$, represent stereo frames at time k . Edges, $\mathbf{T}_{i,j} \in T_{vo}$, describe the transform between two nodes i and j , where $\mathbf{T}_{i,j}$ represents the motion of the camera from j to i . How these transforms are constructed is described in Section 2.6.4. We use a *relative* reference frame, details of which are discussed in Section 2.3.1. For this we store landmarks relative to frame s and observe them in frame i . To move a landmark

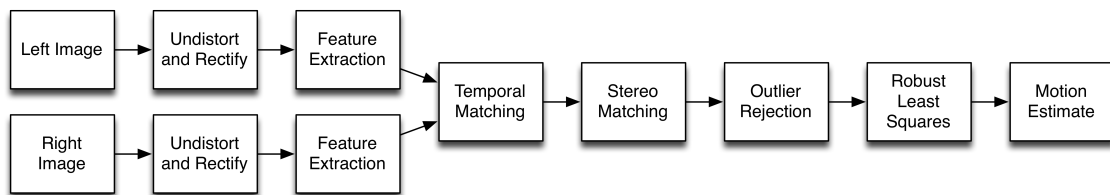


Figure 2.2: The stereo VO pipeline. Each image is corrected and features are extracted. These are matched to landmarks from previous frames, then stereo matched to create new measurements. Outliers are removed before the solution is passed to a robust least squares step which produces an accurate pose estimate. Each of these steps are discussed in detail in Section 2.4 and Section 2.5.

from one frame to another we apply the following transformation

$$\bar{\mathbf{p}}_i = \mathcal{T}_{i,s}(\mathbf{p}_s) = \mathbf{T}_{i,s}\bar{\mathbf{p}}_s \quad (2.3)$$

where $\bar{\mathbf{p}}$ is the 4×1 homogeneous representation of $\mathbf{p}$ and $\mathcal{T}_{i,s}(\cdot)$ is a function which returns the concatenated sequence of transforms between i and s - $\mathbf{T}_{i,s}$. We also denote observation of landmark g in frame i as $\mathbf{z}_i^g$. G_{vo} grows incrementally as new frames arrive. To prevent unnecessary creation of new nodes and edges, we adopt a keyframe approach whereby a new node is only added to the graph if the motion delta between two frames is above some threshold or the number of trackable features changes significantly¹. This prevents unnecessary graph growth while the camera is stationary or moving slowly. The pipeline of the VO process is shown in Figure 2.2, the blocks in this diagram are discussed in detail in Section 2.4 and Section 2.5.

Note that this approach assumes the world is static. If tracked points are associated with a dynamic object, these can induce false motion. Consider the case of sitting on a stationary train in a platform. Looking out of the window is another stationary train. If that train starts to move away, it is often difficult to tell who

¹Typically new keyframes are triggered by the motion delta threshold, but the operating environment may result in many new features for a small motion change. Including a feature count threshold prevents the system attempted to run on a sub-optimal number of features.

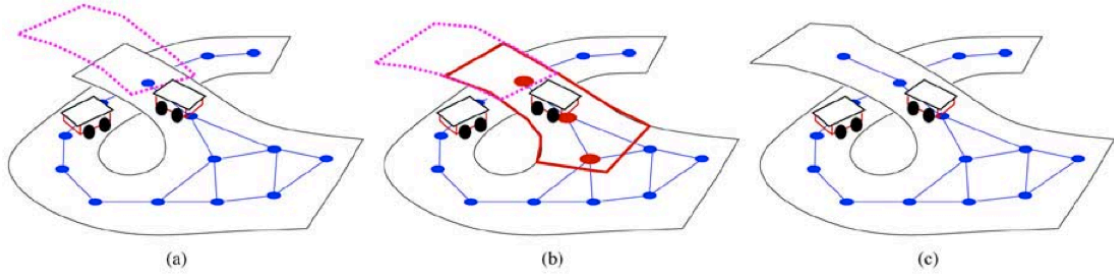


Figure 2.3: Howard used the idea of locally, but not globally consistent sub-maps to side step the problems imposed by a single globally correct Euclidean map in the classical SLAM problem. An example of this idea is depicted in the figure above. As the robot explores from (a) through to (c) it allows multiple representations of the same place to be present in the map, and does not enforce consistency between them. The new purple section of the map was observed previously, but is only appended to the latest local sub map. Image from [46].

is actually moving. To a VO system, which assumes the world is static, our train would appear to be moving, even though we are stationary. This is perhaps an extreme example. In reality, some amount of dynamic scene motion can be tolerated through the use of robust estimation techniques, but eventually the motion estimate will degrade in the face of dynamic objects.

2.3.1 VO, SLAM and Frames of Reference

The classical Simultaneous Localisation and Mapping (SLAM) problem attempts to maintain a globally consistent robot position and associated map containing observed landmarks [27]. One of the motivations for retaining the map is so that when the robot returns to some previously visited location after a period of exploration (mapping) it can localise against its map. This moment is called *loop closure* as the robot has completed a loop. Despite the desire for a globally accurate pose, due to estimation errors this is virtually impossible to achieve in reality, so when a loop closure happens the robot will not have the same metric position in its map representation. This motivated the idea of loop closure detection, which can be thought of as topological SLAM. The problem is simply: given a new measurement, am I in

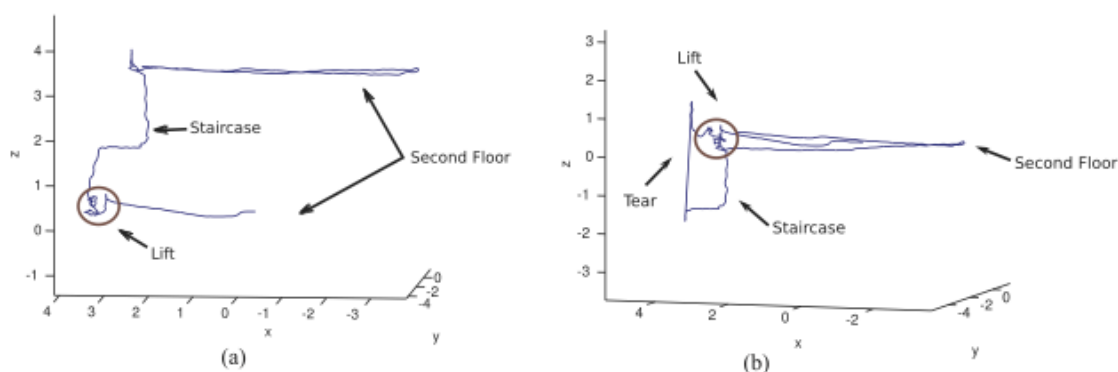


Figure 2.4: Sibley *et al.* extended the idea of mapping in a manifold to a continuous representation in 6 DoF. Here the robot started on the second floor before travelling down a staircase. The robot returns to the second floor via the elevator. Enforcing global consistency at loop closure produces a tear in the map due to the unobserved ego-motion of the camera in the elevator - (b). Plotting this gives the impression the robot was somehow teleported - indeed it was, by the elevator. When adopting a relative approach, plotting in a single globally correct Euclidean frame and expecting the result to look correct is wrong. The assumption is that locally the trajectory will be accurate, but globally it is inaccurate. The relative representation deals with the elevator teleportation problem more elegantly - (a) - note there is no tear. Image from [97].

a new place or somewhere I have been before? This problem, sometimes called place recognition, has received significant attention in recent years [20, 43, 8]. The state of the art, by leveraging techniques from large scale image retrieval has been shown to work on datasets covering 1,000km [22]. While the use of place recognition helps in the SLAM problem, it does not solve it, as the idea of global consistency is still present.

Most robotics tasks do not require a *globally* consistent map, if only the *local* region is correct, tasks such as localisation, autonomous retro-traverse, path planning and obstacle avoidance can still be carried out. Motivated by this fact, Howard proposed the idea of operating on a manifold [45]. In this framework, the local map is metrically correct, but allowed to be globally inconsistent. Many points in the map can refer to the same physical place in the world, potentially without knowledge of each other. Loop closure becomes optional and the requirement for global consistency is relaxed.

tency is removed. Howard used this approach to build 2D planar sub-maps with overlapping regions and no global consistency for multi robot mapping - Figure 2.3. Furgale and Barfoot used the idea of sub-maps in 3D for autonomous retro-traverse [35]. Regardless of the fact that their global pose estimate was completely wrong, they were still able to move along the local sub-maps and drive the previously traversed route. Sibley *et al.* dropped the sub-mapping requirement and describe a *continuous relative representation* [95]. When in exploration mode, only the window affected by the new measurements is optimised, which is typically small. At loop closure this window size increases, but is still relatively small as the impact of the loop closure information on regions far away is beyond machine precision. This approach allows the robot to elegantly deal with unobservable ego-motion, an example of travelling in an elevator is shown in Figure 2.4.

In contrast to SLAM, VO is purely interested in accurate frame to frame estimation, with the map being something of a by-product. In this thesis we are not interested in tackling the SLAM problem and do not assume a single Euclidean frame of reference. In Chapter 4 we analyse the performance of an open loop VO system. In Chapter 5 we combine this with a separate process that performs pure localisation to produce a large scale navigation system. The use of a *relative* frame of reference is important and discussed further in this chapter.

2.3.2 A Complete System

We refer to the whole process of feature matching and camera motion estimation as the “front-end”. In early systems this was all that could be managed in real time due to hardware constraints. Bundle adjustment (BA) is a way to optimise a set of poses and landmarks, where the number of poses is typically larger than one, i.e. it is beyond what the front-end is solving [105]. It has since been shown that if bundle adjustment can be included in the VO, typically optimising a small number

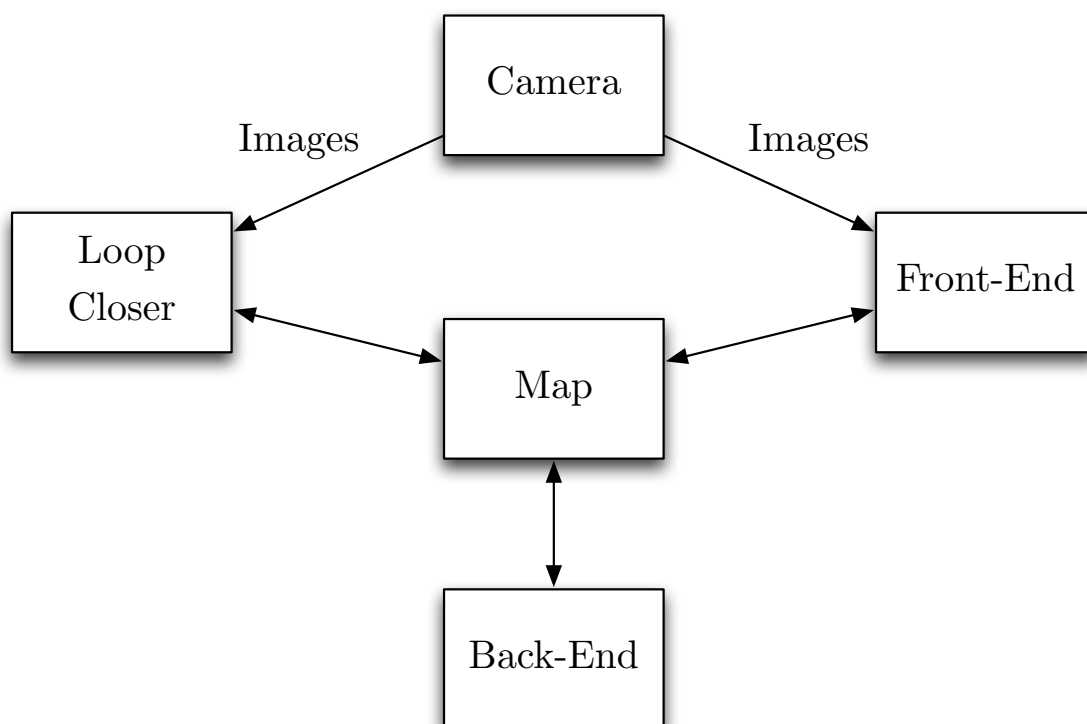


Figure 2.5: A sketch of a complete SLAM system. Images are passed to both a front-end for local metric pose estimation and to a loop closer for topological place recognition. A back-end continually optimises the underlying map. Such systems have been presented in [94] and [55].

of recent keyframes (and associated landmarks), the quality of the estimation can be improved [53]. This can be done either in serial or parallel with the front-end. Running the BA solver in a separate thread to the front-end frees both processes from the constraints of the other, the front-end won’t skip frames by waiting for the BA solver to finish and the BA solver isn’t restricted to a small time budget, as it is a relatively expensive process. This was popularised in the Parallel Tracking and Mapping (PTAM) system [51]. We refer to this optional extra solver as the “back-end”. Figure 2.5 sketches a complete system layout. Images from the camera are fed to both the front-end and a loop closer. The front-end runs continuously on the image stream providing pose updates. Occasionally (depending on the motion of the camera) new keyframes will be incorporated into the map, along with their

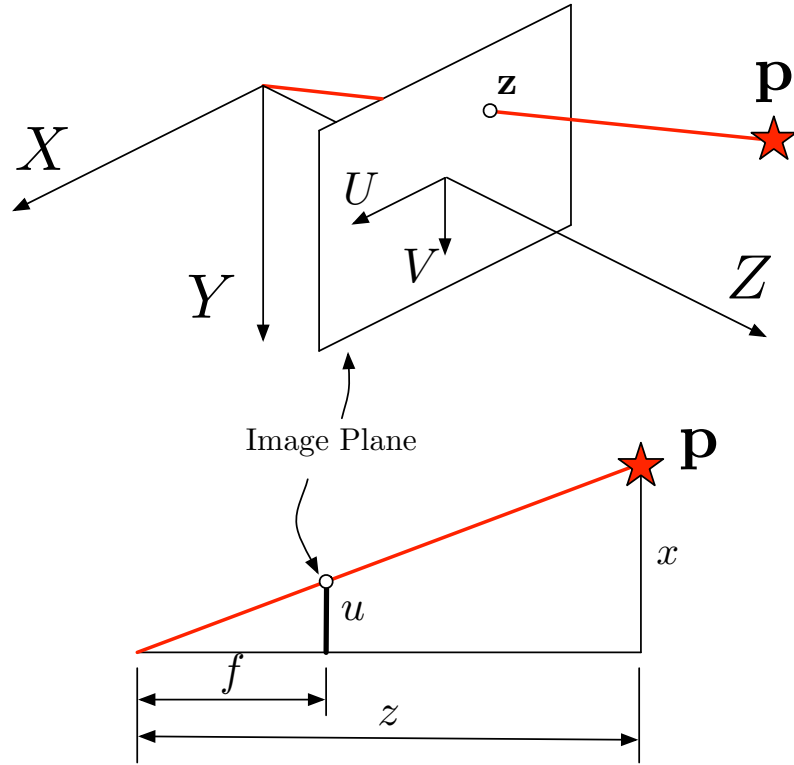


Figure 2.6: The pinhole camera model. The world point $\mathbf{p}$ projects to the image point $\mathbf{z}$. The pixel position can be computed via similar triangles.

associated landmarks. The loop closer opportunistically adds additional constraints to the map when it detects the robot has returned to a previous place. The back-end continually refines the map with a more computationally expensive solver.

2.4 Stereo Cameras

We begin with a brief overview of stereo cameras. By using stereo instead of monocular cameras, we can simplify the data association and estimation problems. We start with the standard model for a single camera and then cover the stereo model.

2.4.1 Camera Models

A camera projects world points in $\mathbb{R}^3$ to points in $\mathbb{R}^2$ on the image plane. To estimate the motion of the camera, we require a model of how this projection happens.

Assuming we have an ideal camera we can use the standard pin-hole camera model, shown in Fig 2.6. We define a camera frame, where the origin is placed at the optical centre of the camera. A point $\mathbf{p} = [x, y, z]^T$ in the camera frame projects to a pixel location on the image plane $\mathbf{z} = [u, v]^T$. For a known camera focal length, f the pixel location can be computed using similar triangles:

$$\frac{u}{f} = \frac{x}{z} \quad (2.4)$$

$$u = \frac{fx}{z} \quad (2.5)$$

Applying a similar process to y , the pixel position becomes:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{fx}{z} \\ \frac{fy}{z} \end{bmatrix} \quad (2.6)$$

Unfortunately equation (2.6) is non-linear. However, by adopting a homogeneous representation, the projection becomes linear. This is achieved via the following matrix:

$$\begin{bmatrix} u' \\ v' \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = [\mathbf{K}|\mathbf{0}]\bar{\mathbf{p}} \quad (2.7)$$

where $\mathbf{K}$ is the 3×3 camera calibration matrix. The details of $\mathbf{K}$ are discussed below, but the bottom row will stay the same, meaning $w = z$. The inhomogeneous form of $\mathbf{z}$ can be recovered by:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{u'}{w} \\ \frac{v'}{w} \end{bmatrix} \quad (2.8)$$

$\mathbf{K}$ typically includes some extra details about the camera. The origin of the image plane does not normally align perfectly with the optical axis, so an origin offset can be applied via $[u_0, v_0]$. This location is the principal point. If the camera has non-square pixels, a scaling of the focal length is needed. This can be captured by having different focal lengths, f_u and f_v . Finally a skew parameter s can be added if the image axes are not perpendicular. This is typically zero for most cameras. $\mathbf{K}$ then takes the form:

$$\mathbf{K} = \begin{bmatrix} f_u & s & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.9)$$

This collection of variables are referred to as the intrinsic camera parameters, as they affect how points in the camera frame project onto the image plane. It is possible to estimate them “online”, as part of a wider estimation problem [105], but it is common to know them *a-priori*. Camera calibration can be carried out by observing a known pattern and solving for $\mathbf{K}$, tools that make this process automatic exist [9]. Some manufacturers provide this information with the camera.

2.4.2 Image Distortion Correction and Rectification

In reality camera lenses do not obey the pinhole model described in Section 2.4.1. They typically exhibit some form of distortion caused by the manufacturing process. The calibration that recovers the parameters of $\mathbf{K}$ can also be used to compute the parameters of a mapping function that turns a distorted camera image to one that obeys the pinhole camera model. This is known as distortion correction.

In the case of stereo cameras, it is also useful to align the left and right images so the row in the left image corresponds to the same row in the right image. This process is called image rectification. When searching for correspondences in stereo



(a) Raw image from left camera



(b) Raw image from right camera



(c) Distortion corrected and rectified left image



(d) Distortion corrected and rectified right image

Figure 2.7: These four images show the before and after effects of distortion correction and stereo rectification on a pair of images captured from a stereo camera. After the mapping is applied, point features appear on the same row in each image. This is perhaps most noticeable for the foreground structure. Each image also obeys the pinhole camera model.

pairs, rectification vastly simplifies the search region to a 1-D line search in u , as feature points will share the same v coordinates.

For stereo cameras which are fixed together rigidly such as the Point Grey Bumblebee2, the distortion correction and rectification only needs to be computed once. Further, these mappings can be combined into a single look-up table for each lens to enable fast processing. After rectification we can model our stereo camera as two fronto-parallel pinhole cameras separated by a distance b , known as the baseline. This model is shown in Figure 2.8. From this point on we will assume our stereo im-

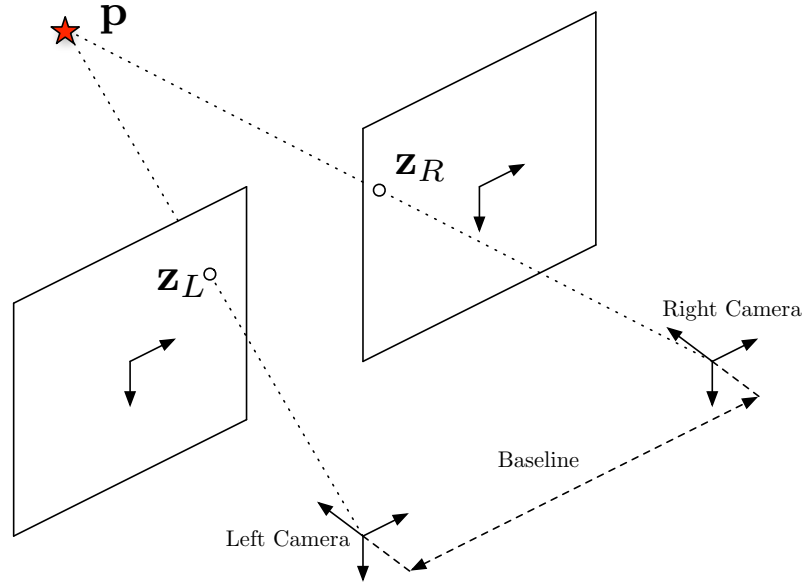


Figure 2.8: The stereo camera model, with two fronto-parallel cameras separated by a known baseline, b . The 3-D position of $\mathbf{p}$ relative to the camera can be recovered via $\mathbf{z}_L$ and $\mathbf{z}_R$.

ages have been distortion corrected and rectified. Colour images are also converted to greyscale, as all future operations do not require colour. We adopt the convention of choosing the stereo camera origin to coincide with the origin of the left camera.

2.5 Front-End Pipeline

The job of the front-end pipeline is to provide data associations between landmarks and the current stereo frame. We will show in Chapter 4 that taking the time to produce high-quality associations can have a dramatic impact on the performance of the system.

2.5.1 Feature Extraction and Description

After the raw images have been corrected, point features are extracted. As we will be searching for these points across multiple frames we want them to be repeatable and unambiguous. Feature extraction and description has been and continues to be

an active area of research, with many algorithms providing a range of choices. One of the best descriptors in terms of performance is the Scale Invariant Feature Transform (SIFT) [62], and to a slightly lesser extent, Speeded-Up Robust Features (SURF) [3] which makes some approximations to SIFT in favour of speed. These algorithms have been used extensively in computer vision but until relatively recently have not seen use in VO implementations as they are too computationally expensive. With the advent of programmable graphics processing units (GPUs) real time performance has been achieved [19], and they have found use in some implementations [35]. The FAST corner detector [87, 88] is popular for real time problems due to its low computational cost, however it is only a feature detector, and thus needs to be combined with a feature descriptor. Binary Robust Independent Elementary Features (BRIEF) [12] and their derivatives [89, 61] are feature descriptors, which offer fast descriptor computation and matching, with performance approaching that of SURF. The choice of the FAST corner extractor combined with a BRIEF-style descriptor has proven a popular choice for real time applications, and it is this combination we use in our system. The results of the FAST corner extractor can be seen in Figure 2.9.

Although FAST only returns integer valued pixel locations, we will show in Chapter 4 that being able to measure sub-pixel feature positions produces a notable increase in accuracy. Thus, as in [71], we refine putative FAST matches using Efficient Second-Order Matching (ESM) [70] to obtain sub-pixel measurements.

D-Nets [106] is a very recent and interesting divergence from the approaches previously described. It breaks away from patch-based matching, instead performing matching via edges between a set of connected nodes spread over the image. It promises better performance than previous approaches such as SIFT, while potentially being faster to compute².

²The released code runs in a few seconds on the CPU, but the authors claim the algorithm is suitable for a GPU implementation which would significantly increase the speed

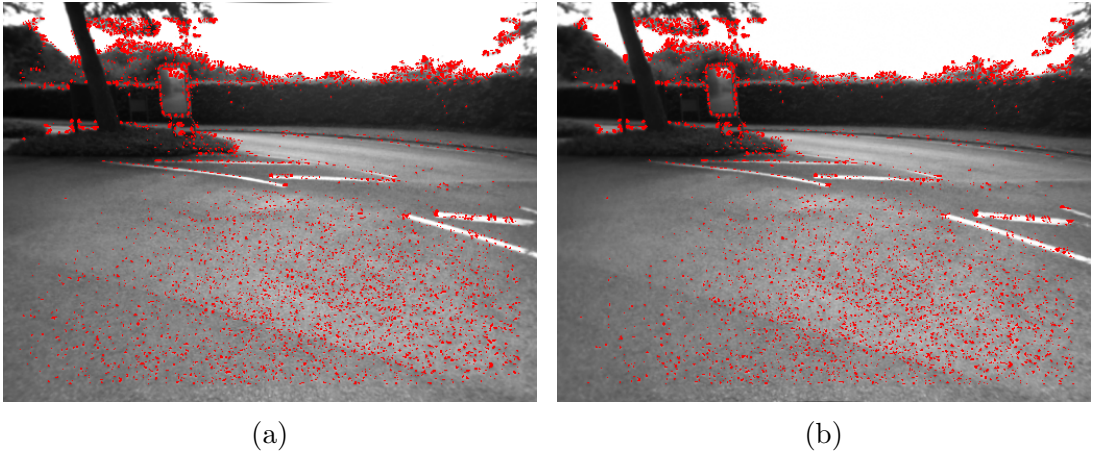


Figure 2.9: FAST Corner extraction on the left (a) and right (b) images of a stereo frame. With a low threshold FAST will produce a large number of corners, which is useful for coverage, but means a subset need to be selected.

2.5.2 Stereo Matching

Stereo matching is the process of finding the same feature in both the left and right stereo images. As we have rectified the stereo frames, given a pixel position in one image, called the reference pixel, we can perform a restricted 1-D line search in the other stereo image to find the matching feature. Matching can be performed using dense stereo techniques, dense matching on a subset of points (such as FAST corners), or descriptor matching. We take the FAST corner positions in the corresponding row and for each one compute the Sum of Absolute Differences (SAD) with respect to the reference pixel position. The best score is returned as a match, and if the row contains no FAST corners failure is returned.

The reference pixel position can come from two sources. Firstly, when spawning new landmarks, given a proposed reference pixel, a stereo correspondence is required to triangulate the landmark. The second case is when a landmark has been temporally matched in one image of a new stereo frame, described in Section 2.5.3, a stereo match is required to complete the process. As we produce sub-pixel feature positions from matching, we search both the `floor()` and `ceil()` rows of the floating point

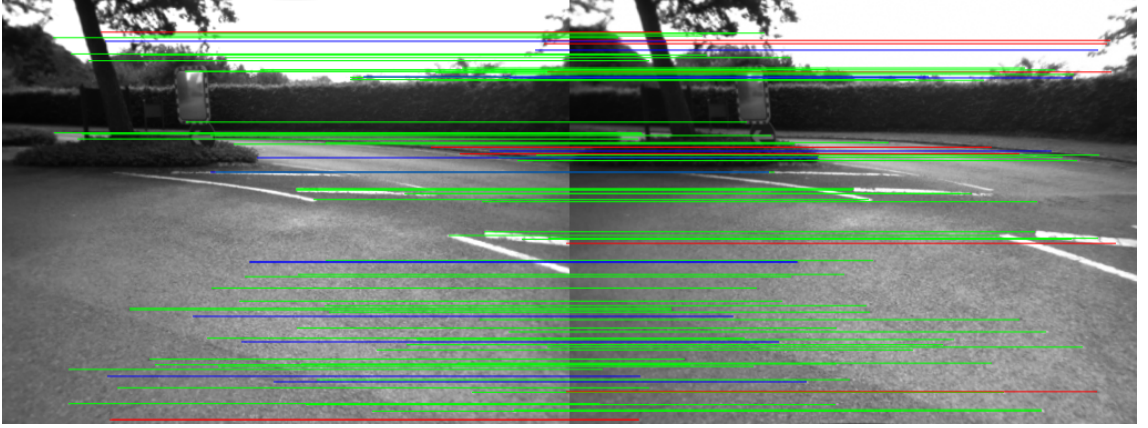


Figure 2.10: Stereo matching finds feature correspondences between two stereo images. Because the frame is rectified, only a 1-D line search is required. Here we show matches from newly created landmarks (blue) and temporal matching (green - inlier measurements, and red - outlier measurements).

value.

2.5.3 Temporal Matching

Finding the feature matches between two stereo frames is called temporal matching. It is not as simple as stereo matching as the camera could have undergone arbitrary motion between the two capture times. This is the step where having a better feature descriptor will produce better data associations, improve the motion estimate and reduce the likelihood of tracking failure. Previous approaches, which could not benefit from the recent BRIEF descriptor, either computed an initial rough motion estimate - via coarse image matching [71], complimentary sensors [13] or a motion model [91] - or assumed the camera would not move very far between frames [79]. Landmarks would then be projected into the new frame and searched for within a fixed window centred on the projected point. The difficulty comes in deciding the size of this window, as there is a trade-off between performance and robustness. If the window is small, matching will be fast, but may be in error. Conversely if the window is large, matching will be more robust but more computationally

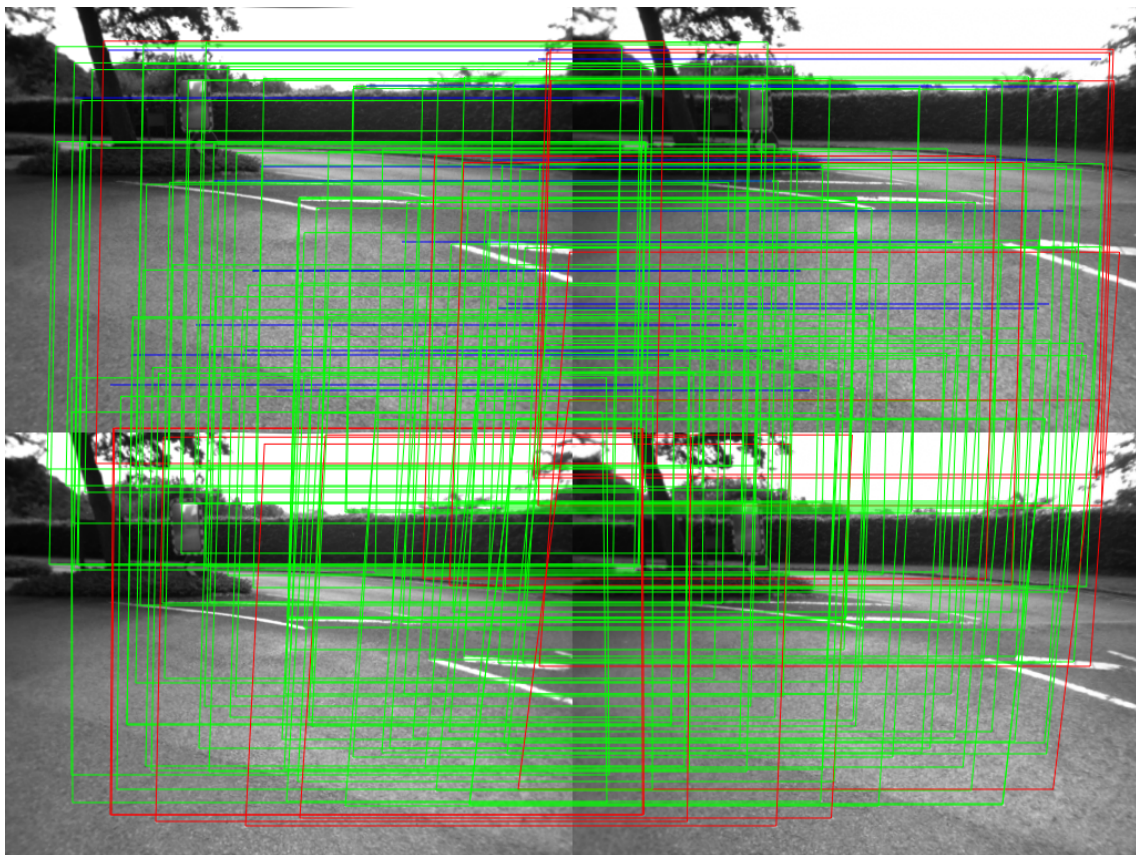


Figure 2.11: Stereo matching finds feature correspondences between two stereo frames. Here we show the current stereo frame (top row) and the previous frame (bottom row). Temporal and stereo matches are shown. Green indicates matches that agree with the motion estimate, while red are outliers. Blue matches are newly created landmarks (current frame only).

expensive. On failure, some implementations will fall back to searching across the whole image. With the advent of BRIEF it is possible to remove this windowing constraint, as searching across the whole image is fast enough for real time operation. This removes the need to compute an initial motion estimate or assume a motion model and selection of an appropriate search window size.

2.6 Motion Estimation

Given a set of correspondences from the matching process, we want to recover the latest motion of the camera. This can be achieved using numerical optimisation

techniques, however these methods perform poorly when presented with data containing significant numbers of outliers - a common by-product of the matching step. To reduce the number of outliers the data is passed to a more robust but less optimal estimator as a pre-cursor to the final optimisation step. This outlier rejection step is covered in the next section, and the final numerical optimisation is discussed in Section 2.6.2.

2.6.1 Outlier Rejection

The stereo and temporal matching steps produce a set of correspondences between image locations. This set can contain significant numbers of outliers - matches that do not imply the true motion of the camera. These can come from dynamic objects in the scene or incorrect data associations. It is important to remove as many of these as possible before the numerical optimisation step.

RANSAC

RANdom Sample Consensus (RANSAC) [33] is an algorithm for robustly estimating a model solution given a set of outlier ridden data. It proceeds by instantiating a model using the minimum number of data points, randomly sampled from the input data. The model is then scored by counting the number of data points which agree with it. This model generation and scoring is repeated until it is likely a good model has been generated. The model and the input data, divided into inliers and outliers are returned. RANSAC finds a correct solution faster for models requiring fewer data points as the likelihood of selecting an outlier in the random set is reduced. It has been shown that scoring the model using an M-estimator can improve the solution at no extra cost [103], this approach is dubbed MSAC (M-estimator sample consensus) [104]. RANSAC works well for removing outliers from the matching step and has been used in several VO implementations [96, 53].

Preemptive RANSAC [79] is designed to produce a reliable model estimate in a fixed time budget. Instead of generating and testing each hypothesis in sequence, preemptive RANSAC generates *all* putative models at the beginning. The models are then repeatedly tested against a small subset of the input data and the worst performing models are removed until a winner is found. This produces a speed up compared to traditional RANSAC as time is not wasted testing all the data against bad models. Active Matching [14] takes a different approach to outliers, by attempting to prevent them occurring in the first place. Feature matching is performed in sequence, with the previous result influencing the search region of the next match and the ordering that maximises the expected mutual information gain at each step. The result being that time spent matching a few initial features quickly reduces the likely positions of future ones, and preventing outliers.

Model Choice

The RANSAC algorithm returns the best model given a set of data. This model choice is important as estimating the transform between two stereo frames can be done multiple ways. Early approaches to VO triangulated the 3D position of feature measurements in each stereo frame, and then found the rigid transform which minimised the error between the two sets of 3D points. This can be minimally solved with 3 data points [44]. An alternative method is to find the transform which minimises the projection of the 3D points in the first frame to the 2D image coordinates in the second frame - the perspective from n points (PnP) problem. This problem has been studied for several values of n [60, 52, 37], but the minimum required number of points is 3, and this P3P case is important for the RANSAC algorithm due to its minimal data point requirement.

When computing the 3D position of a feature, the depth component has the largest error. As noted by Nistér [79], this can harm 3D to 3D registration methods,

but, under the assumption that the inter frame motion is small (reasonable in VO), the depth error has relatively little impact on the solution of 3D to 2D methods. For this reason PnP methods are typically preferred. Motion can also be recovered from 2D to 2D feature correspondences, however this is only up to scale, and the minimum solution requires 5 data points [80].

In our implementation we use a MSAC estimator with a P3P model to produce an initial guess of the camera motion and a set of inlier matches which is passed on to the numerical optimisation.

2.6.2 Numerical Optimisation

RANSAC produces an estimate of the camera motion, but it is only based on 3 data points. We now wish to refine this (and possibly the scene structure - 3D positions of the landmarks) using all the inlier measurements, $\mathbf{z}_i, i \in 1, \dots, M$, to achieve the best possible estimate. We parameterise what we want to solve for in a set of state variables contained in the state vector $\mathbf{x}$. Given a likelihood function

$$\mathcal{L} = \prod_{i=1}^M p(\mathbf{z}_i | \mathbf{x}) \quad (2.10)$$

which represents the probability of obtaining the measurements $\mathbf{z}$ given $\mathbf{x}$; we want to find the estimate $\hat{\mathbf{x}}$ that maximises this function. This is called the maximum likelihood (ML) estimator.

$$\hat{\mathbf{x}}_{ml} = \arg \max_{\mathbf{x}} \prod_{i=1}^M p(\mathbf{z}_i | \mathbf{x}) \quad (2.11)$$

Now if we assume each measurement is corrupted by Gaussian noise

$$\mathbf{z}_i = \bar{\mathbf{z}}_i + \mathbf{w}_i = h_i(\mathbf{x}) + \mathbf{w}_i, \quad \mathbf{w}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_i) \quad (2.12)$$

where $\bar{\mathbf{z}}_i$ is the uncorrupted measurement value and $\mathbf{w}_i$ is a zero mean Gaussian with covariance $\mathbf{P}_i$. $\mathbf{P}_i$ is a $m \times m$ covariance matrix where m is the size of the measurement vector. We make the assumption that every observation has the same covariance and set all $\mathbf{P}_i = \mathbf{P}_C$, where the C subscript indicates a constant covariance. $h(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a prediction function which relates the n dimensional state to the measurements we observe, such that given the true state, $h_i(\mathbf{x}) = \bar{\mathbf{z}}_i$. The likelihood function then becomes Gaussian

$$p(\mathbf{z}_i|\mathbf{x}) = \prod_{i=1}^M \frac{1}{\sqrt{(2\pi)^m \det(\mathbf{P}_C)}} \exp\left(-\frac{1}{2}(\mathbf{z}_i - h_i(\mathbf{x}))^T \mathbf{P}_C^{-1}(\mathbf{z}_i - h_i(\mathbf{x}))\right) \quad (2.13)$$

which is maximised when the exponent is zero, i.e. our prediction matches our observations. Equivalently we can minimise the negative log likelihood, which ignoring the normalising constant becomes

$$\hat{\mathbf{x}}_{ml} = \arg \min_{\mathbf{x}} (-\log \mathcal{L}) = \arg \min_{\mathbf{x}} \sum_{i=1}^M \frac{1}{2}(\mathbf{z}_i - h_i(\mathbf{x}))^T \mathbf{P}_C^{-1}(\mathbf{z}_i - h_i(\mathbf{x})) \quad (2.14)$$

We now write the error for a single measurement as

$$\mathbf{e}_i = \mathbf{z}_i - h_i(\mathbf{x}) \quad (2.15)$$

How $h(\cdot)$ is computed is discussed in Section 2.6.4. Given our set of M measurements we aim to minimise

$$\arg \min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^M \mathbf{e}_i^T \mathbf{P}_C^{-1} \mathbf{e}_i = \frac{1}{2} \mathbf{e}^T \mathbf{P}^{-1} \mathbf{e} \quad (2.16)$$

where $\mathbf{e} = [\mathbf{e}_1, \dots, \mathbf{e}_M]^T$, $\mathbf{P}^{-1} = \text{diag}\{\mathbf{P}_C^{-1}, \dots, \mathbf{P}_C^{-1}\}$ is a block diagonal matrix with $M \times M$ block size and $f(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}$ is a general function to optimise. Equation

2.16 indicates we should minimise the weighted sum of the squared errors to obtain the best estimate of the state. This form is also known as *least squares*, and as we have shown is optimal under a Gaussian likelihood assumption.

If $h(\cdot)$ is linear, $\mathbf{x}_{ml}$ can be computed in one step by solving $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = 0$. However $h(\cdot)$ is typically not linear and so a solution needs to be computed iteratively. In robotics this is typically done with gradient based methods such as Gauss-Newton (GN) and Levenberg-Marquardt (LM). In this work we compute $\mathbf{x}_{ml}$ using GN, and so we now give a brief overview of this approach.

Given an initial guess $\mathbf{x}_0$ we wish to find some increment $\delta \mathbf{x}$ where $\mathbf{x}_1 = \mathbf{x}_0 + \delta \mathbf{x}$ such that $f(\mathbf{x}_1) < f(\mathbf{x}_0)$. To achieve this we take the Taylor series expansion of our prediction function

$$h(\mathbf{x}_0 + \delta \mathbf{x}) = h(\mathbf{x}_0) + \mathbf{J} \delta \mathbf{x} \quad (2.17)$$

where $\mathbf{J}$ is the Jacobian of $h(\cdot)$ and takes the form

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{J} = \begin{bmatrix} \frac{\delta h_1}{\delta \mathbf{x}_1} & \dots & \frac{\delta h_1}{\delta \mathbf{x}_m} \\ \vdots & & \vdots \\ \frac{\delta h_n}{\delta \mathbf{x}_1} & \dots & \frac{\delta h_n}{\delta \mathbf{x}_m} \end{bmatrix} \quad (2.18)$$

Placing the expanded form of $h(\cdot)$ into the least squares solution produces

$$\begin{aligned} f(\mathbf{x}_0 + \delta \mathbf{x}) &= \frac{1}{2} (\mathbf{z} - (h(\mathbf{x}_0) + \mathbf{J} \delta \mathbf{x}))^T \mathbf{P}^{-1} (\mathbf{z} - (h(\mathbf{x}_0) + \mathbf{J} \delta \mathbf{x})) \\ &= \frac{1}{2} (\mathbf{e} - \mathbf{J} \delta \mathbf{x})^T \mathbf{P}^{-1} (\mathbf{e} - \mathbf{J} \delta \mathbf{x}) \\ &= \frac{1}{2} (\mathbf{e}^T \mathbf{P}^{-1} \mathbf{e} - \mathbf{e}^T \mathbf{P}^{-1} \mathbf{J} \delta \mathbf{x} - (\mathbf{J} \delta \mathbf{x})^T \mathbf{P}^{-1} \mathbf{e} + (\mathbf{J} \delta \mathbf{x})^T \mathbf{P}^{-1} \mathbf{J} \delta \mathbf{x}) \\ &= \frac{1}{2} (\mathbf{e}^T \mathbf{P}^{-1} \mathbf{e} - 2 \delta \mathbf{x}^T \mathbf{J}^T \mathbf{P}^{-1} \mathbf{e} + \delta \mathbf{x}^T \mathbf{J}^T \mathbf{P}^{-1} \mathbf{J} \delta \mathbf{x}) \end{aligned} \quad (2.19)$$

Taking the derivative with respect to $\delta \mathbf{x}$ and equating the result to zero yields

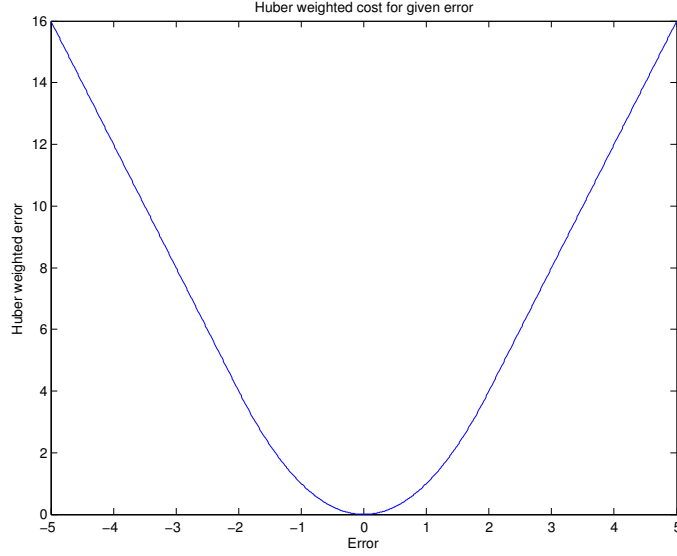


Figure 2.12: Huber’s function for $k = 2$. The Huber cost function is a hybrid between quadratic and L_1 costs functions. Small error values incur a quadratic cost while the cost becomes linear for larger values. This allows for a continuous first derivative while maintaining stability to outliers.

$$\begin{aligned} \frac{\partial f(\mathbf{x}_0 + \delta \mathbf{x})}{\partial \delta \mathbf{x}} &= -\mathbf{J}^T \mathbf{P}^{-1} \mathbf{e} + \mathbf{J}^T \mathbf{P}^{-1} \mathbf{J} \delta \mathbf{x} \\ &= 0 \\ \Rightarrow \delta \mathbf{x} &= (\mathbf{J}^T \mathbf{P}^{-1} \mathbf{J})^{-1} \mathbf{J}^T \mathbf{P}^{-1} \mathbf{e} \end{aligned} \tag{2.20}$$

This is sometimes known as the normal equation. The estimate of $\mathbf{x}$ can be iteratively refined by repeatedly computing and applying the update $\delta \mathbf{x}$. At each stage $\mathbf{J}$ needs to be evaluated at the current guess $\mathbf{x}$. We will discuss the choice of $\mathbf{x}$, $h(\cdot)$ and resulting $\mathbf{J}$ in Section 2.6.4.

2.6.3 Robust Estimation

The least squares approach described above finds a solution which minimises the squared residual error r_i^2 of each measurement:

$$\arg \min_{\mathbf{x}} f(\mathbf{x}) = \sum_{i=1}^M \mathbf{e}_i^T \mathbf{P}_C^{-1} \mathbf{e}_i = \sum_{i=1}^M r_i^2 \quad (2.21)$$

where $r_i^2 = \mathbf{e}_i^T \mathbf{P}_C^{-1} \mathbf{e}_i$. The effect of the squared term means that any outliers in the data - measurements with large errors - will heavily influence the solution of $\mathbf{x}$. As RANSAC does not provide a perfect segmentation of the data it is important to be resilient to these corrupting measurements. This can be achieved through the use of M-estimators, a robust form of estimation [47]. The idea is to re-weight the residual errors depending on their magnitude, reducing the impact of outliers. The function to be minimised then becomes

$$\arg \min_{\mathbf{x}} f(\mathbf{x}) = \sum_{i=1}^M \rho(r_i^2) \quad (2.22)$$

where $\rho(\cdot)$ is the re-weighting function. There are several choices for $\rho(\cdot)$, but Huber's function [47] is popular due to its good performance in most situations [112] and this is the one we use. Huber's function is quadratic for small errors and linear for large errors. This gives first order derivatives while providing robustness to outliers. An example can be seen in Figure 2.12. Huber's function takes the form

$$\rho(r)_{Huber} = \begin{cases} r^2 & \text{if } |x| \leq k \\ k(|r| - \frac{k}{2}) & \text{otherwise} \end{cases} \quad (2.23)$$

where k is a chosen constant. This process is also known as re-weighted least squares.

2.6.4 Choice of State

The use of robust least squares optimisation typically happens in two places. Firstly it is common to perform a (computationally cheap) pose refinement after the RANSAC step in the front-end. This allows the system to include all the inlier measurements

and produce a better estimate. Secondly, in the back-end, a variety of states can be solved for to refine the underlying map. The choice of what is estimated is communicated through the state vector $\mathbf{x}$. In the case of a single pose optimisation (for example in the front-end), the state is simply a 6 DoF delta transform. At the other end of the spectrum the full solution (bundle adjustment) will contain delta transforms and landmark variables. The choice of $\mathbf{x}$ also influences $h(\cdot)$. We present the form of $h(\cdot)$ and its derivative for the simple pose-only, front-end refinement case, under the assumption that all landmarks have been moved so they are stored relative to the previous frame. We will then briefly discuss how these can be extended to form a more complex back-end solver.

Given a landmark $\mathbf{p}_s^g$ stored in frame s and an observation of g from frame i : $\mathbf{z}_i^g$, we wish to find the state error given in Equation (2.15). The predicted state value is computed with the following function:

$$h_{i,s}^g(\mathbf{x}) = \mathcal{K}(\mathcal{C}(\mathcal{T}_{i,s}^g(\mathbf{x}))) \quad (2.24)$$

where $\mathcal{T}_{i,s}^g$ is the function which transforms the landmark g so that it is relative to frame i . $\mathcal{C}$ is a sensor and robot to vision transform which moves the landmark into the appropriate stereo camera frame (left or right) and switches it from our extrinsic robotic frame to the internal computer vision frame. Finally, $\mathcal{K}$ is the camera projection function. The derivative of this function for state $\mathbf{x}$ becomes:

$$\frac{\partial h_{i,s}^g}{\partial \mathbf{x}} = \frac{\partial \mathcal{K}}{\partial \mathcal{C}} \frac{\partial \mathcal{C}}{\partial \mathcal{T}_{i,s}^g} \frac{\partial \mathcal{T}_{i,s}^g}{\partial \mathbf{x}} \quad (2.25)$$

Starting from the right we now compute each individual Jacobian.

Transform Derivative

Remembering that we simply want to refine the 6 DoF pose estimate and all landmarks have been moved to the previous frame $j = i - 1$ (by application of Equation (2.3) with the target frame $i = j$), we compute the following:

$$\frac{\partial \mathcal{T}_{i,j}^g}{\partial \Delta \mathbf{t}} = \mathbf{T}_{i,j} \frac{\partial \Delta \mathbf{T}}{\partial \Delta \mathbf{t}} \quad (2.26)$$

where we describe the error by a delta transform $\Delta \mathbf{T}$ parameterised by a 6 vector $\Delta \mathbf{t} = [\delta x, \delta y, \delta z, \delta \theta_r, \delta \theta_p, \delta \theta_q]^T$ and have set $\mathbf{x} = \Delta \mathbf{t}$. We evaluate the delta transform about zero: $\left. \frac{\partial \Delta \mathbf{T}}{\partial \Delta \mathbf{t}} \right|_{\Delta \mathbf{t}=\mathbf{0}}$. We define our transformation matrices to be created from $\mathbf{t}$ in the following way:

$$\mathbf{T}(\mathbf{t}) = \begin{bmatrix} \mathbf{R}(\mathbf{t}) & \boldsymbol{\rho}(\mathbf{t}) \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (2.27)$$

where

$$\mathbf{R}(\mathbf{t}) = \mathbf{R}_z(\theta_q) \mathbf{R}_y(\theta_p) \mathbf{R}_x(\theta_r), \quad \boldsymbol{\rho}(\mathbf{t}) = [x, y, z]^T \quad (2.28)$$

and $\mathbf{R}_x(\cdot), \mathbf{R}_y(\cdot), \mathbf{R}_z(\cdot)$ are the rotation matrices about the denoted axis. Taking derivatives with respect to each element in $\mathbf{t}$ and evaluating at zero leads to the following generator matrices:

$$\begin{aligned}
 \frac{\partial \mathbf{T}}{\partial x} = \mathbf{G}_1 &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \frac{\partial \mathbf{T}}{\partial y} = \mathbf{G}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \\
 \frac{\partial \mathbf{T}}{\partial z} = \mathbf{G}_3 &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \frac{\partial \mathbf{T}}{\partial \theta_r} = \mathbf{G}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \\
 \frac{\partial \mathbf{T}}{\partial \theta_p} = \mathbf{G}_5 &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \frac{\partial \mathbf{T}}{\partial \theta_q} = \mathbf{G}_6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
 \end{aligned} \tag{2.29}$$

$$\frac{\partial \mathbf{T}}{\partial \mathbf{t}} = [\mathbf{G}_1 \quad \mathbf{G}_2 \quad \mathbf{G}_3 \quad \mathbf{G}_4 \quad \mathbf{G}_5 \quad \mathbf{G}_6] \tag{2.30}$$

where $\frac{\partial \mathbf{T}}{\partial \mathbf{t}}$ is a $4 \times 4 \times 6$ tensor. Right multiplying by a 3×1 vector $\mathbf{v} = [x, y, z]^T$:

$$\begin{aligned}
 \frac{\partial \mathbf{T}}{\partial \mathbf{t}} \mathbf{v} &= [\mathbf{G}_1 \quad \mathbf{G}_2 \quad \mathbf{G}_3 \quad \mathbf{G}_4 \quad \mathbf{G}_5 \quad \mathbf{G}_6] \begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix} \\
 &= \begin{bmatrix} \mathbf{I} & [\mathbf{v}]^\times \\ 0 & 0 \end{bmatrix}
 \end{aligned} \tag{2.31}$$

where

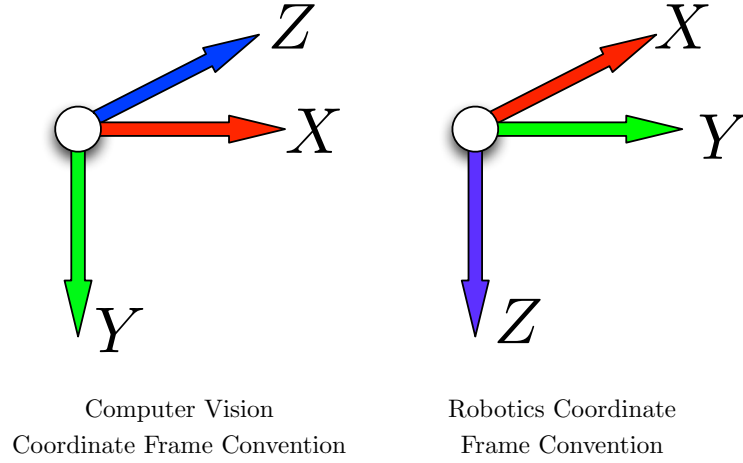


Figure 2.13: In computer vision the typical coordinate frame convention adopted is Z forward, X right and Y down, shown on the left. In robotics the convention often used is X forward, Y right and Z down, shown on the right. Switching from robotics to vision frame conventions can be achieved with the rotation matrix defined in Equation (2.34). Transforming from vision to robotics can be achieved using the inverse.

$$[\mathbf{v}]^\times = \begin{bmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{bmatrix} \quad (2.32)$$

The complete transform derivative for a landmark $\mathbf{p}_j^g$ stored in frame j and observed in frame i is then

$$\frac{\partial \mathcal{T}_{i,j}^g}{\partial \Delta \mathbf{t}} = \mathbf{T}_{i,j} \begin{bmatrix} \mathbf{I} & [\mathbf{p}_j^g]^\times \\ 0 & 0 \end{bmatrix} \quad (2.33)$$

Sensor and Robot to Vision Transform Derivative

The sensor and robot to vision transform function performs two tasks. The first is to move the landmark from the sensor frame i , (the frame in which we are estimating the motion) to each of the camera sensor frames via the transforms $\mathbf{T}^l$ and $\mathbf{T}^r$ (left and right respectively). Using the model depicted in Figure 2.8, with the origin at

the left camera sensor, the transform for the left camera is $\mathbf{I}_{4 \times 4}$. The transform for the right camera is generated from the 6 DoF pose parameter $\mathbf{t} = [0, -b, 0, 0, 0, 0]$, where b is the baseline of the camera for the stereo camera, i.e. the right camera is distance b along the y axis relative to the left camera.

Additionally, the function enables easy switching between two different coordinate frame conventions. Extrinsically we use a robotics (or “aero”) frame of reference which defines X forward, Y right and Z down. To perform the projection step we use the standard computer vision frame: X right, Y down and Z forward. These two frame conventions are shown in Figure 2.13. To switch from robotics to vision frames we define the following rotation matrix $\mathbf{C}_{v,r}$:

$$\mathbf{C}_{v,r} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad (2.34)$$

where v and r denote the robotics and vision frames respectively. The full function, given a landmark in the frame of the sensor $\mathbf{p}_i$ is:

$$\mathcal{C}(\mathbf{p}_i) = \begin{bmatrix} \mathbf{C}_{v,r} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{T}^* \begin{bmatrix} \mathbf{p}_i \\ 1 \end{bmatrix} \quad (2.35)$$

and its derivative is:

$$\frac{\partial \mathcal{C}}{\partial \mathbf{p}_i} = \begin{bmatrix} \mathbf{C}_{v,r} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{T}^* \begin{bmatrix} \mathbf{1} \\ 0 \end{bmatrix} = \mathbf{C}_{v,r} \mathbf{R}^* \quad (2.36)$$

where $\mathbf{T}^*$ is either the left or right transform depending on the observation and $\mathbf{R}^*$ is the rotation matrix from $\mathbf{T}^*$.

Perspective Projection Derivative

Finally, we want to compute the perspective projection derivative. Given a point in the frame of the camera, denoted $\mathbf{p}_c$, we can write the projection equation as follows:

$$\mathcal{K}(\mathbf{p}_c) = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{bmatrix} \begin{bmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \\ 1 \end{bmatrix} \quad (2.37)$$

For convenience we split this into two functions, $\mathcal{K}_1$ and $\mathcal{K}_2$:

$$\mathcal{K}(\mathbf{p}_c) = \mathcal{K}_2(\mathcal{K}_1(\mathbf{p}_c)) \quad (2.38)$$

where

$$\mathcal{K}_1(\mathbf{p}_c) = \begin{bmatrix} x'_c \\ y'_c \end{bmatrix} = \begin{bmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \end{bmatrix} \quad (2.39)$$

and

$$\mathcal{K}_2(\mathbf{p}'_c) = \begin{bmatrix} u_c \\ v_c \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{bmatrix} \begin{bmatrix} x'_c \\ y'_c \\ 1 \end{bmatrix} \quad (2.40)$$

where we have used the dash notation to mark fraction values. Now taking derivatives of each and combining:

$$\frac{\partial \mathcal{K}_1(\mathbf{p}_c)}{\partial \mathbf{p}_c} = \begin{bmatrix} \frac{1}{z_c} & 0 & \frac{-x_c}{z_c} \\ 0 & \frac{1}{z_c} & \frac{-y_c}{z_c} \end{bmatrix} \quad (2.41)$$

$$\frac{\partial \mathcal{K}_2(\mathbf{p}'_c)}{\partial \mathbf{p}'_c} = \begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix} \quad (2.42)$$

$$\frac{\partial \mathcal{K}(\mathbf{p}_c)}{\partial \mathbf{p}_c} = \begin{bmatrix} \frac{f_u}{z_c} & 0 & \frac{-f_u x_c}{z_c} \\ 0 & \frac{f_v}{z_c} & \frac{-f_v y_c}{z_c} \end{bmatrix} \quad (2.43)$$

where Equation (2.43) is our projection derivative.

The Full Jacobian

Combining Equations (2.33), (2.36) and (2.43) leads to the full Jacobian used to refine the pose estimate after the RANSAC step in the front-end.

These equations can be extended to solve a larger state vector containing more poses and possibly landmarks, which is typically done by the back-end solver. For landmark refinement, the derivative of Equation (2.24) is required with respect to the landmarks, it shares similar components to the transform derivative with only the last term changing, instead requiring $\frac{\partial \mathcal{T}_{i,s}^g}{\partial \mathbf{p}^g}$.

In PTAM [51] a full bundle adjustment (all poses and landmarks are refined) is performed over the whole map, but as computational cost is proportional to the number of variables in BA, the system is limited to small workspaces. Konolige *et al.* [53, 1] solve the full BA problem but only in a recent local window, while Strasdat *et al.* [100] extend this to an inner and outer region optimisation, performing full BA in the inner window and the cheaper pose only problem in the outer. Sibley *et al.* [98] solve the problem when the map is in a truly relative setting. This leads to a slightly modified Jacobian compared to previous methods as the observations of a landmark in frame i which is stored in frame s affects all the relative transforms which join s to i .

It is interesting to note that while these systems do improve the estimate, the

delta is dependant on the motion of the camera through the scene. Konolige *et al.* [1] report good improvement when including bundle adjustment for a robot moving at modest pace through the world. The Kitti Vision Benchmark Suite [38] - discussed in more detail in Chapter 3 - is benchmarking dataset for VO algorithms, with the data coming from a moving vehicle in an urban environment. The top performing methods do not use BA, and two methods have been submitted with and without BA enabled, both of which perform slightly better without BA. One reason for this is due to the very short landmark lifetime caused by the speed of the vehicle, i.e. features are only seen for a few frames. This means the benefits of multi-frame BA cannot be leveraged to improve the already good VO estimate. This, combined with our assumption of a relative frame means these solvers, while nice to have, are not essential. Their utility is perhaps best felt at loop closure events, where they can join up and smooth the two ends of a trajectory and make the motion between them appear locally accurate.

2.7 Conclusions

In this chapter we have provided a high level overview of the various components required for a complete VO system, as well as some background on the subject. Further details and system performance are described in Chapter 4. The VO system also forms the key building block of the long-term navigation framework presented in Chapter 5.

Chapter 3

Systems and Datasets

3.1 Introduction

In this chapter we outline some of the robotic platforms and datasets used throughout this thesis. Taking time to create helpful and robust software has aided algorithm understanding and made the transition from development to deployment easier. Further, having a standard set of datasets with ground truth has allowed for thorough testing of the system. Most of the datasets introduced below are used extensively in the following chapters.

3.2 Datasets with Ground Truth

In Chapter 4 we wish to analyse the performance of a stereo visual odometry system. To do this objectively we need to know the true motion of the camera, referred to as ground truth. If this is known, changes to the system can be applied and their effects on performance analysed quantitatively. When operating outdoors, one of the best available sources of ground truth data is an Inertial Navigation System (INS). Good INS units typically provide reliable pose estimates, but can still

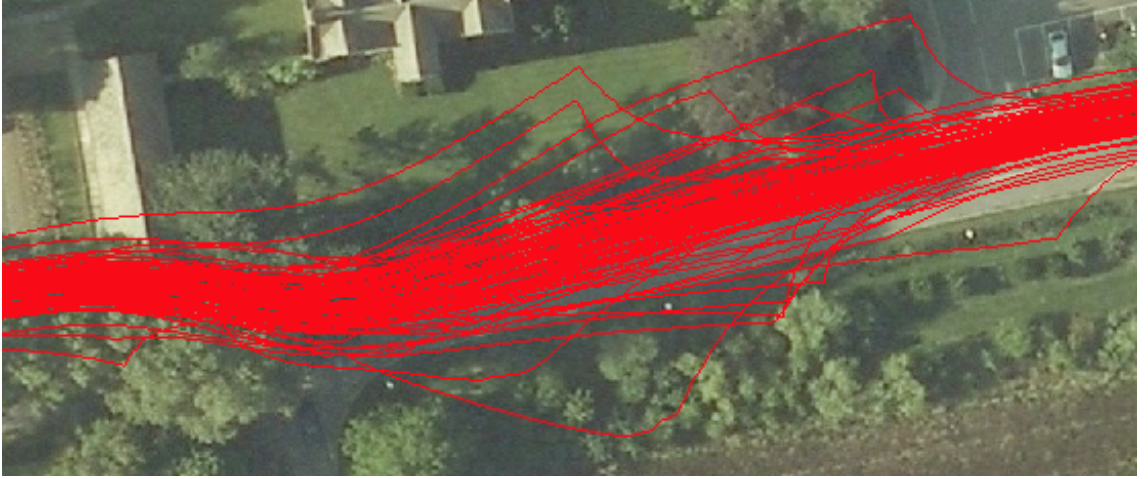


Figure 3.1: A plot of the position estimate of our vehicle equipped with a high end INS driving the same route over a three month period. Passing under trees and near tall buildings interfere with the GPS signal and the IMU quickly diverges from the true position.

suffer when the Global Position System (GPS) signal is attenuated by buildings and overhanging trees - Figure 3.1. Conversely, synthetic images can be created from textured 3D models using computer graphics. Sequences created in this manner are useful because the exact camera motion is known, which is helpful for testing. However it is difficult to model many of the effects seen in real world data, such as changing lighting, dynamic objects and motion blur, so it is unwise to test exclusively on synthetic data. To balance the short-comings of real and synthetic data with respect to ground truth, we test on four different datasets containing sequences from both, this also prevents tuning the VO system to a specific dataset. The first dataset is generated from a synthetic world, with the remaining three from real robotic platforms; we refer to this set of four as the VO Test Datasets.

3.2.1 Synthetic Dataset

To create synthetic visual data a program was written using OpenGL that can render a model from different positions and store the result. It reads in textured OBJ files¹

¹The OBJ file format is a widely used way of representing simple 3D models in plain text.

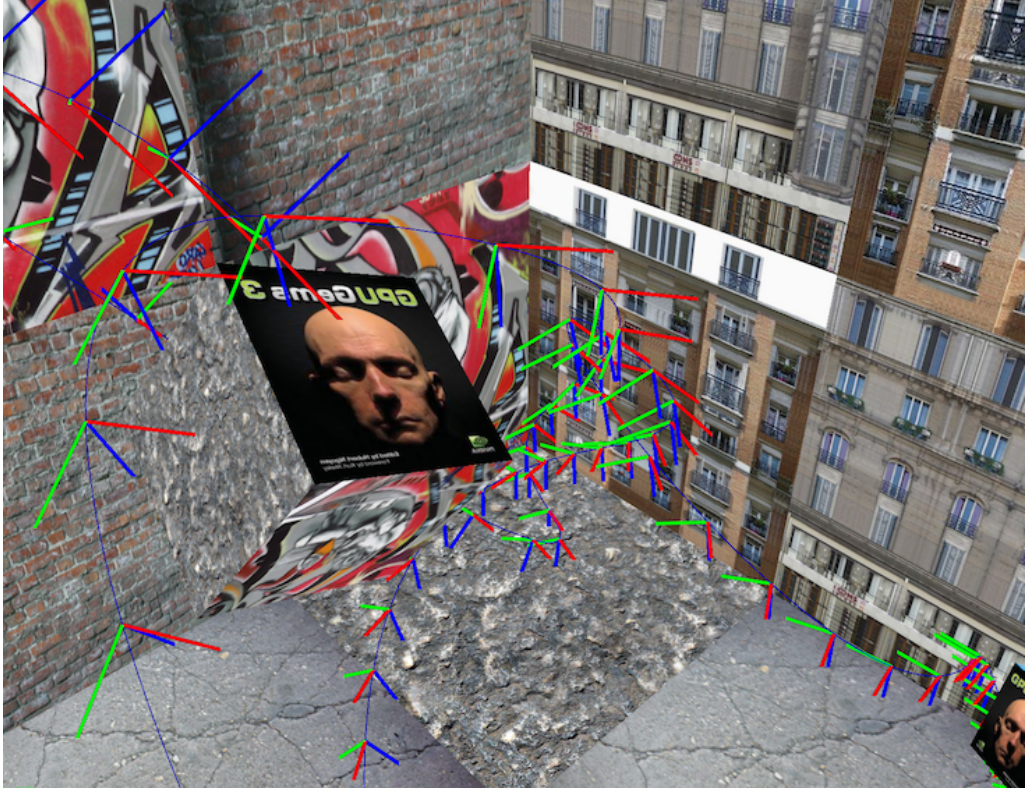


Figure 3.2: A screen shot from the artificial world used to create the “Synthetic” dataset. The camera trajectory is shown in blue, and poses shown by the red, blue and green axis. The model used is a simple block world with rich texturing.

and allows the user to specify a sequence of poses they wish the artificial camera to follow. Once this has been provided, the model is rendered from each pose and a call is made to `glReadPixels` to capture the screen buffer. The scene is rendered at four times the desired resolution to reduce aliasing effects.

The model used in this thesis is a simple block world with rich textures on each face. The dataset aims to simulate difficult and perhaps slightly unusual motion such as pure rotation and rotation about the optical axis. A screen shot from the model along with the camera trajectory can be seen in Figure 3.2. We refer to this dataset as “Synthetic”, and details can be found in Table 3.1.

Table 3.1: VO Test Datasets - Summary

Name:	Synthetic	Kitti06	Begbroke	Kitti00
Camera	OpenGL	Flea2	Bumblebee2	Flea2
# Frames	443	1,104	10,110	4,544
Length	70.0m	1.23km	2.07km	3.72km
Ground Truth	Computer	RT3003	RT3042	RT3003
Size	502Mb	139Mb	3.67Gb	550Mb
Image Size	512x384	620x188	512x384	620x188
Capture Rate	N/A	10Hz	20Hz	10Hz



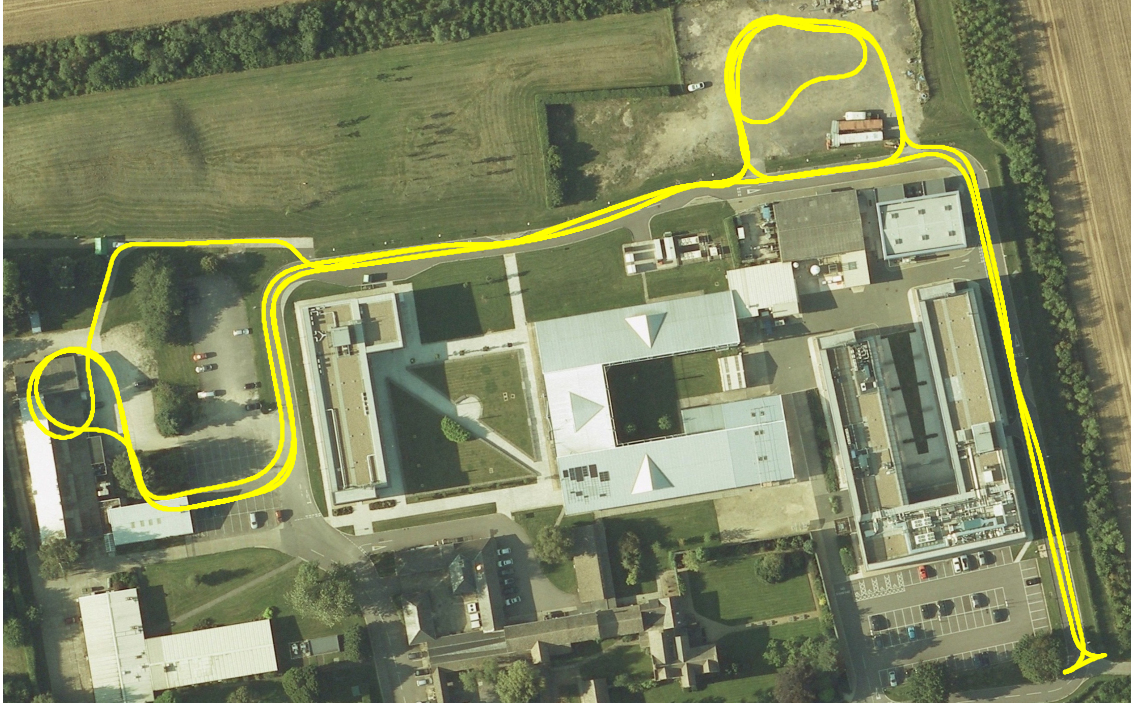
Figure 3.3: The platforms used to collect our real world VO Test datasets. (a) is the Mobile Robotics Group Wildcat vehicle, which is based on the Bowler Wildcat. (b) is the VW Passat used to collect the Kitti sequences. Image from [38].

3.2.2 Real Datasets

We also test on three datasets collected from two different robotic platforms. The first was collected from the Mobile Robotics Groups Wildcat vehicle, shown in Figure 3.3a. The route was just over 2km long and was collected on an overcast afternoon with a Point Grey Research Bumblebee2 camera. Ground truth is provided by a dual-antenna OXTS RT3042 INS, with corrections via radio signal from a local base station. The measurements from this unit are accurate in position to 0.1m, heading to 0.1 degrees and pitch and roll to 0.03 degrees [83]. Care was taken

to avoid overhanging trees which can interfere with the INS estimate. We refer to this dataset as “Begbroke” and further details can be found in Table 3.1. The route and example imagery can be seen in Figure 3.4.

The second and third real world sequences come from the Kitti Vision Benchmark Suite [38]. We use sequence 00 and 06 from the Visual Odometry datasets, which we refer to as “Kitti00” and “Kitti06”. Kitti06 is a 1.2km loop, shown in Figure 3.5 while Kitti00 is a longer 3.7km route from a residential area, shown in Figure 3.6. The data was collected from a modified VW Passat, Figure 3.3b, in Karlsruhe, Germany using two Point Grey Research Flea2 cameras with varifocal lenses. Ground truth is provided by an OXTS RT 3003 INS. The measurements from this unit are accurate in position to 0.02m, heading to 0.1 degrees and pitch and roll to 0.03 degrees [83]. More details can be found in Table 3.1.



(a)



(b)



(c)

Figure 3.4: (a) is the trajectory driven for the Begbroke dataset. The route starts and ends in the North East car park. Ground truth was provided by an INS. Care was taken to avoid overhanging trees and differential corrections were provided by radio link from a base station mounted on the roof of one of the central buildings. (b) and (c) are example images from the dataset.



(a)



(b)

Figure 3.5: (a) shows the route driven for the Kitti visual odometry sequence 06, and (b) is the starting image. Ground truth was provided by an INS with differential corrections. Overhead imagery courtesy of Google.



(a)

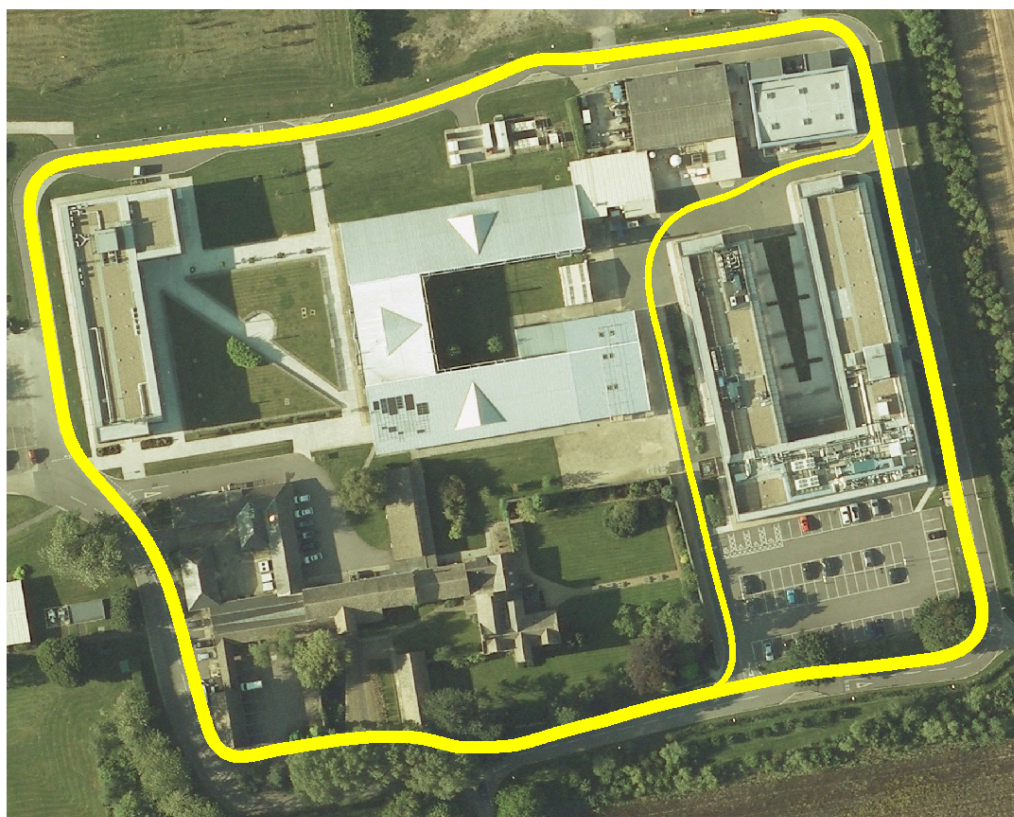


(b)

Figure 3.6: (a) shows the route driven for the Kitti visual odometry sequence 00, and (b) is the starting image. Ground truth was provided by an INS with differential corrections. Overhead imagery courtesy of Google.

3.3 Datasets with Multiple Loops

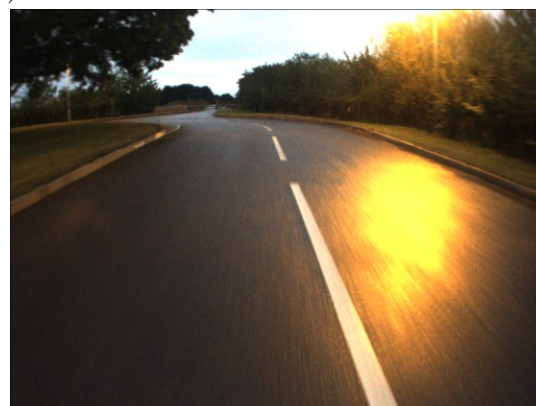
In Chapter 5 we develop a lifelong navigation system, with the VO system playing a key role. To help analyse the performance of our approach we captured a dataset which contains repeated loops of the same environment over a long period of time. The Begbroke Science Park where the Mobile Robotics Group has its field laboratory was chosen as the test site. Over a three month period during the summer of 2011, 53 loops were driven with the Group’s Wildcat vehicle, totalling 37km. Efforts were made to go out at different times of day and in different weather conditions. Dataset collection times ranged from early morning to late evening and experienced the full range of British summer weather - bright sunshine, overcast conditions and rain. The two routes driven are depicted in Figure 3.7. The majority of the loops were driven on the thicker outer circuit, with a few driving the thinner section through the middle.



(a)



(b)



(c)

Figure 3.7: Repeated data collection at the Begbroke Science Park for long-term navigation testing. 53 traverses were collected over a three month period in different weather conditions and at different times of day. (a) shows the two routes driven, (b) and (c) are two examples of the conditions from two different outings.

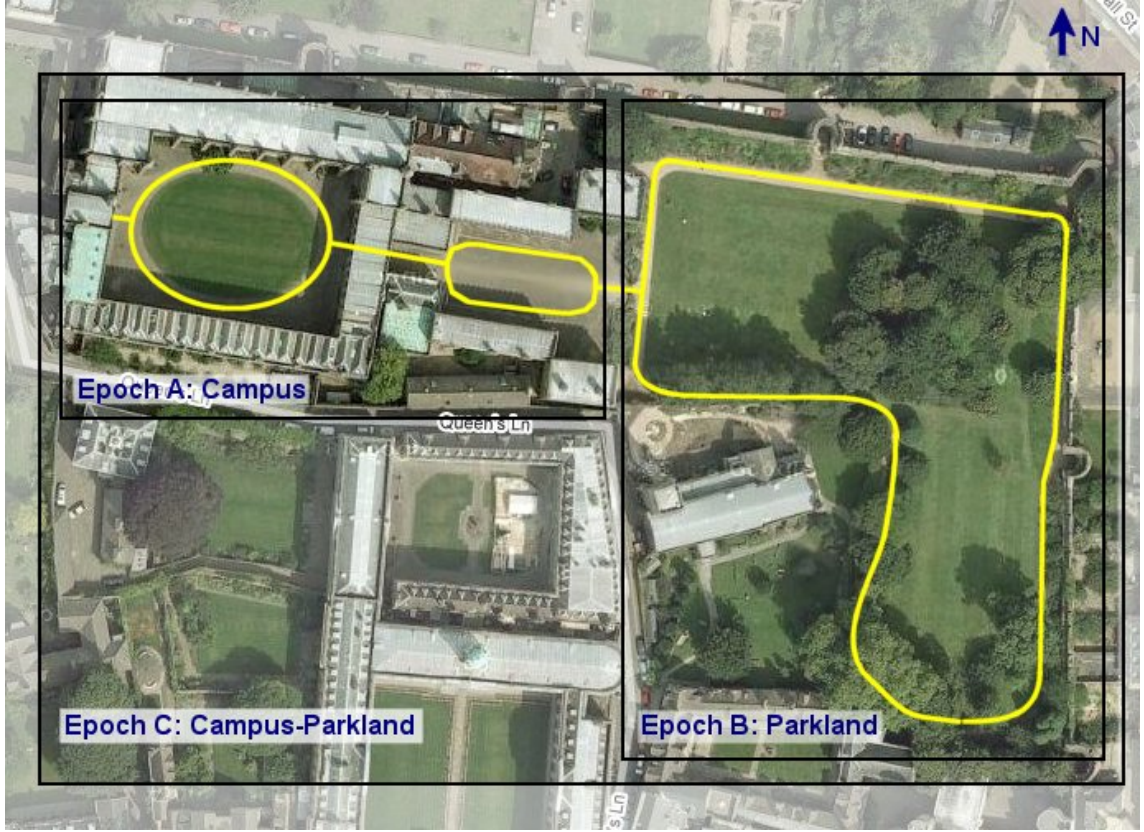


Figure 3.8: Overhead view of the route taken for the New College Dataset. Epochs are used to describe different parts of the dataset.

3.4 Published Datasets

During the winter of 2008-2009, several datasets were collected around the grounds of New College, Oxford using the groups Segway based robot “Lisa”. Lisa is equipped with two SICK LMS291 lasers, a Bumblebee2 stereo camera, a Ladybug2 omnidirectional camera, a GPS unit and an Inertial Measurement Unit (IMU). One of the datasets collected was published as The New College Dataset [99] - a peer reviewed data paper - and an accompanying website was created². Since its release, The New College Dataset has been widely used by the robotics community [111, 56, 101, 63, 77]. Unfortunately due to the lack of a large number of repeated loops or high quality ground truth we do not use it for testing or analysis.

²<http://www.robots.ox.ac.uk/NewCollegeData/>

3.5 Implementation Details

The stereo visual odometry system and lifelong navigation approach described in this thesis were written in C++. The code is cross-platform, working on OS X, Linux and Windows. This was made significantly easier by using the following libraries:

- **Eigen:** A fast C++ vector, matrix and linear algebra library. eigen.tuxfamily.org
- **Boost:** A set of high quality peer-reviewed libraries that extend the core functionality of C++. www.boost.org
- **OpenGL:** A cross-language, cross-platform API for rendering 2D and 3D graphics. www.opengl.org
- **FLTK:** Standing for “Fast Light Toolkit”, FLTK is a cross-platform C++ Graphical User Interface (GUI) library. www.fltk.org
- **MOOS:** Standing for “Mission Oriented Operating System”, MOOS is a cross-platform robotics middleware software suite. It is used to communicate with other processes running on the robot. www.robots.ox.ac.uk/~mobile/MOOS/wiki
- **CMake:** A cross-platform build system, useful for generating and maintaining make and project files across different operating systems. www.cmake.org

A screenshot of the application used to develop the VO systems is shown in Figure 3.9. Powerful visualisation tools make development and debugging significantly easier.

3.5.1 Acquiring, Managing and Consuming Data

To achieve the long-term navigation discussed in Chapter 5 we need the ability to save the output of the exploration system, in this case the VO. For this task we use a combination of Google Protocol Buffers³ and SQLite⁴.

Google Protocol Buffers is a cross-platform, cross-language serialisation and de-serialisation library. The user writes `.proto` file containing the items they wish to store. Google Protocol Buffers then generates (in the case of C++) the corresponding class files which can read and write the class data via `std::string`. SQLite is, as the name suggests, a lightweight Structured Query Language (SQL) library. Each node and transform in the map is stored in corresponding `node` and `transforms` tables in the database. Each entry is simply a unique id and the string blob from the Google Protocol Buffers. The combination of these two tools allows for cross-platform saving and loading of the VO output.

3.6 Conclusions

In this chapter we have introduced several datasets that we will use in later chapters for testing. We have also given a brief insight into the engineering effort needed to create and manage the applications and large amounts of data used in this thesis.

³<http://code.google.com/p/protobuf/>

⁴<http://www.sqlite.org>

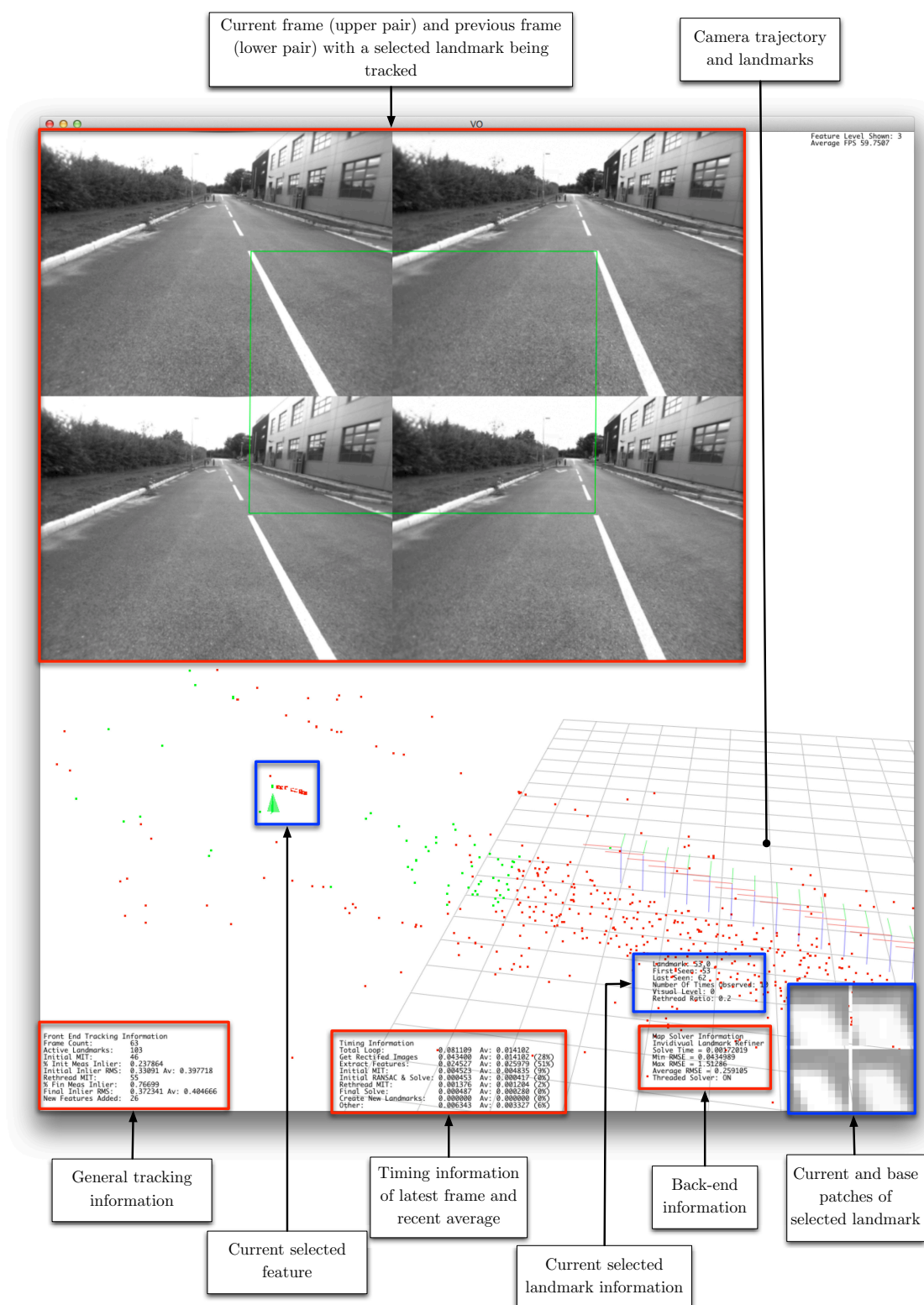


Figure 3.9: A screenshot of the VO GUI. General items are marked with a red box. Information related to a specific feature and associated landmark picked by the user is marked in blue. The GUI provides timing and estimation information as well as specific information about individual features and landmarks.

Chapter 4

Stereo Visual Odometry Analysis

4.1 Introduction

This chapter aims to provide a more detailed analysis of our stereo visual odometry system. We will discuss some of the key factors in more detail and provide extensive results from the VO Test Datasets presented in Chapter 3. We begin by describing the error metrics used and layout a default, vanilla setup for which we present performance results. We then change different aspects of the algorithm and system parameters, such as feature count, type and distribution, and analyse their effects on performance. Finally we conclude with a few remarks about long-term performance and report on the system's use by third parties.

4.2 Error Metrics

In this chapter we will be quantitatively evaluating different system options. We are interested in the motion error (pose of the camera), rather than the structure error (position of the landmarks), as the motion of the camera is what is used in most robotics applications. It is also more difficult to acquire accurate ground truth

of the structure of real world sequences. We report the error of the relative poses rather than the global position error as we are assuming a relative frame of reference. We also present cumulative error plots rather than the single error at the end of a trajectory sequence as relative pose errors can cancel each other out over the course of a sequence.

As we are using relative transforms, the components of $\mathbf{t}$ are typically very small apart from along the axis of translation. Reporting the errors for each axis is uninformative at best (errors in rotation are typically $< 10^{-3}$) so generally we will report a combined metric. The group which describes rigid body transformations in three dimensional space - the special Euclidean group $SE(3)$ - does not have a single metric, so we split it into two parts. Firstly we report the translational error ρ_e , from the predicted $\boldsymbol{\rho}_p$ and ground truth value $\boldsymbol{\rho}_{gt}$, using the following equation:

$$\rho_e = \|\boldsymbol{\rho}_p - \boldsymbol{\rho}_{gt}\|_2 \quad (4.1)$$

where $\|\cdot\|_2$ is the l^2 norm.

For the rotations we report the error in angle, from the axis angle formulation. Taking a rotation matrix $\mathbf{R}$ we compute the angle via:

$$\phi = \arccos\left(\frac{\text{tr}(\mathbf{R}) - 1}{2}\right) \quad (4.2)$$

Finally we also report percentage errors. For some predicted value v_p and a corresponding ground truth value v_{gt} , we compute the percentage error Ψ_e using the following formula:

$$\Psi_e = 100 * \frac{|v_p - v_{gt}|}{|v_{gt}| + \sigma_r} \quad (4.3)$$

where $|\cdot|$ is the absolute value and σ_r is a robust factor to prevent dividing by zero when $|v_{gt}| \rightarrow 0$ (for example when the camera is stationary). This stops the

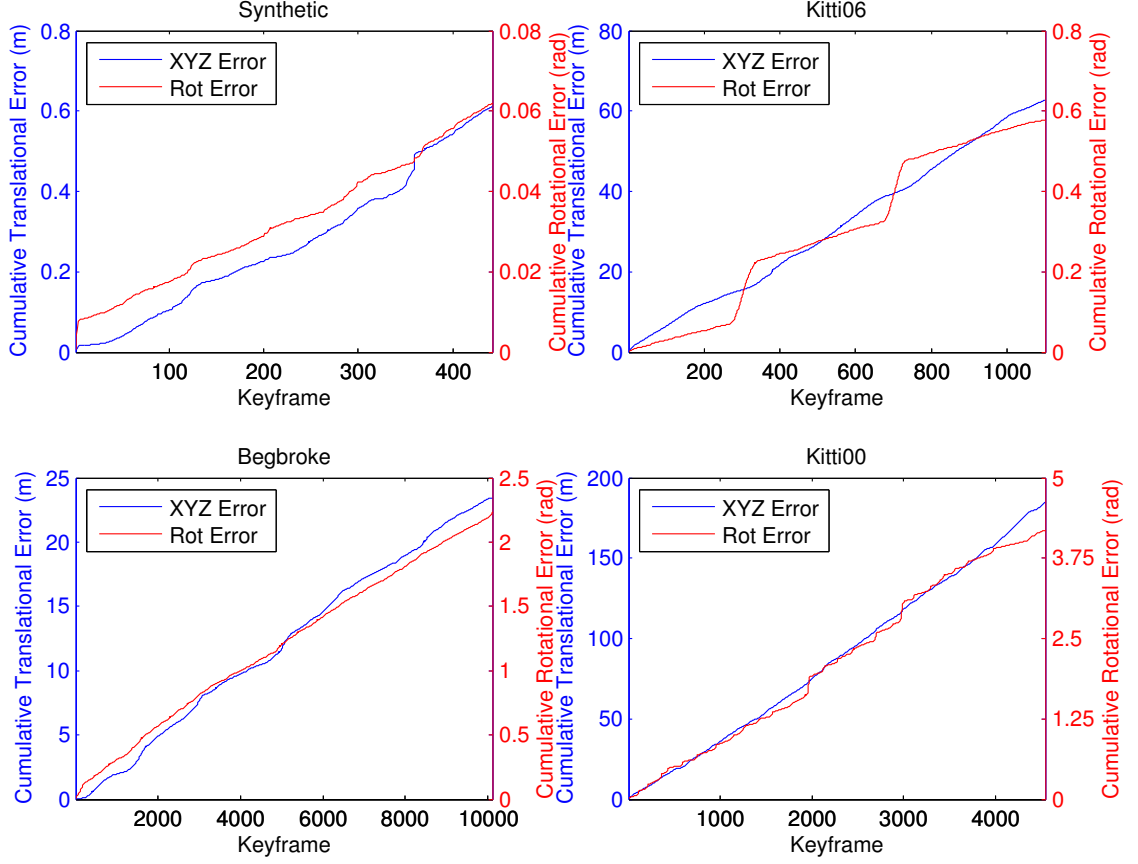


Figure 4.1: Cumulative translational and rotational error plots for the vanilla VO system running on the VO Test Datasets.

result becoming Inf . We set $\sigma_r = 0.001$ for our experiments. We note that this metric is bias for different ground truth, i.e. the percentage error will be larger when the ground truth value is smaller for the same error. However we will not look to compare the system across datasets (which will have different ground truth values), but only change the system and compare the differences against the same ground truth. This means all comparisons will have the same biasing.

4.3 Default Setup

We now describe our default VO setup, which we refer to as our “vanilla” settings. This actually includes all of the features that make the system work well, even

though we refer to it as the vanilla setup. Here we are just using the front-end system without any back-end solver or loop closure signals. We aim to track 125 features per frame, which are FAST extracted corners, matched with BRIEF, and refined with ESM using 9×9 pixel patches centred on the feature location. FAST extracts corners with scores above a certain threshold which are then ranked using the Harris corner score [41]. We force features to be distributed evenly across the image, the details of which are discussed in Section 4.4.2. We also perform an operation called “rethreading”, which aims to increase the likelihood of matching all of the available features. This is described in Section 4.4.3.

Pose Estimation Performance

We begin by discussing the results of our vanilla VO system operating on the VO Test Datasets. We report mean, median, standard deviation, minimum and maximum errors for various metrics of the system on each dataset. These can be found in Appendix A in Tables A.1, A.2, A.3 and A.4. We also show the cumulative error plots for ρ_e and ϕ_e in Figure 4.1.

The mean translational errors per axis are of the order of *cm* in the worst case. The worst performing is Kitti06 with a ρ_e of 0.057m, however we note that the percentage error is still $\sim 5\%$, in line with the other datasets. Kitti06 has two long straight sections (see Figure 3.5) where the vehicle travels relatively fast, and capturing at only 10 Hz, results in large displacements between frames. Conversely Begbroke has a higher sampling rate - images are captured at 20 Hz - and we see a reduced error, and percentage error per relative transform. The occasional very large percentage translation error, for example in Synthetic, is when the camera is stationary, but the estimate has returned a small motion, resulting a large *percentage* error. This also skews the standard deviation. These occasional large percentage errors also bias the means, so we also report the medians, which are always lower.

This problem also affects the rotations which we discuss below. We see in Figure 4.1 the error in translation is linearly increasing and is not overly affected by the specific motion of the camera.

The mean rotational errors about each axis are less than 10^{-3} , with the Kitti datasets having slightly worse performance. Again this is probably being caused by the lower sampling rate, and we note the percentage error of ϕ_e in the Kitti datasets to be approximately equal, and again lower for Begbroke. From the cumulative plots in Figure 4.1 we see rotational error is more influenced by the motion of the camera. This is perhaps most noticeable in the Synthetic and Kitti06 sequences. The rotational error in Synthetic jumps in the first few frames, this correlates to the pure rotation the camera experiences at the beginning of the sequence. Equally we see two large steps in the error plots for Kitti06, these match to the points when the vehicle performs a 180° turn at each end of the loop. Smaller steps can be seen in Kitti00, where the vehicle performs 180° turns. The motion of the camera is slightly smoother in Begbroke, combined with a higher sampling rate, produces a smoother error curve.

Timing

In Figure 4.2 we give a break down of the timing requirements for each section of the VO, recorded when processing the Begbroke dataset. The total average loop time is 42.6ms. Get Image and Rectify refers to (in these settings) reading the image from disk and performing any rectification needed. Rectification takes approximately 5ms with precomputed look-up tables for the camera. The remaining time is the cost of loading the image from disk. We find this can be quite variable, especially if the operating system has tried to cache the images due to previous recent use, so we omit this timing block from all subsequently reported timings to avoid any biasing. FAST extraction includes the cost of scoring and ranking the features. We

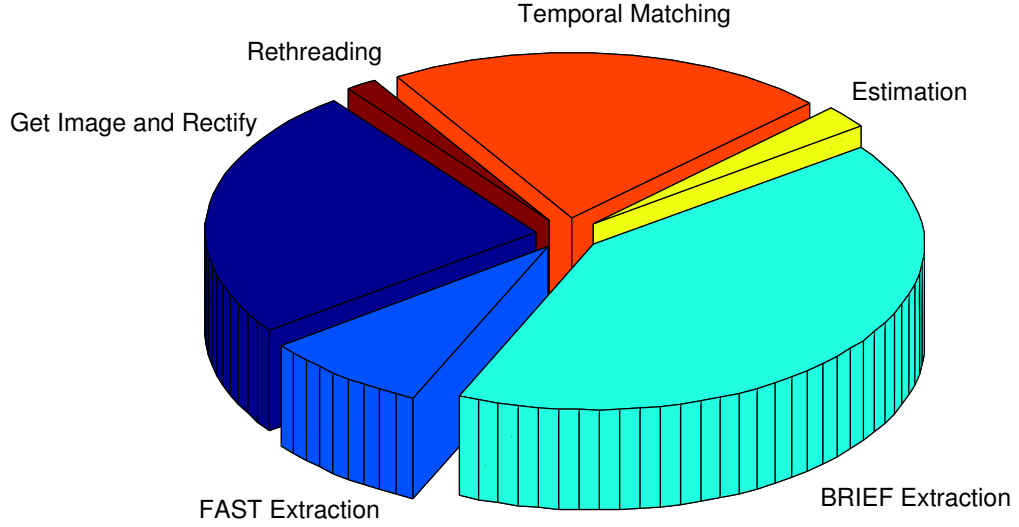


Figure 4.2: Timings for the vanilla VO settings - total time taken is 42.6ms per frame. The bulk of the timing is taken by the BRIEF extraction step. These timing results are for a dual thread application (left and right images processed in parallel). Loading an image from disk produces variable timings due to operating system caching, so we omit this from future timing results. The image rectification step takes 5ms.

note the bulk of the time is used executing the BRIEF Computation step. This refers to the extraction of BRIEF features for the FAST corners which survived a local non-maximal suppression, which is typically in the order of 4,000-5,000 corners for a 512×384 stereo pair. Note we use a low initial threshold as this generate as many corners as possible, even in regions that are relatively featureless, before we apply the non-maximal suppression to reduce the number of local repetitions. These timings are produced from an implementation which takes advantage of multi-core machines to run the FAST and BRIEF components in parallel - one core for each image - reducing the cost by a factor of two. If only one core is available these steps dominate the total cost further. Temporal matching refers to the initial matching step using BRIEF, searching across the whole image, and we see this also takes a significant proportion of the time. The second matching loop, rethreading, is significantly cheaper as we are only searching for a reduced number of features in

a very small window. Finally the estimation slice refers to the RANSAC step and both least squares iterations (before and after rethreading), and is relatively cheap.

Further Details

Table 4.1 includes some other information for the vanilla system on the VO Test Datasets. We include the average time to compute each frame (not including the Get Image and Rectify step). The Kitti datasets are the cheapest, because their images are smaller than Begbroke and Synthetic. Synthetic is the most expensive as it is heavily textured, producing lots of FAST corner responses (and corresponding BRIEF descriptors). Most Begbroke images observed some amount of sky which does not produce many responses and results in an image which is computationally cheaper to process. We also include the average reprojection error per landmark for each sequence, all of which are under 0.4 pixels. Synthetic is the best as the rich texturing produces highly repeatable and accurate feature positions. The Kitti sequences are the worst, likely being caused by the long range nature of the features and lower sampling rate. Finally for interest, we integrate all our relative poses to get a global position at the end of the sequence, and compute the error to the final global ground truth pose. We refer to this as the “Error At Finish”. This is not a particularly useful metric, as re-running the system with a different random seed (for RANSAC) would produce a different set of results.

Table 4.1: Further performance metrics for vanilla VO on the VO Test datasets. “px” is the unit of pixels.

	Synthetic	Kitti06	Begbroke	Kitti00
Av. Time Per Frame (s)	0.0580	0.0234	0.0362	0.0294
Av. Reprojection Error (px)	0.1903	0.3681	0.2261	0.2820
Error At Finish: x (m)	-0.0028	-0.9073	-67.1160	-27.0435
Error At Finish: y (m)	-0.0745	0.7676	-8.0666	-58.1990
Error At Finish: z (m)	0.0691	-0.1126	5.0725	20.0861
Error At Finish: θ_r (rad)	-0.0276	-0.0120	-0.0611	-0.0187
Error At Finish: θ_p (rad)	-0.0031	0.0078	0.0230	0.0983
Error At Finish: θ_q (rad)	0.0289	-0.0269	-0.5135	0.5999

4.4 Front-End Analysis

The purpose of the front-end is to provide high quality data associations between the latest stereo frame and the local landmarks and then compute an estimate of the motion of the latest frame. In this section we vary different aspects of the front-end and discuss their impact on the performance. Throughout this section we will use the term rethread ratio. This is defined in Section 4.4.3 but in short is the number of landmarks that need to be matched a second time due to a poor initial data association.

4.4.1 Number of Features

One of the obvious questions when using sparse point feature based systems is how many features to use, or how many landmarks to track per frame. Too few and quality of the estimation degrades, too many and computation time increases with no corresponding increase in estimation quality. FAST typically produces far too many features from an image to be useful for VO, so a way of selecting the top ones is required. We use the Harris corner score [41] to rank the returned features. We then select the top F features (subject to any distribution requirements - discussed in Section 4.4.2) to be used for tracking in the next frame.

We ran the system with F values of 20, 30, 50, 75, 100, 125, 150, 200 and 300. Cumulative translational error plots are shown in Figure 4.3 and 4.4, average computation time per frame is plotted in Figure 4.5 and Table 4.2 gives more detailed performance statistics. We found that tracking less than 20 features significantly impacts performance and regularly results in catastrophic failure. Tracking 20-50 features per frame causes some performance degradation and occasional instability (for example 30 features on Kitti06), but are computationally the cheapest. Once we are tracking at least 75 features the performance is more reliable, and larger values

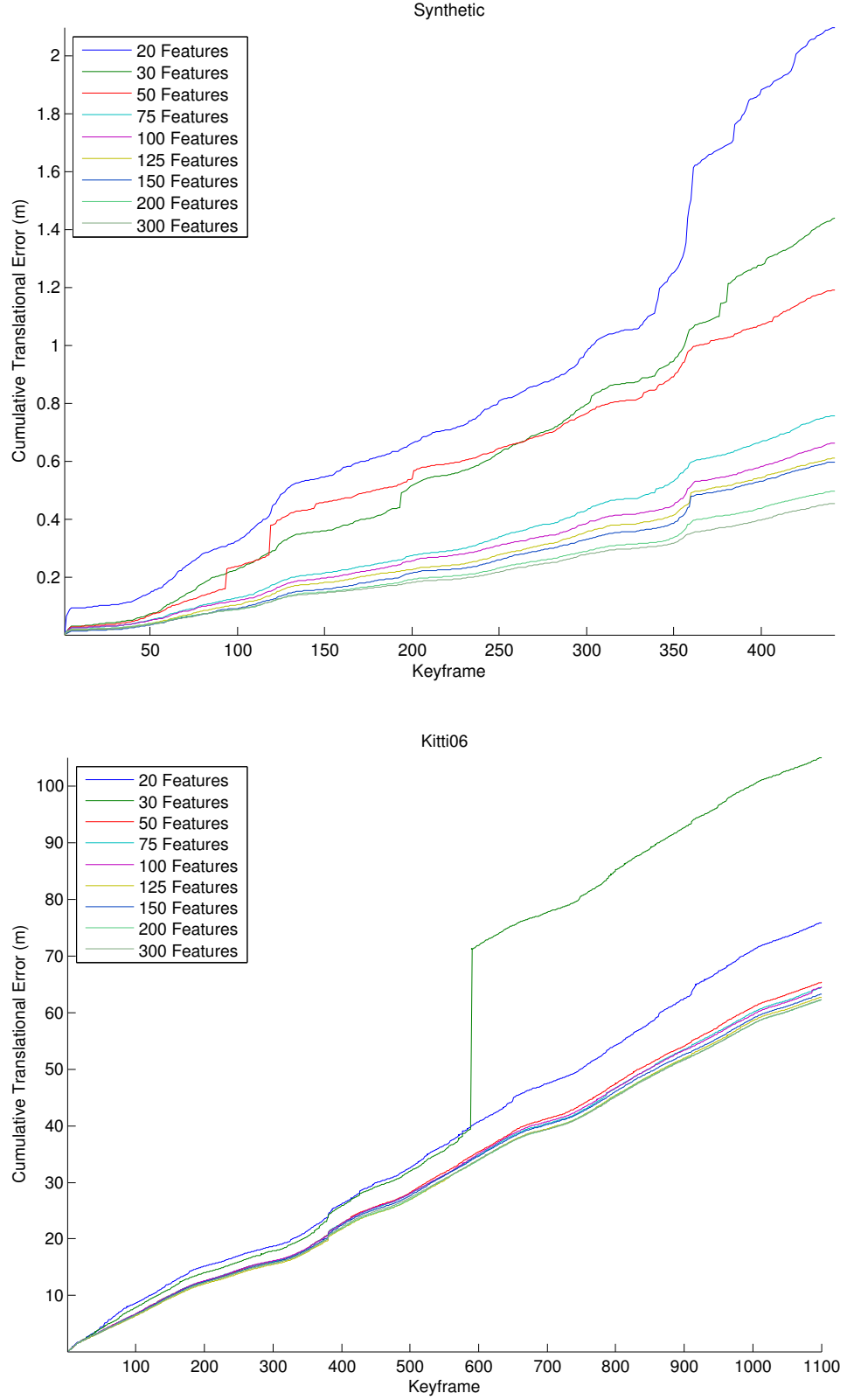


Figure 4.3: Cumulative translational error plots for the Synthetic (top) and Kitti06 (bottom) datasets with varying numbers of features tracked from frame to frame.

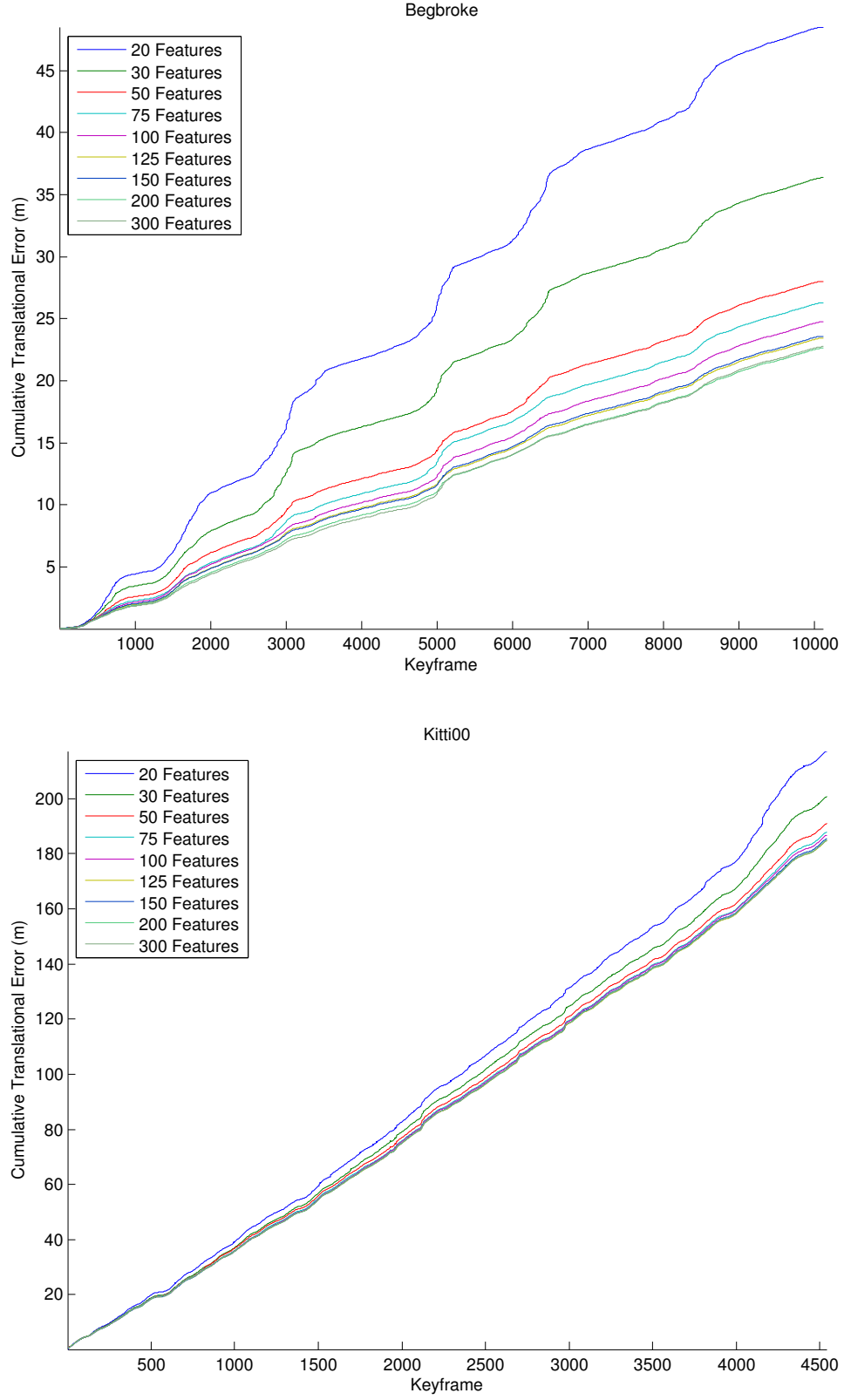


Figure 4.4: Cumulative translational error plots for the Begbroke (top) and Kitti00 (bottom) datasets with varying numbers of features tracked from frame to frame.

Table 4.2: Results for varying numbers of features, F , on the VO Test Datasets. T/F refers to average timings per frame and rr is the rethread ratio.

Synthetic									
F	20	30	50	75	100	125	150	200	300
ρ_e	0.005	0.003	0.003	0.002	0.001	0.001	0.001	0.001	0.001
ϕ_e	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
$\Psi_e(\rho_e)$	23.864	9.125	8.354	6.557	6.573	4.612	4.061	5.140	4.943
$\Psi_e(\phi_e)$	2.477	2.119	2.855	1.269	1.104	1.020	0.975	0.812	0.762
T/F	0.041	0.042	0.046	0.049	0.054	0.056	0.060	0.066	0.087
rr	0.127	0.145	0.157	0.158	0.166	0.174	0.174	0.179	0.194
Kitti06									
F	20	30	50	75	100	125	150	200	300
ρ_e	0.069	0.096	0.059	0.059	0.059	0.057	0.058	0.057	0.057
ϕ_e	0.001	0.003	0.001	0.001	0.001	0.001	0.001	0.001	0.001
$\Psi_e(\rho_e)$	6.229	7.925	5.342	5.265	5.289	5.120	5.174	5.098	5.090
$\Psi_e(\phi_e)$	16.187	102.277	9.724	9.534	8.788	8.518	8.349	8.221	7.911
T/F	0.014	0.014	0.016	0.019	0.020	0.023	0.025	0.027	0.034
rr	0.228	0.243	0.312	0.310	0.316	0.366	0.362	0.366	0.385
Begbroke									
F	20	30	50	75	100	125	150	200	300
ρ_e	0.005	0.004	0.003	0.003	0.002	0.002	0.002	0.002	0.002
ϕ_e	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
$\Psi_e(\rho_e)$	3.358	2.794	2.362	2.205	2.126	2.037	2.041	2.026	1.996
$\Psi_e(\phi_e)$	8.538	8.093	7.560	7.323	7.229	7.120	7.036	7.012	6.963
T/F	0.028	0.027	0.029	0.032	0.034	0.035	0.038	0.041	0.052
rr	0.162	0.204	0.223	0.236	0.249	0.244	0.258	0.267	0.281
Kitti00									
F	20	30	50	75	100	125	150	200	300
ρ_e	0.048	0.044	0.042	0.041	0.041	0.041	0.041	0.041	0.041
ϕ_e	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
$\Psi_e(\rho_e)$	6.138	5.699	5.410	5.340	5.305	5.259	5.294	5.271	5.264
$\Psi_e(\phi_e)$	12.398	10.305	9.174	8.561	8.431	8.303	8.202	8.123	8.060
T/F	0.019	0.019	0.022	0.024	0.025	0.029	0.031	0.033	0.043
rr	0.207	0.222	0.270	0.271	0.274	0.317	0.311	0.311	0.332

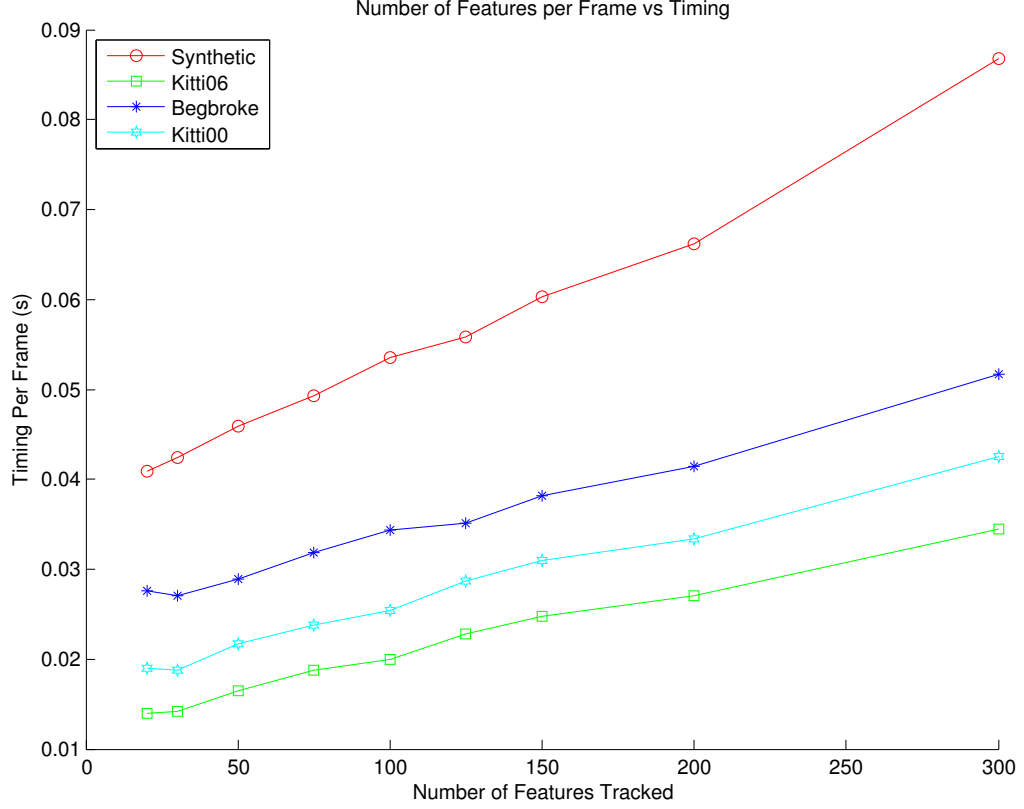


Figure 4.5: Timing performance of the system with different numbers of tracked features on the VO Test Datasets. Increasing the number of features tracked is more computationally expensive and brings diminishing returns in terms of estimation performance.

of F brings increasingly smaller benefits in estimation performance while increasing the computation time. For example stepping from 200 to 300 features tracked per frame results in only $\sim 1\%$ increase in estimation quality for a $\sim 30\%$ increase in computation time. For this reason we have picked $F = 125$ as our default value, a compromise between timing and estimation performance.

4.4.2 Feature Distribution

Once the number of features per frame has been decided, how they are distributed across the image is also important. Simply selecting the top scoring features described in Section 4.4.1 from the *whole* image produces feature distributions which

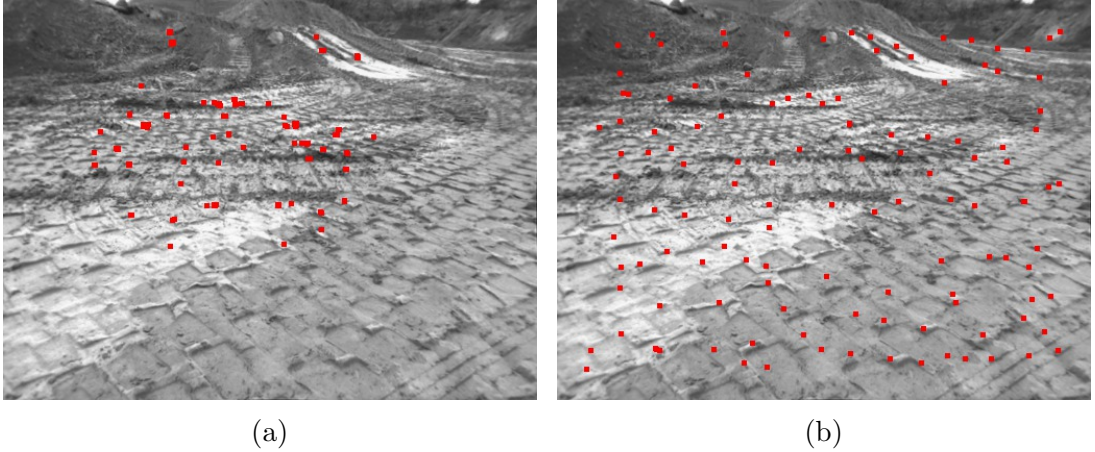


Figure 4.6: The same image with and without an enforced feature distribution (features shown in red). (a) has just selected the top F features from the whole image. Note how features cluster around the strongest textures. This leads to a poorly constrained system and low quality motion estimates. (b) uses a Quad Tree to ensure an even distribution of features, resulting in better motion estimates. Original images courtesy of SCISYS.

cluster around the image regions that give the strongest corner responses. This leads to poorly constrained motion estimates and hurts the solution. Mei *et al.* [71] showed that by using a Quad Tree [30], a more favourable distribution can be enforced which gives better motion estimates.

A Quad Tree is a data structure that can be used to partition 2-D spaces; it is a tree where each node has four child nodes. Placing a Quad Tree over the image enables the image space to be recursively divided up into successively finer regions, and the desired number of features at each region can be specified. We opt for an even distribution across the image, effectively implying no prior knowledge about the scene. If some information about the scene structure was known, it could be incorporated, for example masking the sky. By distributing our F features across the whole image, as determined by the Quad Tree bins (i.e. we only add a feature at a specific location if the Quad Tree’s bin at that location is not full), we achieve an even spread of features that improves the motion estimate. This prevents our F features becoming clustered around the strongest features and producing poorly

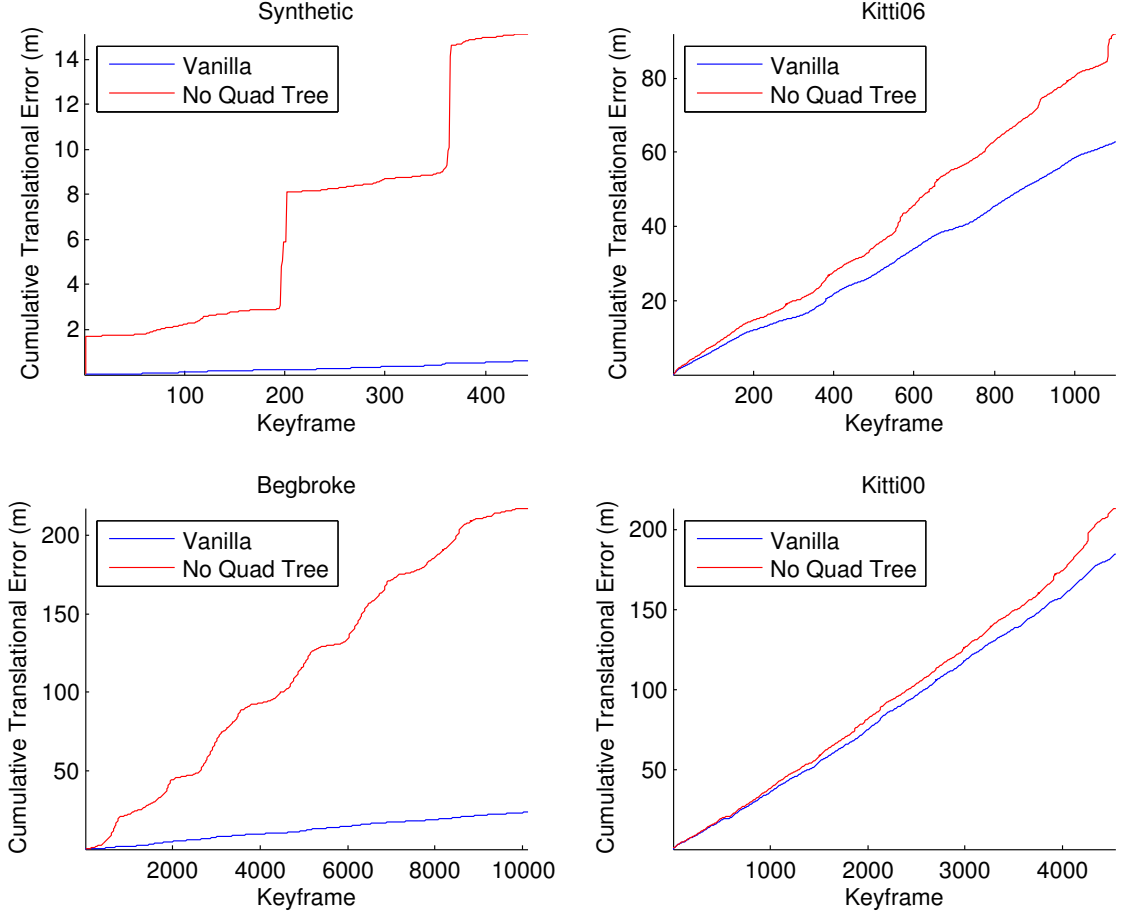


Figure 4.7: Cumulative translational error plots for the system with a regular feature distribution enforced (Vanilla) and without (No Quad Tree). In Synthetic and Begbroke where smaller percentages of the total image area have strong texture responses the effect is most notable as the feature distribution can become particularly concentrated.

constrained estimates.

We present results for the system with the Quad Tree disabled, so the top F features are selected from the whole image. Cumulative translational error plots vs the vanilla settings (with the Quad Tree enabled) are shown in Figure 4.7. Table 4.3 lists further results of the system with the Quad Tree disabled and the delta to the vanilla system given in parentheses. We see that on all datasets, the performance is worse. The performance degrades faster on the datasets with larger images (Synthetic and Begbroke) as the features are more likely to be poorly distributed due

Table 4.3: Results of the VO system with the Quad Tree disabled (i.e. no feature distribution enforced). The delta to the vanilla settings (with Quad Tree) are listed in parentheses.

	Synthetic	Kitti06	Begbroke	Kitti00
ρ_e	0.034 (-0.033)	0.084 (-0.027)	0.022 (-0.019)	0.047 (-0.006)
ϕ_e	0.019 (-0.019)	0.001 (-0.000)	0.000 (-0.000)	0.001 (-0.000)
$\Psi_e(\rho_e)$	406.148 (-401.536)	7.809 (-2.689)	18.402 (-16.364)	6.124 (-0.866)
$\Psi_e(\phi_e)$	135.334 (-134.314)	16.912 (-8.394)	9.122 (-2.002)	11.048 (-2.745)
T/F	0.062 (-0.004)	0.026 (-0.002)	0.038 (-0.001)	0.029 (0.001)

to higher levels of detail. We also see three cases of catastrophic failure in the Synthetic dataset. This is caused by the very strong textures resulting in very poor feature distributions, which leads to catastrophic estimation failures. We also note the timing cost of using a Quad Tree is minimal, so its inclusion is fully justified.

4.4.3 Rethreading

Rethreading is a way to give features a second chance to be matched by performing two temporal matching loops. The first loop generates matches using BRIEF descriptors with Efficient Second-order Minimisation (ESM), and computes a solution using RANSAC and least squares. ESM is discussed in detail in Section 4.4.5. This motion estimate is then used to seed the second data association loop, where any features that failed to match in the first round are re-matched using simple maximal sum of absolute difference (SAD) scoring¹ (followed by ESM). Searching is done in a very restricted search region (typically a 4×4 pixel window centered on the predicted location). A second round of least squares is then performed including any new matches found from the rethread step. Rethreading is useful for picking up features which failed BRIEF matching and is fast due to the very small search window and low number of features. We define the rethread ratio as the number of features that were marked as outliers by the initial estimation step (either from

¹SAD produces a simple similarity score between two image patches.

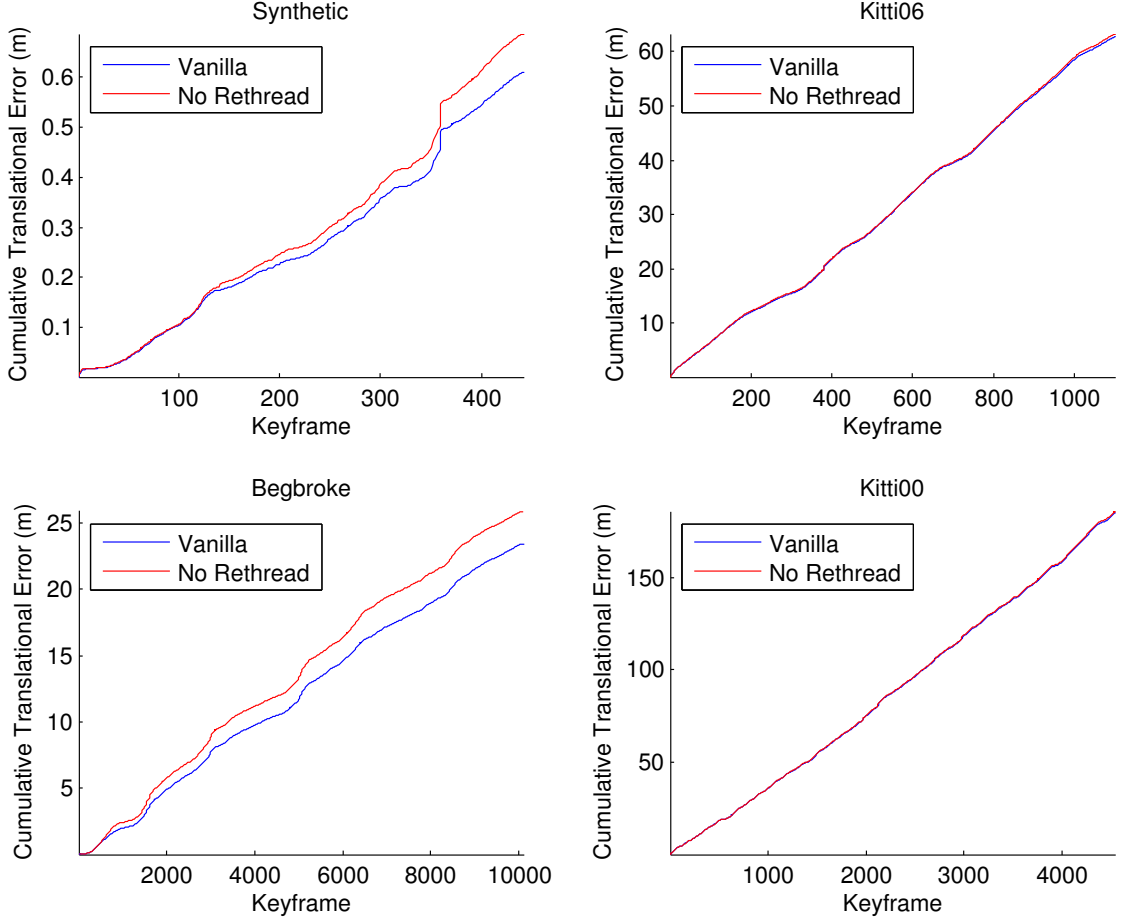


Figure 4.8: Cumulative translational error plots with rethreading enabled (Vanilla) and no rethreading.

RANSAC of least squares) divided by the total number of landmarks:

$$rr = \frac{F_{outliers}}{F_{total}} \quad (4.4)$$

We plot cumulative translational errors in Figure 4.8 and further results are listed in Table 4.4. The delta to the vanilla settings is shown in parentheses. We note that the system benefits from rethreading in the Synthetic and Begbroke sequences but does not have much of an impact for the two Kitti sequences. The lack of performance increase when using rethreading on the Kitti sequences is probably because the images are a) smaller, and b) richer in features compared to the other

Table 4.4: Performance results for VO without rethreading. Delta to vanilla settings (with rethreading) in parentheses.

	Synthetic	Kitti06	Begbroke	Kitti00
ρ_e	0.002 (-0.000)	0.058 (-0.000)	0.003 (-0.000)	0.041 (-0.000)
ϕ_e	0.000 (-0.000)	0.001 (-0.000)	0.000 (-0.000)	0.001 (-0.000)
$\Psi_e(\rho_e)$	4.813 (-0.201)	5.172 (-0.051)	2.122 (-0.084)	5.294 (-0.035)
$\Psi_e(\phi_e)$	1.106 (-0.086)	8.989 (-0.472)	7.326 (-0.206)	8.402 (-0.100)
T/F	0.057 (0.001)	0.022 (0.001)	0.035 (0.001)	0.028 (0.001)

two datasets. This means better quality features are available all over the image and BRIEF can match them with a high success rate. Compare this for example to the Begbroke dataset which always contains a large section of road. This produces lots of high frequency features which are difficult to match without a strong motion prior. These types of features are not correctly matched in the first round but with the aid of rethreading can be correctly picked up and then aid the solution. The computational costs of rethreading is minimal, $\sim 1\text{ms}$.

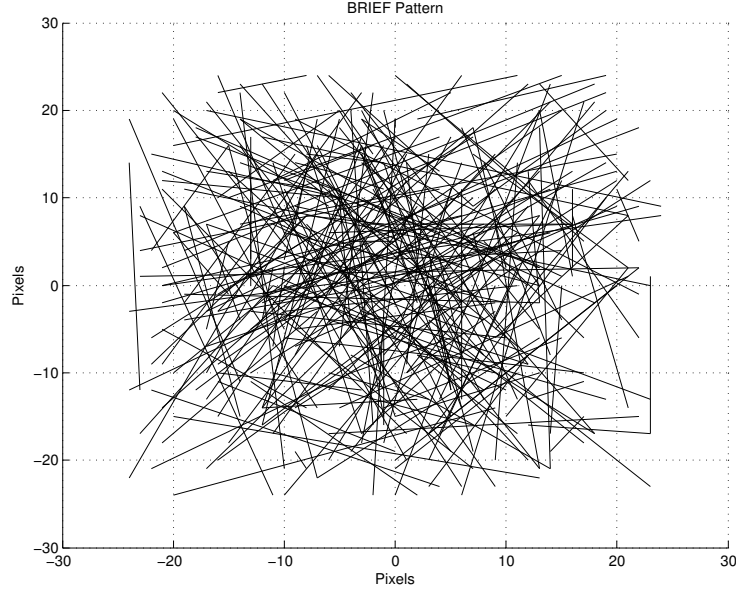


Figure 4.9: A sample BRIEF pattern using the “G II” model, an isotropic Gaussian centered on the feature.

4.4.4 The BRIEF Feature Descriptor

BRIEF is a recently developed binary based feature descriptor [11]. It does not provide point of interest extraction so needs to be combined with a feature extractor such as FAST. Given a point of interest, the associated descriptor vector is computed by evaluating a set of predefined binary tests centred on the point of interest. The intensities of pairs of pixel locations are compared and the result is returned. Given the local image patch P surrounding a feature, the t th test is the comparison of two pixel locations, $\mathbf{q}_1^t$ and $\mathbf{q}_2^t$:

$$\tau_t(P) = \begin{cases} 1 & \text{if } \mathcal{S}(P)(\mathbf{q}_1^t) < \mathcal{S}(P)(\mathbf{q}_2^t) \\ 0 & \text{otherwise} \end{cases} \quad (4.5)$$

where $\mathcal{S}(\cdot)$ is a smoothing function that takes a local image patch produces a smoothed version of the patch. The binary descriptor $\mathbf{d}$ is formed from t binary tests:

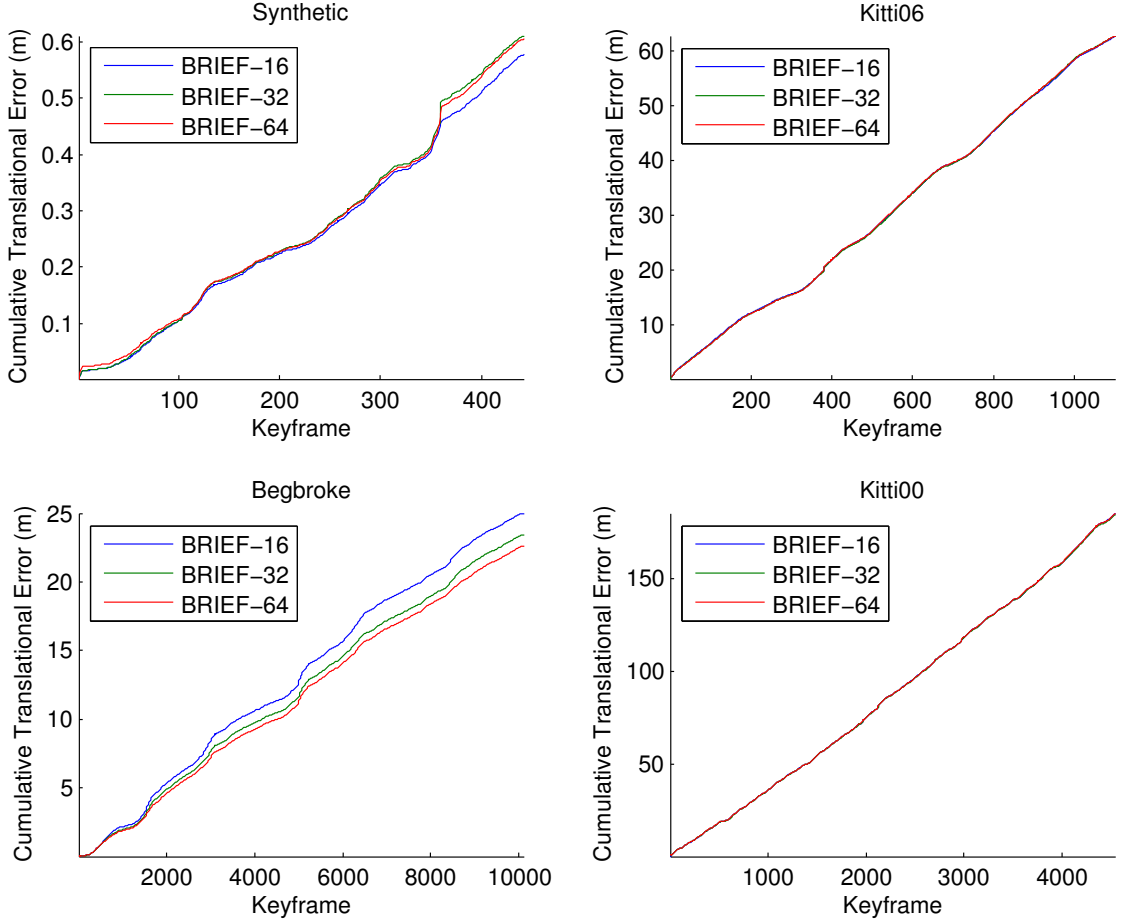


Figure 4.10: Cumulative translational error plots for BRIEF descriptors of different lengths. Interestingly BRIEF-16 is competitive when compared to longer descriptors. This is because it is “good enough” for the small changes typically experienced in frame to frame VO.

$$\mathbf{d} = [d_1, d_2, \dots, d_t], \quad d_t = \tau_t(P) \quad (4.6)$$

We provide smoothing, $\mathcal{S}(\cdot)$, via a box filter, which can be implemented efficiently using an integral image. The shape of the test distribution is important to the performance. We use the “G II” pattern [11], where the test points are sampled from an isotropic Gaussian centered on the feature position. An example of which can be seen in Figure 4.9.

The attraction of BRIEF is the speed at which point descriptors can be extracted

and matched, as well as the low memory and storage requirements. The matching score between two vectors is computed with an XOR operation followed by a bit count, which can be performed extremely quickly by taking advantage of new instruction sets such as SSE4 [48] found on recent CPUs. Descriptor comparison is fast and discriminative enough to allow for temporal matching across the whole image. This allows BRIEF to achieve a similar matching performance when compared to high quality descriptors like SIFT and SURF, while also enjoying the speed of simpler methods such as SAD matching that require a restricted search window. This restriction implies some assumed motion model and should the initial match fail, the system needs to either widen the search window or fall back on a more robust matching approach like SIFT.

The number of binary tests, t , has a strong effect on the quality and cost of the descriptor. Standard sizes for t are 128, 256 or 512, which corresponds to 16, 32 or 64 bytes per descriptor. We refer to these as BRIEF- $\{16,32,64\}$ or B- $\{16,32,64\}$. As we have shown in Figure 4.2, the computation time is dominated by BRIEF descriptor extraction. We vary the length of our BRIEF descriptor and present cumulative translation error plots in Figure 4.10 with further results in Table 4.5. Interestingly we see that the performance is not particularly affected by the choice of t . BRIEF-16 performs best on Synthetic but worst on Begbroke, and they are all roughly equal on the Kitti datasets. Reassuringly the rethread ratio (rr), which indicates how many successful matches BRIEF returns (lower is better), goes down for the longer descriptors - implying the longer descriptors are doing a better job. However this increase in descriptiveness comes at a time cost, moving from BRIEF-32 to BRIEF-64 increases the computation time on average by 17ms.

The reason for this relatively invariant performance to descriptor length is twofold. Firstly, the reported performances of these types of descriptors are typically done on

Table 4.5: Performance results for varying BRIEF descriptor lengths. T/F refers to average computation time per frame. rr is the rethread ratio.

	Synthetic			Kitti06		
	B-16	B-32	B-64	B-16	B-32	B-64
ρ_e	0.001	0.001	0.001	0.057	0.057	0.057
ϕ_e	0.000	0.000	0.000	0.001	0.001	0.001
$\Psi_e(\rho_e)$	4.425	4.612	6.130	5.109	5.120	5.122
$\Psi_e(\phi_e)$	0.996	1.020	0.988	8.642	8.518	8.614
T/F	0.042	0.058	0.089	0.019	0.023	0.032
rr	0.189	0.174	0.168	0.397	0.366	0.347
	Begbroke			Kitti00		
	B-16	B-32	B-64	B-16	B-32	B-64
ρ_e	0.002	0.002	0.002	0.041	0.041	0.041
ϕ_e	0.000	0.000	0.000	0.001	0.001	0.001
$\Psi_e(\rho_e)$	2.094	2.037	2.015	5.278	5.259	5.274
$\Psi_e(\phi_e)$	7.134	7.120	7.134	8.288	8.303	8.306
T/F	0.028	0.036	0.054	0.025	0.029	0.040
rr	0.288	0.244	0.219	0.344	0.317	0.298

datasets which exhibit strong viewpoint, rotation or lighting change². Performance degrades when these factors are maximal. However, in the case of VO, due to the high sampling rate of the camera, these factors simply do not change much from frame to frame. Therefore we can get away with a worse descriptor, because typically the conditions under which we expect it to perform are simply not that challenging. This is partly why we see BRIEF-16 being competitive with BRIEF-64. The second factor is that rethreading helps to refine any good initial guess. Provided the descriptor is good enough to get some correct matches, RANSAC and least squares will produce a reasonable solution, which can be refined with rethreading. We see in the BRIEF-16 case, rethreading has to “work the hardest” in the sense that it is successfully matching the highest percentage of features. However the final result is similar to longer descriptors.

²<http://www.robots.ox.ac.uk/~vgg/research/affine/>

4.4.5 Sub-Pixel Precision

Sub-pixel matching is the ability to locate feature positions in an image with floating point accuracy, i.e. beyond integer valued locations. Feature point extractors such as SURF and SIFT produce sub-pixel feature locations, but FAST does not and so matches need to be manually refined. For small image patches that are typically used to describe feature locations, and a good initial guess from BRIEF or patch based matching, this can be achieved very quickly using Efficient Second-order Minimisation (ESM) [65, 4].

ESM approximates second-order minimisation performance by using the Jacobian of the current estimate *and* the target patch. For most minimisation problems this second Jacobian is not known, but because we are trying to find an image position that matches a *known* target patch, it can be computed. Simply taking the average of these two Jacobians results in good performance. The Jacobians can be computed for desired parameters such as translation and scale. Using ESM has been shown to increase the performance of a VO system [71].

In Figure 4.11 we show the VO performance with and without sub-pixel refinement and list further results in Table 4.6 with the comparison to the vanilla settings given in parentheses. We can see that using sub-pixel measurements brings performance increases on all of the sequences. The best increase (in terms of translational error) is on the Synthetic dataset. This is probably because there is less noise in the images (compared to the real world datasets) which aids sub-pixel refinement more. Sub-pixel matching comes at the modest cost of ~ 5 ms and is well worth doing.

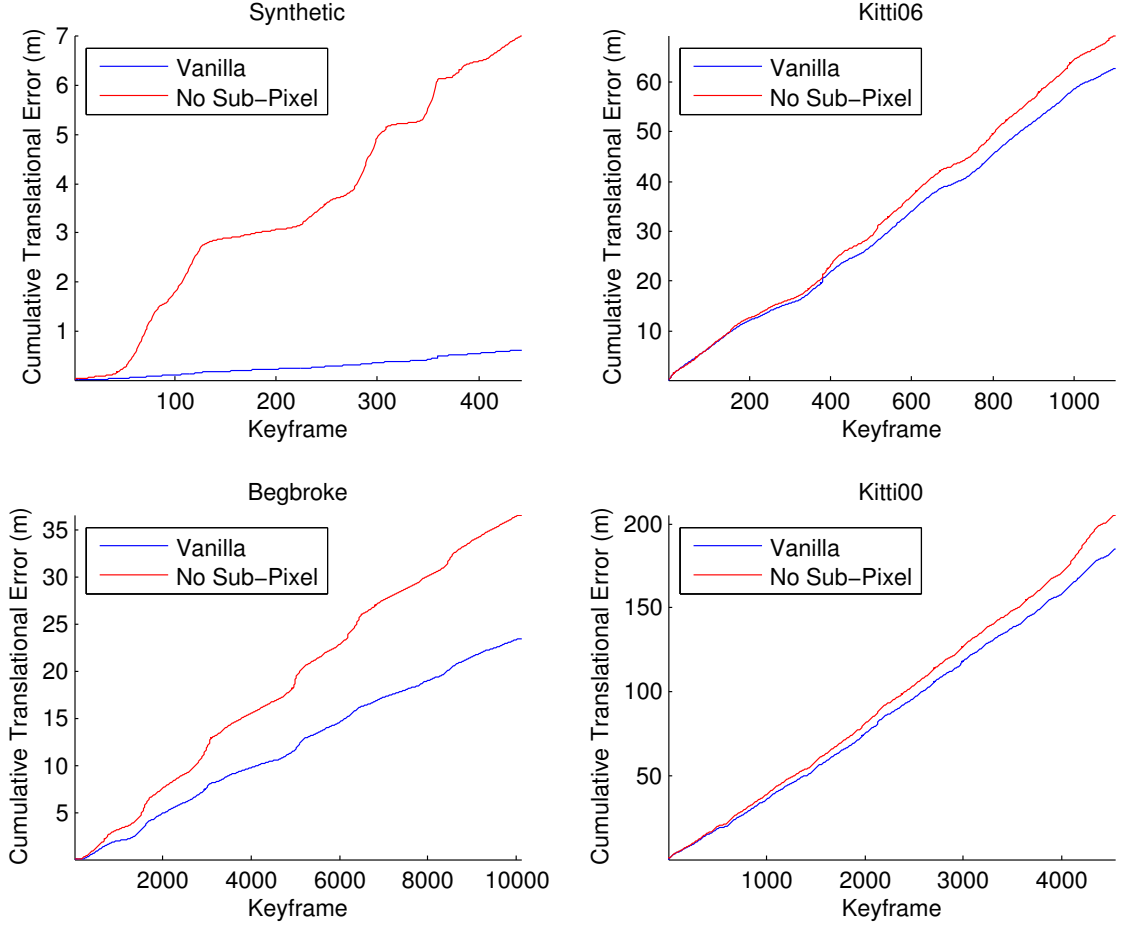


Figure 4.11: Cumulative translational error plots with sub-pixel refinement enabled (Vanilla) and no sub-pixel refinement.

Table 4.6: Performance results for VO without sub-pixel feature refinement. Delta to vanilla settings (with sub-pixel refinement) in parentheses.

	Synthetic	Kitti06	Begbroke	Kitti00
ρ_e	0.016 (-0.015)	0.063 (-0.006)	0.004 (-0.001)	0.045 (-0.005)
ϕ_e	0.001 (-0.001)	0.001 (-0.000)	0.000 (-0.000)	0.001 (-0.000)
$\Psi_e(\rho_e)$	17.134 (-12.522)	5.608 (-0.488)	3.232 (-1.194)	6.021 (-0.762)
$\Psi_e(\phi_e)$	9.419 (-8.399)	18.616 (-10.098)	10.039 (-2.919)	12.634 (-4.331)
T/F	0.051 (0.007)	0.019 (0.005)	0.032 (0.005)	0.024 (0.005)

Table 4.7: Kitti Vision Benchmark Suite ranking table: top three results as of September 2012. Our method is third in translational error and first in rotational error.

Method	Translation	Rotation	Runtime
GT_VO3pt	2.21 %	0.0117 [deg/m]	1.26 s
VISO2-S	2.28 %	0.0154 [deg/m]	0.05 s
Our Method	2.45 %	0.0109 [deg/m]	0.05 s

4.5 Results on the Kitti Vision Benchmark Suite

We also ran our VO system on the Kitti Vision Benchmark Suite [38]. We provide the average translational and rotational error plots from their analysis tools in Figure 4.12. They use the same translational and rotational error metrics that we use, except over multiple segment lengths. We also replicate the ranking table as of September 2012 in Table 4.7. Our method is ranked third by translational error and first by rotational error. The top ranked translational method uses the computationally expensive SIFT descriptor for data associations, this is reflected in the much higher runtime cost: 1.26 seconds compared to the 0.05 seconds of our approach. Our method is similar in approach to that of “VISO2-S” in that we both use relatively simple sparse features and RANSAC for outlier rejection to achieve realtime performance. The top three methods are all close in terms of performance, we only perform 0.24% worse in translational error and are only slightly ahead in rotational error compared to the top method.

4.5 Results on the Kitti Vision Benchmark Suite

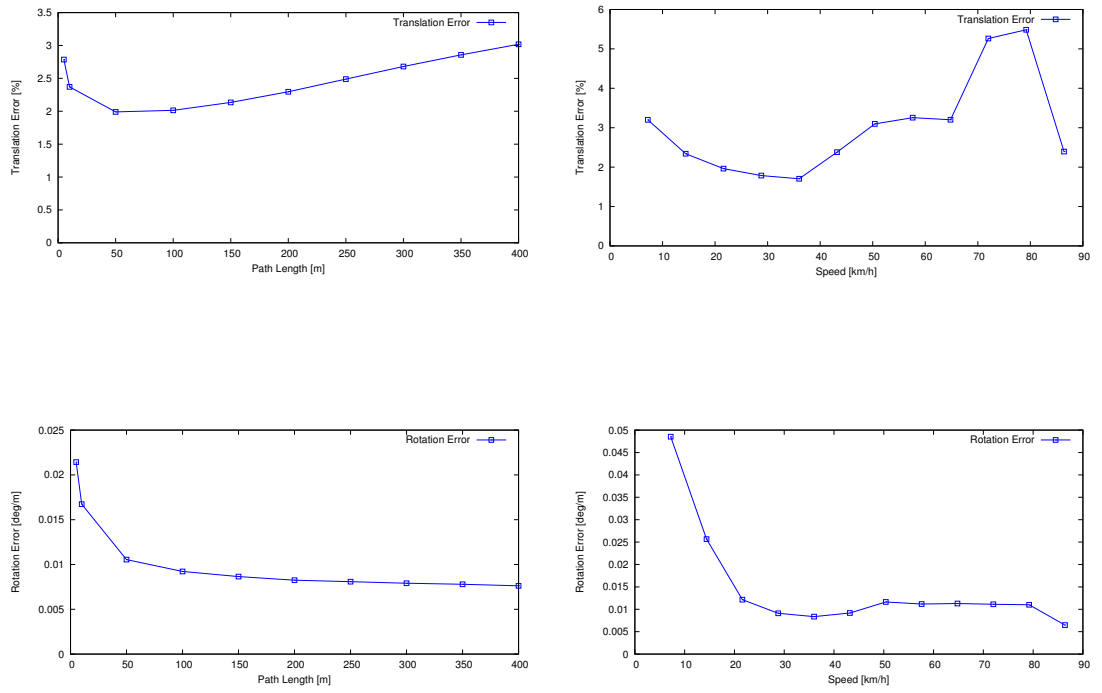


Figure 4.12: Test average results for the Kitti Vision Benchmark Suite. These plots are the output of their analysis and plotting tools. They use the same error metrics as us, except over multiple path lengths.

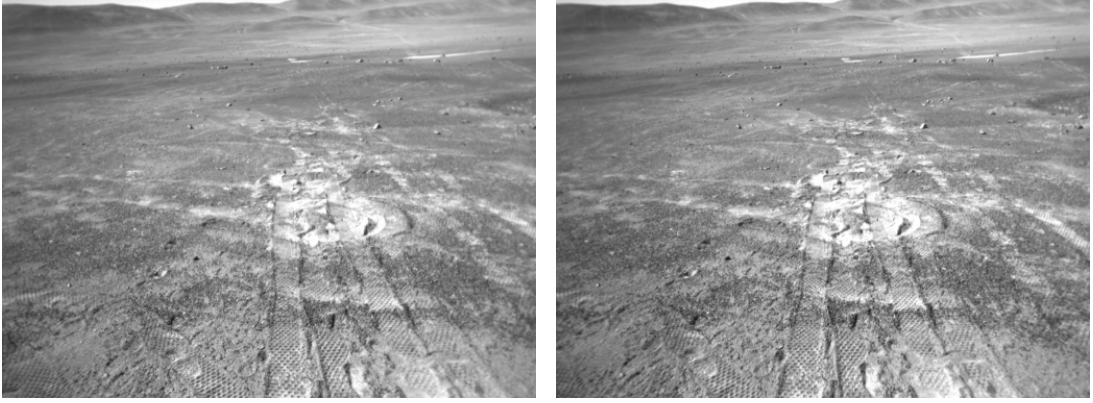


Figure 4.13: A stereo pair from one of the SCISYS datasets collected in Chile. The area was chosen for its similarity to the Martian landscape. The VO system described in this thesis was field tested by SCISYS as part of an autonomous rover project. Images courtesy of SCISYS.

4.6 Results in Other Places

The VO system described in this thesis was used by SCISYS PLC in field trials, both in the UK and Chile. The project, in conjunction with the European Space Agency (ESA), is aiming to develop a Martian rover with higher levels of autonomy. The choice of Chile is due to its similarity to the Martian landscape - Figure 4.13.

We include some qualitative results from their usage in Figure 4.14. Their rover drove two loops of a quarry circuit, following the exact same path on the second pass apart from a slight deviation at the end. We present axis aligned plots of the estimated 3D trajectory. While no ground truth is available for this data, we can see the estimated path on the second loop largely overlaps with the first - Figure 4.14b. Plots including z show there is some deviation in height on the second loop. Interestingly, this positive bias in z has been observed by Lambert [57], when operating their VO system in similar planetary environments. They noted that planetary scenes generates oblique feature point configurations, landmarks towards the top of the image are far away, while landmarks towards the bottom of the image are closer to the camera. Stereo cameras exhibit a range bias which is a

function of the disparity measurement, the larger the range, the bigger the bias [93]. Lambert suggests that the larger biases associated with the landmarks at the top of the image cause the bias in z . They attempt to estimate the bias but note this is difficult due to the effective truncation caused by the stereo matching process. Through simulation they can produce bias trajectories in y if the appropriate oblique landmark configurations are used, and note that the problem disappears if purely random landmark distributions are used.

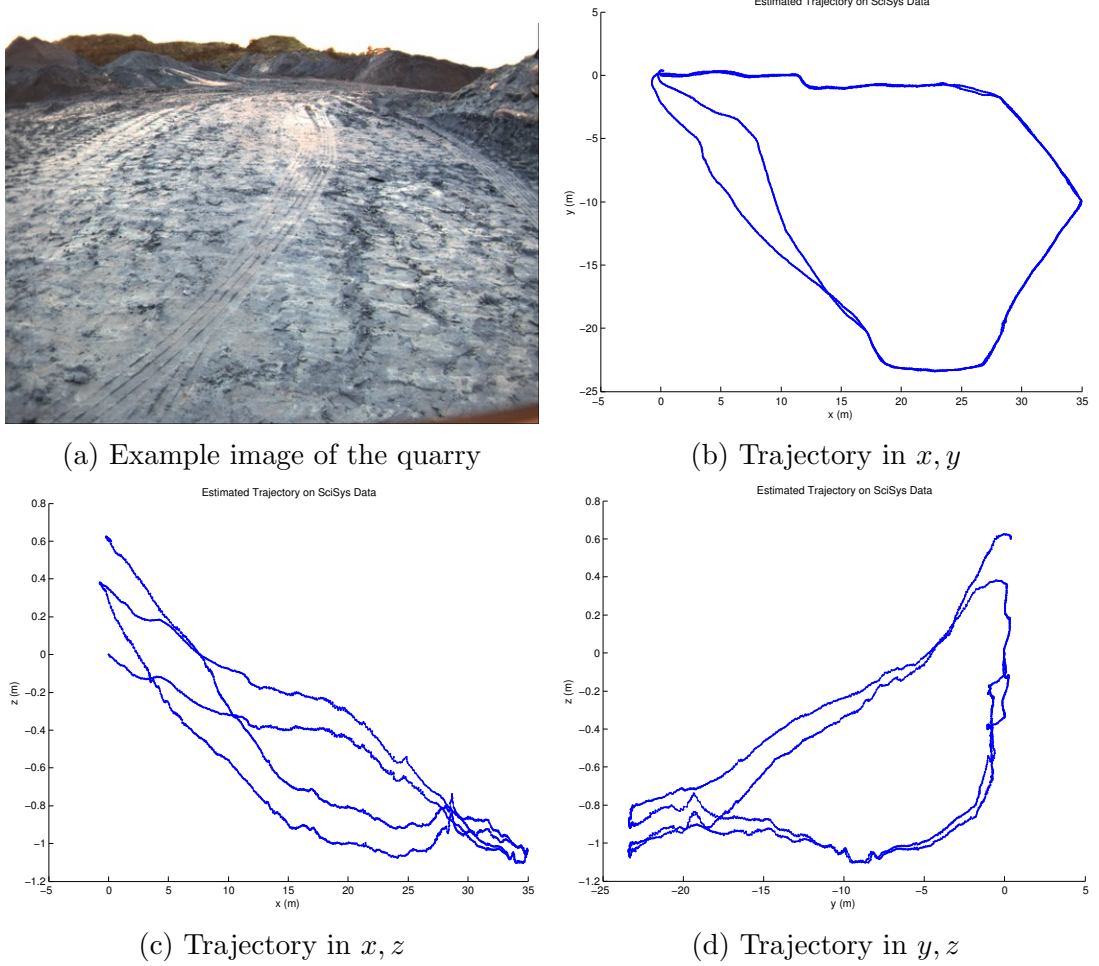


Figure 4.14: Qualitative results of the VO system which was evaluated by SCISYS. Two loops of a quarry environment (a), were driven with a slight deviation from the original path on the second loop (b). Estimates in z are also shown (c), (d). While no accurate ground truth is available, with the knowledge that the robot drove the same route on the second loop, the system is clearly performing well given the overlapping trajectories. This is especially prominent in (b) - note that the robot did take a slightly different route for part of the second loop so they do not exactly overlap. Image data courtesy of SCISYS.

4.7 VO Parameters

We now provide a non exhaustive list of parameters in the VO system and give a brief comment on the systems sensitivity to their values.

- **Image Size:** We have found the size of the images used has a strong impact on the runtime of the system. This makes sense as the image processing steps form a large portion of the time budget. However we have also found the system still performs well at lower image resolutions. For example, we run the Bumblebee cameras at 512x384, however they do offer 1024x768 images.
- **FAST Corner Extraction Threshold:** As discussed, we want this to be low to generate lots of corners, even in areas that are not particularly textured. If the VO is operating in feature rich environments this could be raised, but for a general purpose system we set it low and then use non maximal suppression to remove the resulting clusters that form around the stronger features.
- **Non Maximal Suppression:** We found having no non maximal suppression does not increase the performance, but does increase computation time as more BRIEF features need to be extracted.
- **BRIEF Descriptor Size:** As discussed in Section 4.4.4 we find that longer BRIEF descriptors generally perform better but at higher computational cost.
- **Number of RANSAC Trials:** RANSAC evaluations are relatively cheap so many trials can be performed, ensuring a good solution is likely to be found.
- **Outlier Threshold:** This value determines which matches are used in the least squares optimisation. A balance must be struck between providing a sufficient number of measurements versus not introducing poor matches that corrupt the solution.

4.8 Practical Considerations

A naive implementation of VO would attempt to keep all keyframes in memory for all time. When working over short to modest time frames it is possible to get away with this approach given the amount of memory and computing power modern computers now routinely contain; but these resources are not infinite and if we want a solution that can run indefinitely a modified approach is required. There are two solutions, depending on the requirements of the VO.

If the VO is only required to provide odometry, i.e. the relative motion of the camera, then the historical output is not needed and can simply be forgotten. In this case a fixed number of the latest keyframes is remembered and the rest deleted. If landmarks that have been observed recently are stored relative to a keyframe that is about to be deleted, these need to be transferred to another keyframe, otherwise they can also be removed.

If a more complete navigation system is required, which supports things like loop closure detection and localisation, the historical output is important and needs to be stored for future use. In this case regions of the map that are not in use can be stored on the disk³ and loaded as needed, allowing the application to conform to a constant memory and time budget.

We show results of a naive versus a constant time implementation in Figure 4.15. Here we limit the constant time implementation to 100 keyframes. The naive implementation was terminated by the operating system after 5.6 hours due to memory consumption, where as the constant time version ran for 11.7 hours before being manually stopped.

³If enough keyframes are added this will also fill up, although hard drives are typically much larger than RAM.

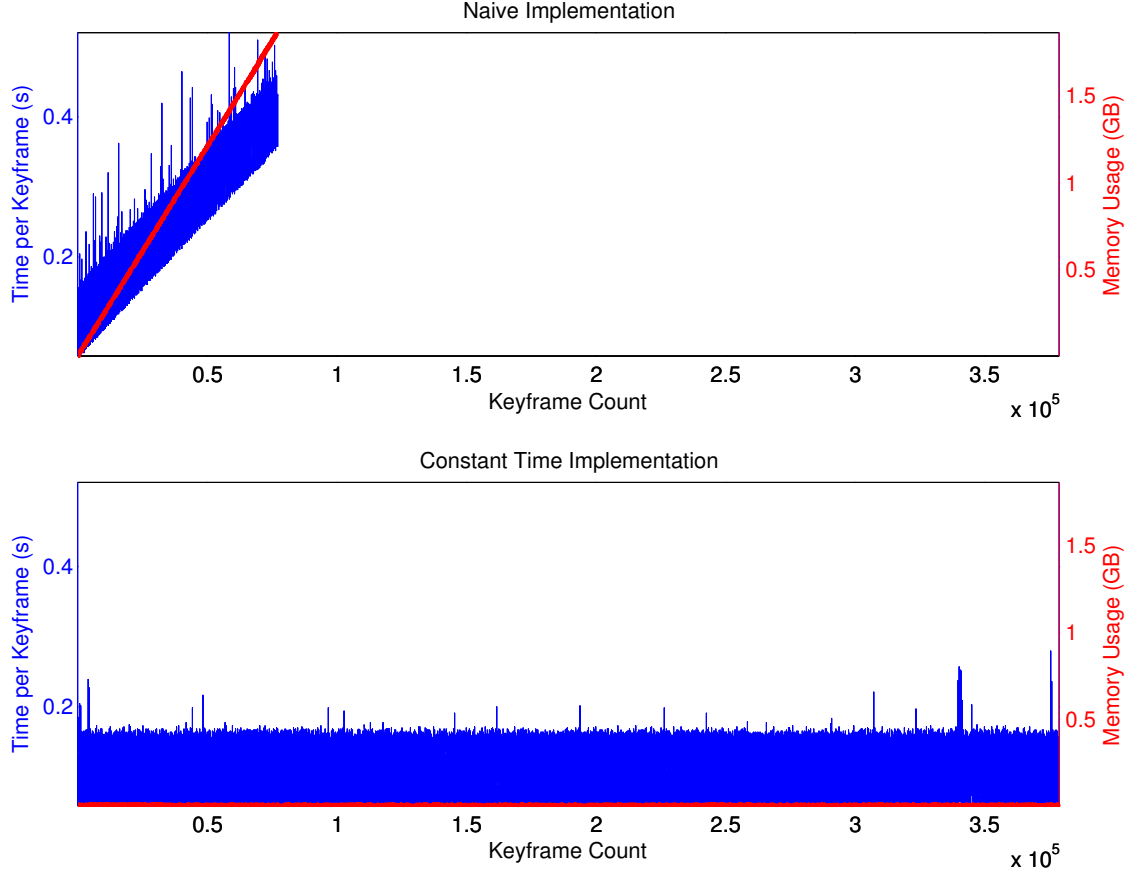


Figure 4.15: Here we show the timing (blue) and memory usage (red) of two VO implementations. The graph axes are equal on both plots for easy comparison. The upper plot is a naive implementation which attempts to keep all keyframes in memory. The result is linear growth in computation and memory costs. The process ran for 5.6 hours, created 77,011 keyframes and used 1.87GB of memory before the operating system killed it. The bottom plot shows a constant time implementation which only keeps the latest keyframes in memory, resulting in constant computation and memory costs. This process ran for 11.7 hours, created 378,759 keyframes and reached a peak memory usage of 0.012GB. The experiment was stopped manually. These experiments were carried out using a Pentium 4.

4.9 Conclusions

In this chapter we have carried out extensive testing of our stereo visual odometry system using a range of datasets - two of which are publicly available. By removing individual components and demonstrating that the performance immediately drops we have shown that our system is the result of many well engineered design choices working together rather than one key component providing all of the quality. We have also included results of its use by third parties and discussed practical issues of long-term operation.

Chapter 5

Lifelong Navigation By Experience

5.1 Introduction

This chapter introduces a new approach to lifelong navigation using the idea of “experience”. In the previous chapters we have described and tested a visual odometry system that we can use to generate locally accurate maps. Like most previous works on the subject of navigation, these maps are useful if we assume the world does not change. Similar appearance is required as the already difficult data association problem becomes impossible if the same place in the world simply looks different. In reality, this assumption does not stand up for the majority of environments we expect autonomous robots to operate in. The world’s appearance can be affected by numerous sources, operating at different time scales. Examples include dynamic objects, changing weather conditions, the time of day, lighting variations and seasonal change.

The majority of previous mapping and navigation approaches compute and save a map on their initial traverse of an environment. They then assume the environment maintains a sufficiently similar degree of appearance to enable enough correct data associations for localisation. The latest landmark observations are fused with old



Figure 5.1: The same place can look very different, depending on when it is observed. This variation may come from structural change, lighting conditions or through the shifting of the seasons. Attempting to produce a single trajectory from multiple experiences may be difficult due to lack of correspondences.

ones, and new landmarks are forever added to the single map - all of which is forced to happen in a coherent frame of reference. However if lifelong operation is expected, and the ability to maintain sufficiently correct data associations is removed, some challenging and difficult questions need to be answered. How do we manage the model of a place as summer drifts into autumn, causing a slow creep in appearance? Should we attempt to capture all features into a single monolithic map, even though they will all never be seen together? Is it possible to undertake some unifying data fusion activity to yield a monolithic map in which we can localise? Further, what should be done if localisation is completely unsuccessful due to drastic appearance change, e.g. it snowed? This chapter attempts to answer some of these questions. We argue that instead of creating an ever expanding monolithic map, we should collect distinct and independent experiences of the world as required. We shall not attempt to combine or fuse these experiences to produce a consistent map. We will not force them to be coherent. Taking this approach has several advantages, but also introduces some new challenges. Although we describe our framework using the vision techniques from previous chapters, it is actually agnostic to the

sensing modality and could be used with other sensors, such as laser range finders. We only require the ability to navigate locally, and be able to “close the loop” - recognising when we have returned to a previously visited place. Both of these are proven competencies. Metric trajectory estimation systems using laser [7] and vision (discussed extensively in previous chapters) have been demonstrated while large scale topological loop-closure systems using vision have been achieved [21] and crude GPS with sub 10m error is often available in outdoor environments. Given these capabilities, we now cover some of the previous work in the area of long-term navigation before presenting our approach and results.

5.2 Background

In recent years SLAM solutions have taken large steps forward, achieving results over successively greater distances. Metric ego-motion estimation systems have been demonstrated performing over several of kilometres [78, 53], while topological approaches have been shown successfully operating over 1,000km [21]. However the problem of achieving similar results over large *time* scales has received relatively little attention, and is becoming a major obstacle to the realisation of lifelong autonomy.

Before we review some of the previous approaches to the long-term navigation problem, we take a moment to discuss the issue of dynamic objects and dynamic environments. The ego-motion estimation algorithms used in exploration or localisation assume the world is static, and so the problem of dynamic objects in the scene is one the SLAM community has been aware of for some time [2]. In this context “dynamic objects” refers to other things moving in the world, e.g. humans and cars. Most modern approaches use probabilistic filters to reject measurements that do not fit with the consensus as outliers. For example the VO system described

in previous chapters uses RANSAC to compute an initial guess of the camera motion, enabling the system to be robust to outlier matches provided by the front-end. Some systems go further, explicitly tracking the moving objects [110, 92]. These solutions are successful provided the amount of dynamic motion in the scene is sufficiently small. In the context of lifelong navigation, objects which are not necessarily dynamic when they are first encountered, can easily move between visits, causing dynamic scenes. An obvious example are parked vehicles. These cases can be solved to an extent by the same techniques described above, by assuming that outliers can be detected and rejected. However if the scene changes sufficiently this approach begins to break down. For example, consider the case of a car park. The configuration of the parked vehicles will change daily, leading to regularly varying and ephemeral scene appearance.

We now review some of the approaches to tackling long-term navigation and localisation, which has been studied using both vision and lidar sensors.

5.2.1 Long-Term Navigation

Burgard *et al.* [10] map the environment using a laser scanner, which they split into sub-maps up to $20m^2$. Each time a sub-map is revisited, the associated sequence of observations is recorded. The set of observation sequences are then clustered using a fuzzy k-means algorithm, and the Bayesian Information Criterion is used to select the model that has the best compromise between fitting the data and number of clusters. This is repeated for every sub-map. When performing localisation in a sub-map, the choice of cluster becomes another variable in their particle filter solution. Results from a robot operating within an office space are presented, where typical changes are door configurations (open or closed). By including the model of the dynamic environment, the localisation error is improved compared to the standard static map approach.

Biber and Duckett [5] also describe a laser based system for dealing with structural change in the environment. They highlight the issue that using simple recency-weighted averaging leads to measurement predictions that have never been seen, as structural change tends to occur in discrete step sizes. They approach the problem by maintaining local maps for a number of points in the environment. Each local map stores a set of measurement sequences. The size of the sequences is fixed, and the resample rate (how quickly new data replaces old data) is varied for each set. This creates a local map that has both short term and long-term structure. Localisation is then performed against a map synthesised from the best fitting time scale sequences of nearby local maps. In their experiments the shortest half-life is a few seconds, resulting in a map which is effectively the current sensor measurements. The longest half-life is nearly two weeks. They show improved localisation accuracy over the period of several weeks for a busy lab environment when compared to the static map captured on the first day.

Milford and Wyeth’s biologically inspired RatSLAM system [72] maintains a non-metric topological graph referred to as an “experience map”. A node in the graph is referred to as an experience and contains a view of the world at that point, a pose estimate and transition information to other experiences (from wheel odometry). This map is relaxed over time to minimise the difference between global node positions and the relative transitions. A new experience is created when the scene is not sufficiently similar in appearance to any of the previous ones. This allows experience creation when the robot visits a new place in the world, and when revisiting areas that have changed. To prevent their map becoming overly saturated with experiences they prune it by placing a regular grid over the map and only allowing one node to exist per cell. The single surviving node is picked at random, they find this produces similar results to other deletion metrics such as connectivity or recency. The system is shown performing successfully for a two week period in

an office environment.

Konolige and Bowman [54] develop their view-based maps system [55] to deal with a changing world. The original system creates a skeleton graph of keyframes from a VO system, while a place recognition component provides loop closure information. The map is in a single frame of reference and is incrementally optimised via Toro [39], an optimisation tool for graph-based SLAM problems. Over time the environment changes, resulting in the current view failing to match against previous keyframes. This causes new keyframes to be added to the system. To keep the density of keyframes down in their graph, for a local metric neighbourhood a maximum number of allowed keyframes is defined. Their deletion scheme favours maintaining view diversity (based on a keyframe similarity metric) and then recency. Their system is shown operating in a lab environment that includes moving people and furniture, and changing lighting conditions over a few days.

Dayoub and Duckett [24] present an image based variant of Biber and Duckett’s previous time-scale approach for topological mapping with an omnidirectional camera. The environment is mapped by a set of known points called reference views. Each view has two explicit memory time scales, short term and long-term. Features seen enough times in the short term memory advance to the long-term memory, while those not repeatedly seen are forgotten. Similarly in the long-term memory, features not seen for a while are forgotten. They show the current view has a higher similarity to these adaptive reference views compared to static reference views over a 9 week period. In this work the reference views are fixed and known, as is the position of the robot relative to the views.

Recently Taneja *et al.* [102] have detected structural change in their dense reconstruction of urban environments, allowing them to only update their outdated parts of the model. By incorporating semantic information they are able to ignore short term changes caused by things like cars and people. In this work they assume they

are able to ignore weather and lighting conditions, given that they are consistent across visits.

5.2.2 Teach and Repeat

Finally we note the work of Furgale and Barfoot [36] who develop a teach and repeat system using stereo vision in an outdoor setting. While not necessarily long-term mapping, they do encounter and highlight some relevant issues. During a teach phase, the robot is manually driven along a desired route, creating a series of connected sub-maps using VO. For the repeat stage, a localisation module is used to retrace the original path. The system is shown performing successful repetitions of taught routes, totalling more than 32km length, in both urban settings and the Canadian High Arctic. They highlight feature repeatability issues caused by changing lighting conditions (using SURF). After ten hours a repeat pass was unable to localise against the original taught pass due to changes in lighting conditions. Further, they found routes taught on overcast days are problematic if used for localisation on sunny days (and vice versa). They do not attempt to capture changing appearance from the original traverse.

We also note the recent work of Lategahn and Stiller [59] who localise a road vehicle in a large sparse prior map created from a previous drive. They show successful localisation over two ~ 1 km trajectories but do not attempt to model changing conditions.

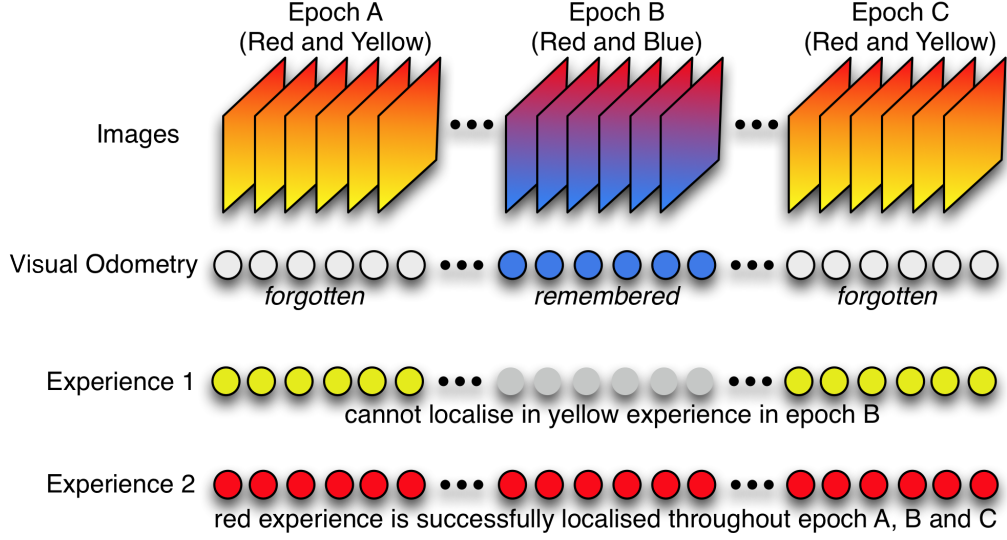


Figure 5.2: An overview of our approach. The visual odometry (VO) system continuously consumes the live image stream. In parallel a series of localisers attempt to localise each live frame in their own experience. In epochs A and C both localisers successfully localise the frames in their experiences, so the VO output is forgotten. In epoch B, localisation is only successful in one saved experience (experience 2), which is deemed too few, so the VO output is saved in a new experience.

5.3 Experience Based Navigation

5.3.1 Overview

We begin with a high level overview of our framework and highlight some of the differences to previous approaches. The main building block used in this work is the VO system developed in the previous chapters. This gives us a sequence of *relative* poses and the associated constellation of visual features. While operating, regardless of whether localisation is currently successful, VO is always performed on the live image stream. On the initial visit to a new area we save this output of the VO like most systems. We refer to the whole sequence of saved poses and related features as an “experience”. When revisiting the area the robot attempts to use the live stream of images to localise in the saved experience. If at any point this is unsuccessful, a new experience is created based on the current appearance of the world. As the robot continues, still saving to this new experience, it is also trying to re-localise

in its previous experience(s). If this is successful at any point, saving is stopped and the system returns to localising in its previous experience. An overview of the approach is shown graphically in Figure 5.2.

Importantly this methodology causes the system to remember more representations for regions that change often, and fewer for regions that are more static. This results in the system naturally capturing the varying complexity of different parts of the world. We call the collection of all experiences the “plastic map”. Note that we handle new routes and complete localisation failures seamlessly - indeed it is the failure of localisation which drives the saving of a new experience. This is because we make the assumption that our localisation fails because of bad or unsolvable data association - what was there before is simply not there now. Making experience creation a function of successful localisation as opposed to some arbitrary image similarity metric, allows it to be directly influenced by what the system is trying to achieve - successful navigation. Choosing an image similarity metric could result in experiences being saved when the current ones are sufficient for localisation, or vice versa.

Another aspect of our experiences is that we keep them independent in terms of features and frames of reference. We only look to link them topologically. Once an experience is created its poses and features are not updated, merged or deleted based on subsequent observations of that place, the experience is fixed, and is either successfully localised against or not. Taking this decision means we can avoid the difficult data association issues that would arise. Further, we do not force experiences that cover the same physical place in the world to live in the same reference frame, relative or global, still we only link them topologically. Again this means we do not have to coerce two different trajectories to be consistent via questionable data associations. The result of these design choices means experiences are independent and so localisation across multiple experiences can be trivially parallelised. However

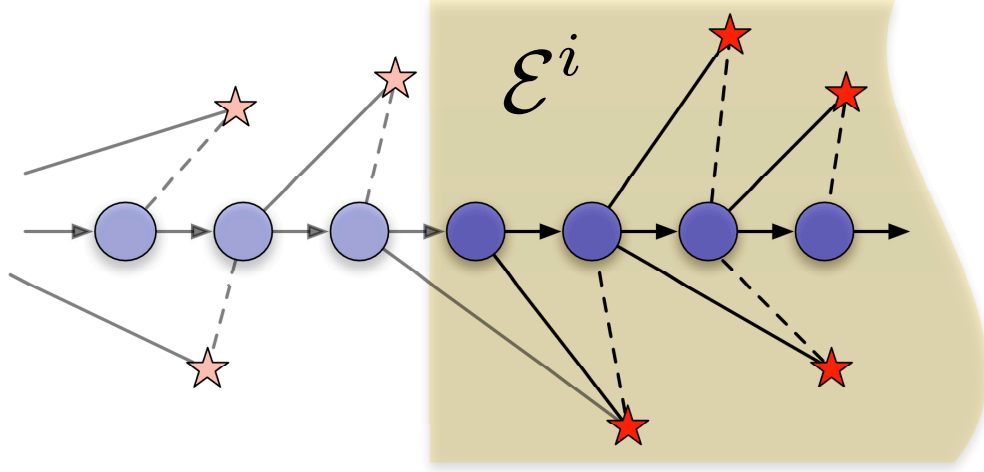


Figure 5.3: An experience is simply the saved output from the visual odometry system. The shaded area is stored as a new experience $\mathcal{E}^i$ while the faded section is forgotten. The experience consists of keyframes (blue circles) with their attached landmarks (red stars) and the transforms between the keyframes.

the lack of a single frame of reference introduces a new challenge of how we find our position in multiple experiences. This issue is addressed in Section 5.4 but we first formalise the notion of an experience, a localiser and how these are used for navigation.

5.3.2 Experiences

In the previous chapters we introduced a VO system which we briefly recap here. The VO operates on a sequence of stereo frames $\mathcal{F}^k = \{\mathcal{F}_0, \dots, \mathcal{F}_k\}$, at time k a stereo frame is processed and a camera node n_k is linked to node n_{k-1} by a 6 degree of freedom transform $\mathbf{t} = [x, y, z, \theta_r, \theta_p, \theta_q]^T$. If the frame, $\mathcal{F}_k$, initialises new landmarks, these are stored relative to n_k . We denote the g th such landmark attached to n_k , where g is a global index¹, as $\mathbf{p}_{s=k}^g = [x, y, z]^T$ - a vector stored relative to the instantiating stereo frame, $s = k$. Finally, n_k also contains a list of all landmarks observed in $\mathcal{F}_k$, many of which will be attached to other nodes - the

¹So every landmark has a unique ID.

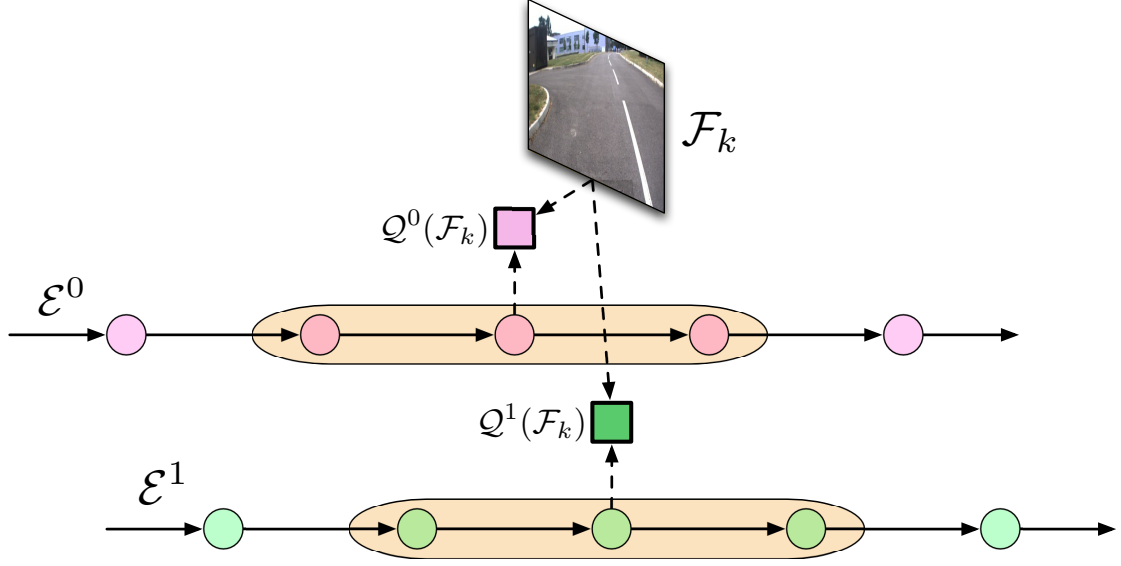


Figure 5.4: Each experience, $\mathcal{E}^i$ has an associated localiser $\mathcal{Q}^i$. The localiser takes as input the current frame and attempts to match it to its assigned experience. The landmarks for matching are taken from a local region (shaded ovals) around the previous position in the experience.

ones in which they were initialised. We often wish to express a landmark stored in frame s in a different frame, i . Transforming a landmark so it is relative to a frame i is done with Equation (2.3).

The VO system runs continuously on the live frame stream. When this needs to be saved (see Section 5.3.4) a new experience $\mathcal{E}^i$ is created and the output from the VO system is stored in this experience. $\mathcal{E}^i$ then, is simply a chain of camera keyframes, which are represented as nodes, the inter-keyframe transforms and associated 3D features. We refer to nodes in experiences as $\mathcal{E}_m^i$. Later we will explain how these chains are related (topologically) to form in concert a plastic map.

5.3.3 Localisers

We now introduce the idea of a localiser. Each saved experience is assigned a localiser. Given a live frame $\mathcal{F}_k$, its task is to calculate the transformation from the frame to a node in the experience. It operates in a very similar way to the live VO

system except the proposed landmark set comes from the saved experience, not the previous live frames. The landmarks are taken from the local region surrounding the previous position in the experience. Additionally the localiser does not attempt to add or update landmarks in either the current VO output or the saved experience. It is completely passive in terms of its impact on both.

An important competency of the localiser is its ability to tell if it is “lost”. This happens when the incoming frame can no longer be localised in the previous experience. There are many ways this can be calculated. Possibilities include the number of landmarks found and/or classified as inliers, and comparisons with the current VO output. We use the latter, which is described in Section 5.5. The output of each localiser at each time step is a binary result indicating if it is still successfully localised:

$$\mathcal{Q}^i(\mathcal{F}_k) = \begin{cases} 1 & \text{if localised} \\ 0 & \text{if lost.} \end{cases} \quad (5.1)$$

If successful, the localiser can be queried for the node in the experience that $\mathcal{F}_k$ was nearest to.

$$\mathcal{E}_m^i \leftarrow \mathcal{Q}^i() \quad (5.2)$$

Once a localiser is declared lost, it stays in this state until it receives outside assistance, discussed in Section 5.4.

5.3.4 Navigating by Experience

We now explain how these experiences are leveraged to achieve long-term navigation in changing environments. While running we always perform VO on the live image stream. The question of whether this is saved as a new experience is a function of

the result of the localisers running on the current experience set. We define N to be the minimum number of successful localisers running at any point in time. If N or more localisers are successful, we believe our current representation of the local region to be sufficient and discard the VO output. However when the number of successful localisers falls below N , we create a new experience from the VO output. This continues until the number of successful localisers returns to N or above, when saving is stopped.

Experience creation is driven by the success or failure of localisation to prior experiences. This results in us naturally capturing the varying complexity of the world. In areas of high visual variation we store more experiences, while in regions that remain visually similar over time, we save relatively few experiences as our prior ones are sufficient for localisation. By allowing N to be greater than 1, the system is more robust to single localisation failures and it gives us more confidence in our position. This creates multiple experiences that encompass the same physical place and visual appearance, leading to common features across experiences. We choose not to merge, average or discard shared features as we want to explicitly avoid the difficult data association problem across experiences.

5.4 Connecting Experiences

One of the disadvantages of storing each experience in its own independent relative frame is that position look-up across experiences is more difficult. Consider the simplest case of two experiences, $\mathcal{E}^1$ and $\mathcal{E}^2$, starting at the same position in the world and following the same trajectory. For a given node $\mathcal{E}_m^1$, it is not possible to simply integrate the local transforms in $\mathcal{E}^2$ until the equivalent position is reached, as they are globally inaccurate. However it is important to link experiences so they can share position information.

Ideally we strive to store the minimum number of experiences needed to represent an environment, a surplus leads to wasted computation and memory resources. We want just the right number to facilitate long-term navigation². This motivates us to fully exploit all the information in each and every experience, and also how they inter-relate. Failure to do so reduces our ability to re-localise within an experience, leading to an unnecessary genesis of a new experience.

The opportunity to exploit experiences optimally occurs naturally in two forms. The first is when starting a localiser to run on an experience, as every experience is not always relevant to the current position in the world. Consider a vehicle travelling from a relatively unchanging area to one of high visual variance, it needs to know when to activate the extra experiences related to that area. The second is when an experience temporarily becomes insufficient to explain the current visual feed (the localiser fails). For example a recently parked high sided vehicle totally obscures the scene as we pass, or the robot is taking a short detour. The experience will be relevant a short while later, at which point it needs to be re-started.

As global position look-up is not possible, we look for other ways localisers can be started or re-started. One option would be to use an external loop closer such as FAB-MAP [21], where the live image is matched to a node in the saved experience. However it has a relatively low recall rate at high precision, so we may find ourselves “lost” for some time before it fires. Another approach would be to annotate experiences with metadata, such Global Positioning System (GPS) points, or higher-order descriptions such as known road junctions or buildings, however these are not always available and the GPS signal often suffers from significant drift over time. There are occasions when we do need these higher order loop closing techniques, and these are discussed in Section 5.4.2, but before we need to resort to these external mechanisms, we look to exploit topological links between experiences.

²We note that certain parts of the environment require more experiences than others, on account of greater visual variation.

5.4.1 The Experience Graph

As previously explained, an experience is a set of connected nodes. Consider if these were sub-graphs in a single large graph containing all experiences, G . Now assume we have a method of introducing an edge between two sub-graphs that indicates the two connected nodes observed the same physical place in the world. When a successful localiser arrives at a node with one of these edges, it can query the edge for which experience (and location within that experience) is at the other end. The localiser associated with the other experience can then be informed where to start (if the localiser is not currently active) or to restart (if the localiser is lost). Using this connected graph, localisers can inform each other when to start, and to aid each other when lost.

The focus here is on how the inter-experience edges of G can affect the performance of the localisation system. By introducing as many high quality edges as possible, we increase the information shared between experiences. These can then be used to aid localiser initialisation and restarting. The edges we introduce here do not contain any transform data, they mere indicate that the two nodes are at the same place. In the results section we present four methods for discovering the structure of G , i.e. how the node similarity is determined. The first approach uses a very sparse G to highlight its importance, the second is a method which only uses the live stream to provide links. The final two are introspective processes which run between vehicle outings. Their aim is to take any new experiences created from the previous outing, introduce the sub-graph formed from their nodes to G , and then look for opportunities to create edges between the newly inserted nodes and all other nodes (experiences). We will outline these processes in more detail later.

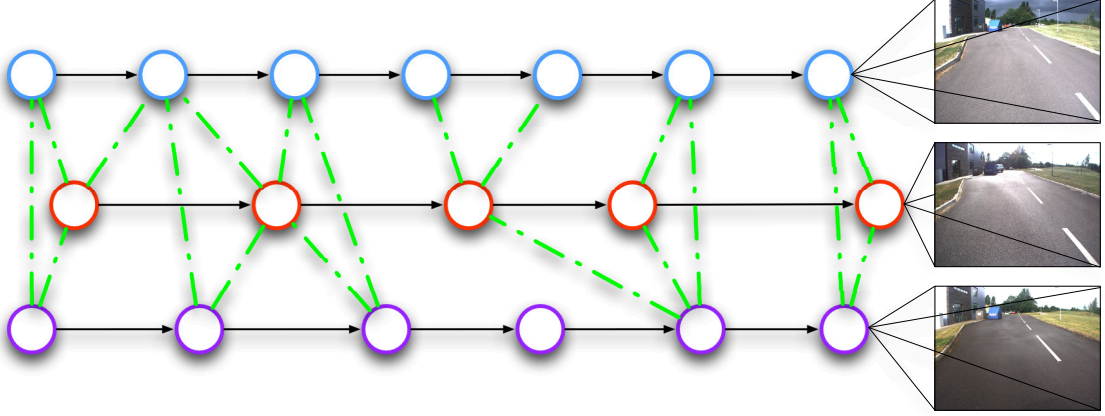


Figure 5.5: Each experience can be represented as a graph, where the stereo frames are represented as nodes (shown as circles) and metric transformation information describes how they are connected (indicated by the directed black arrows). The figure above shows three experiences, blue, red and purple. These can be formed into a single larger graph G , which contains all experiences. Edges can then be added between experiences (denoted as undirected dashed green lines) indicating that nodes refer to the same physical place in the world. This is demonstrated for the far right node in each experience. Each associated image is of the same place in the environment, so edges can be created between them. By increasing the quality and quantity of these edges we assess what impact this has on localisation performance.

5.4.2 External Loop Closing

Sometimes all of the localisation processes become lost, at which point the system does not know where it is in the plastic map. This may be because the current location has changed significantly (e.g. it snowed), or because the robot is exploring a new route. In either case the VO system will continue to process the live frame stream and will be saving the output to a new experience. However, it is possible the robot will subsequently return to a place it can successfully localise in and regain its position in the plastic map. For a solution to this problem, which should become less and less frequent over repeated traverses, we use an external loop closer which can reinitialise lost localisers.

5.5 Implementation

A key property of our system is that once the set of relevant localisers has been computed, localisation of the current live frame in each one is independent and so can be run in parallel. Given that the data association and trajectory estimation steps dominate the computation time, by parallelising them we are able to process frames at 15Hz. The localisers described in Section 5.3.3 use exactly the same processing pipeline discussed in the VO chapters, except the input landmarks are from an experience, not the live map.

The system needs a way to determine if a localiser is successfully localising, i.e. $\mathcal{Q}^i(\mathcal{F}_k) = 1$. For this we chose to compare the localisers translation from $\mathcal{F}_{k-1}$ to $\mathcal{F}_k$ with the same translation computed by the VO system running on the live stream, and require them to be in agreement to some percentage threshold. We investigate how this threshold affects the performance in the results section.

5.6 Results

We now present extensive results testing our proposed navigation framework. To achieve this we require many visits to the same environment - so we use the repeated loops of Begbroke Science Park introduced in Chapter 3. This dataset was collected with the Wildcat vehicle over a three month period and contains 53 loops. The vehicle was driven at different times of day and under different weather conditions. The route, with results, is shown in Figure 5.6. The outer loop, denoted by the thicker line, was driven on the first 47 traverses while the last 6 traverses went via the inner loop, indicated by the thinner line. For illustrative purposes we controlled the signal from the external loop closer so it only fired at 14 predefined points on each loop. The points were spaced approximately evenly along each loop.

5.6.1 Where were Experiences Saved?

Figure 5.6 indicates where we laid down experiences. The colour of the plot shows how many experiences were saved at each point. We find that some regions require more experiences than others. The northern and eastern (the image is orientated north up) regions of the route require relatively few experiences. Typified by open road flanked by buildings and low hedges, these areas are largely unaffected by lighting, weather and time of day changes. Example image sets from the northern and eastern road are given in Figure 5.7 and 5.8. We see here that despite been taken from many different days the scenes looks very similar - meaning localisation is relatively easy from a few experiences.

Conversely we see that some places require many experiences as they exhibit high visual variation. One example of such a place is the north west corner, where the camera observes a car park. The contents and configuration of this space varies daily meaning localisation against previous experiences is difficult. We show example

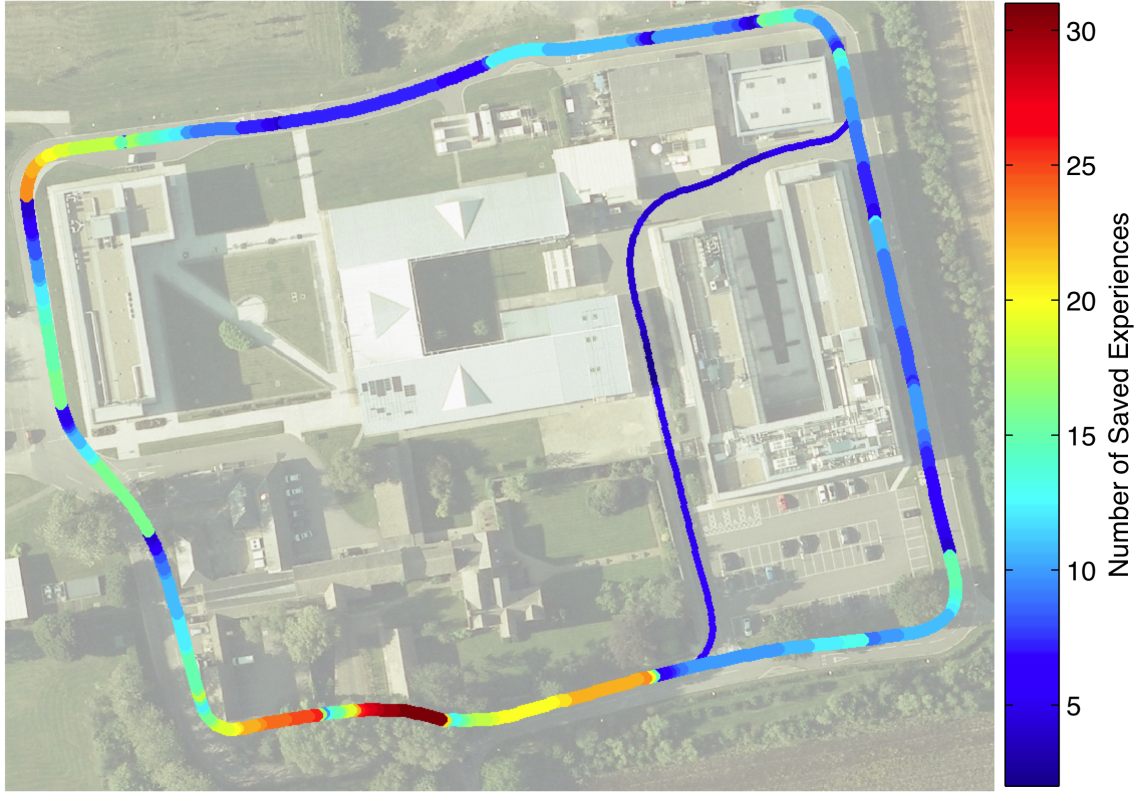


Figure 5.6: Overhead of the two routes visited during data collection. The outer loop, indicated by the thicker line was driven 47 times, while the inner loop, shown by the thinner line was driven 6 times. The colour of the trace shows how many experiences have been laid down at each point. See Figure 5.7, 5.8, 5.9 and 5.10 for examples of places with low and high experience density.

scenes from this place in Figure 5.9. A second example is the south west corner. Here the vehicle is travelling in a section covered by overhanging trees. When the sun is out, it casts strong, intricate and ephemeral shadows. While these are useful for the live VO system, they are often not encountered again meaning previous experiences are not useful. This encourages the use of the live VO system at all times. A set of example images from this region is shown in Figure 5.10.



Figure 5.7: Images taken from the northern road of the Begbroke Science Park across several months. We find this region has high visual stability and as a result reliable localisation can be achieved from relatively few experiences of the area.



Figure 5.8: Images taken from the eastern road of the Begbroke Science Park across several months. Again due to the unchanging nature of the scene it can be modelled with relatively few experiences.



Figure 5.9: Sample images from the north west corner of the Begbroke Science Park. This area generates several experiences due to the car park. The vehicles in the area change daily meaning localisation against previous experiences is difficult.



Figure 5.10: Sample images from the south west corner of the Begbroke Science Park. The section of road is covered by overhanging trees which, when the sun is out, generates ever changing shadow effects. These are useful for the live VO system as they provide strong and reliable features, but due to their ephemeral nature are not useful for localisation on future visits. This results in a large number of experiences being generated for this area.

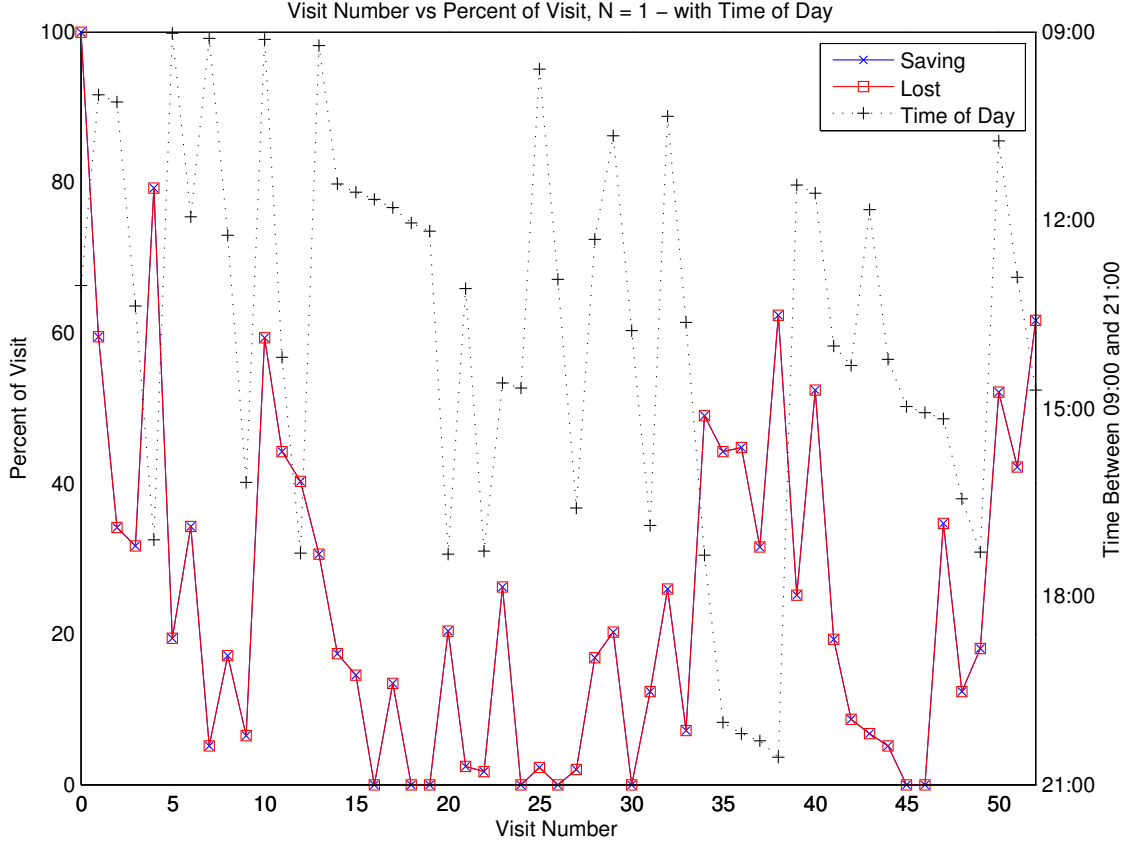


Figure 5.11: System performance on the repeated loops of the Begbroke Science Park. Visit number is displayed on the x-axis and normalised visit percentage on the y-axis. We plot the amount of time the system spent saving (blue) and lost (red) - as $N = 1$ here these values are equal. We also plot the time of day each dataset was collected (black crosses).

5.6.2 When were Experiences Saved?

Figure 5.11 shows the results of the system for $N = 1$ on the Begbroke Science Park loops dataset. N is the minimum number of successful localisers permitted for the system not to save the live VO as a new experience. For example, if $N = 2$ and the system is successfully localising in one prior experience, the current VO output will be saved as a new experience. In Figure 5.11 the x-axis indicates the visit number and the y-axis is the normalised visit distance. We plot both the amount of time spent lost, and the amount of time spent saving experiences. For $N = 1$ these values are the same, but later when $N > 1$ these values will differ. We also plot

the time of day each dataset was collected. Initially everything is new, so on the first visit everything is saved. On the second visit, approximately 60% of the time was spent lost - so the associated VO output during this time was saved, and so on. The first full traverse without any localisation failure occurs on the 16th visit - the normalised percent of the visit spent saving or lost goes to zero. The large jump around traverses 35-38 happens because for the first time we collected data as dusk fell. The roads also had standing pools of water and it was raining lightly - something the system had not encountered before. The second spike at visit 47 is caused by driving the inner loop for the first time. Suddenly no localisations are successful and the whole section is saved until the loop closer fires. We note that for the last 3 runs the system gets more lost, this is a combination of continuing to learn the new inner loop (which in parts is difficult due to distant scene structure) and getting lost at other points around the loop. Figure 5.13 shows examples of localisation failures on visit 4, where strong shadowing effects are encountered for the first time. Figure 5.14 shows examples of localisation failures on visit 38 when driving at dusk on wet roads. However apart from these cases where we explicitly did something out of the ordinary to test the system (like driving at dusk or a different route), we find that as we revisit the route we typically need to remember less and less each time. We will show later that the shape of this graph can be improved. Weather statistics for each visit are given in Figure 5.12.

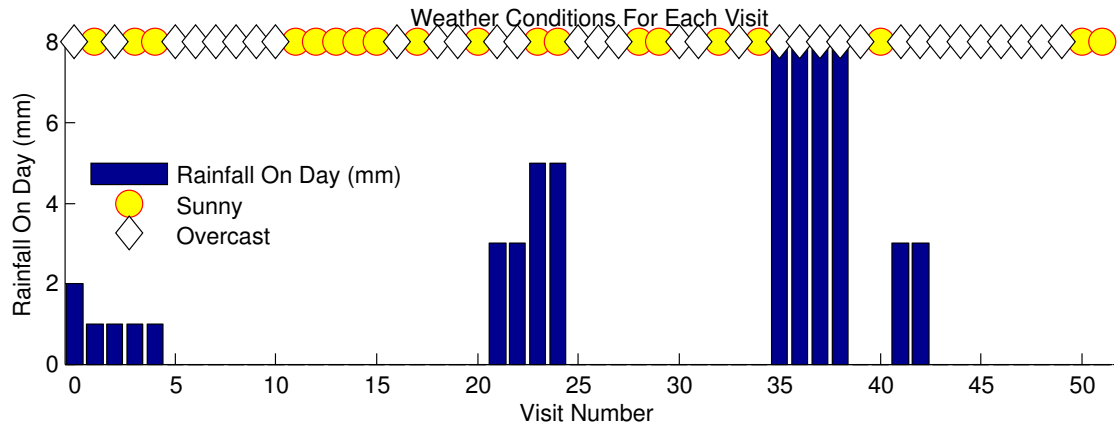


Figure 5.12: Weather statistics for each traverse [107].



(a) Prior



(b) Live



(c) Prior



(d) Live

Figure 5.13: Examples of localisation failure leading to the creation of new saved experiences on visit 4. Frames (a) and (c) are from previously saved experiences and frames (b) and (d) are from the live stream.



(a) Prior



(b) Live



(c) Prior



(d) Live

Figure 5.14: Examples of localisation failure leading to the creation of new saved experiences on visit 38. Frames (a) and (c) are from previously saved experiences and frames (b) and (d) are from the live stream.

5.6.3 Choices of N

By varying N we can influence how many similar representations of the environment the system needs. While this does lead to multiple representations of the same place, it also offers increased robustness. Should localisation fail in one experience for some reason there are still other experiences that can be used. Note that even though these experiences will share some common measurements we make no attempt to fuse, merge or average them as we want to explicitly avoid the difficult data association and fusion problems that would arise. We show plots of the system performance for $N = 2$ and $N = 3$. Note that now the saving (blue line) and lost (red line) are now different as it is possible to be localised but still saving. For example, consider the case when $N = 2$. If the system is only localised in one prior experience, the VO output will be saved as the number of successful localisers is less than N , however the system will not be lost. We plot the saving and lost results for $N = 2$ and $N = 3$ in Figure 5.15 and Figure 5.16. We also tabulate the results in Table 5.1.

We see that for $N = 1$, the system saves the least, but also spends the most time lost. Moving to $N = 2$ brings a significant increase in localisation performance - the time spent lost is reduced by nearly 10%. This comes for less than 5% increase in the amount of experiences saved. This boost is due to the increased robustness and redundancy introduced by having more than one experience available. The system is not so brittle to single localisation failures. We see further increases in performance for $N = 3$. Also note for increasing N we see more visits which were traversed with no localisation failures - the amount of time spent lost goes to zero more often.

The prediction for $N > 3$ is that the system would obviously save more, but also become increasingly robust to localisation failures. For increasing N we would expect the lost results to tend to a constant. After a point increasing N will bring diminishing returns, as further saved data will not bring any more robustness. Remember, in the face of completely new scenes (caused either by appearance or location change)

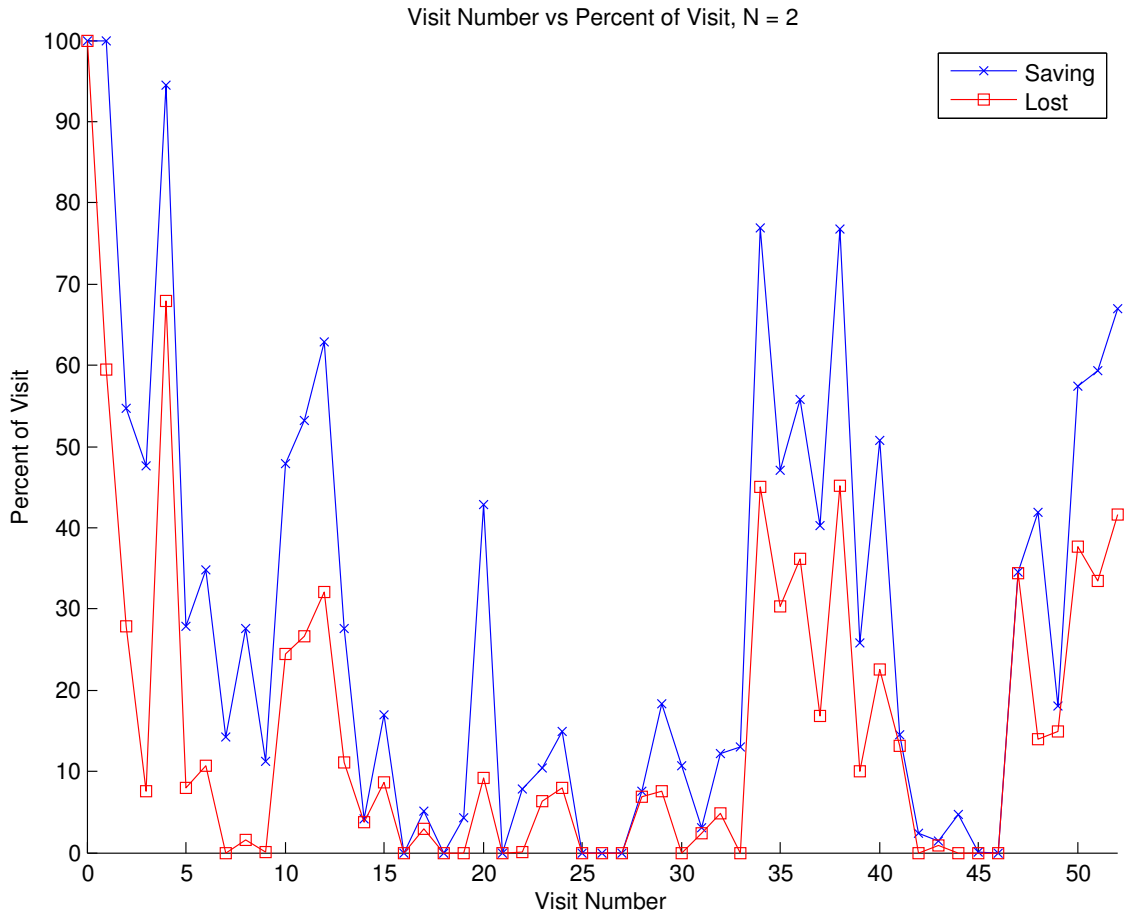


Figure 5.15: Performance results for $N = 2$. The system saved 29.25% of all the possible experiences and was lost 15.73% of the time.

the system would have no advantage when compared to those with lower N values, as there simply isn't the data in memory to localise against.

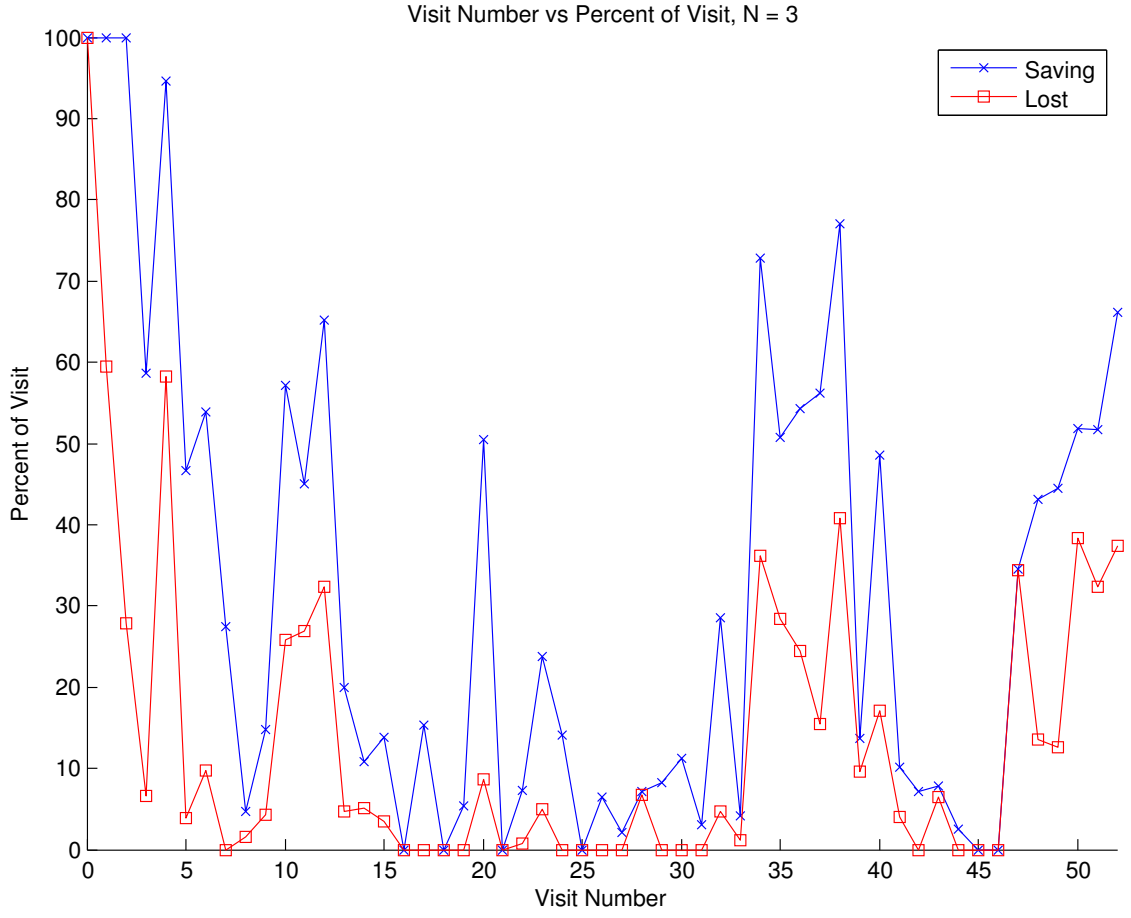


Figure 5.16: Performance results for $N = 3$. The system saved 31.95% of all the possible experiences and was lost 14.12% of the time.

Table 5.1: Experience Based Navigation Results on the Begbroke Loops for different choices of N . The percentages is of the total possible number of experiences - e.g. 100% saving would mean the system always saved the VO output. 50% lost means the system was lost half the time. Lower is better in both cases, and localisation is the one we really care about (not being lost). Here we see for modest increase in saving we get good boosts to localisation.

	Percent Saved	Percent Lost
$N = 1$	24.80%	24.80%
$N = 2$	29.25%	15.73%
$N = 3$	31.95%	14.12%

5.6.4 Are Multiple Experiences Needed?

To demonstrate the value of saving multiple experiences and building the plastic map, we evaluated the performance of the system if we only made one previous experience available for localisation. This represents the model most traditional localisation methods use, where a single previous view of the world is used and assumed correct - often this is the first time the world is experienced. We took each visit in turn and declared it the only available prior experience of the world. Using this single experience, we attempted to localise against it for all other visits. For each run we recorded the amount of time spent lost - i.e. localisation had failed. Results are shown in Figure 5.17. We plot the median performance for each visit and mark the 25th and 75th percentile bounds. Interestingly no single visit is particularly “good” as a prior for most other visits. Also we see the visits conducted around dusk (35-38) are also less useful as they share less similarity to the majority of the dataset. Across all visits the average time spent lost is 65.8%. This result suggests that reliable localisation results from a single prior will be difficult to achieve and motivates the use of multiple experiences working in concert to represent the environment.

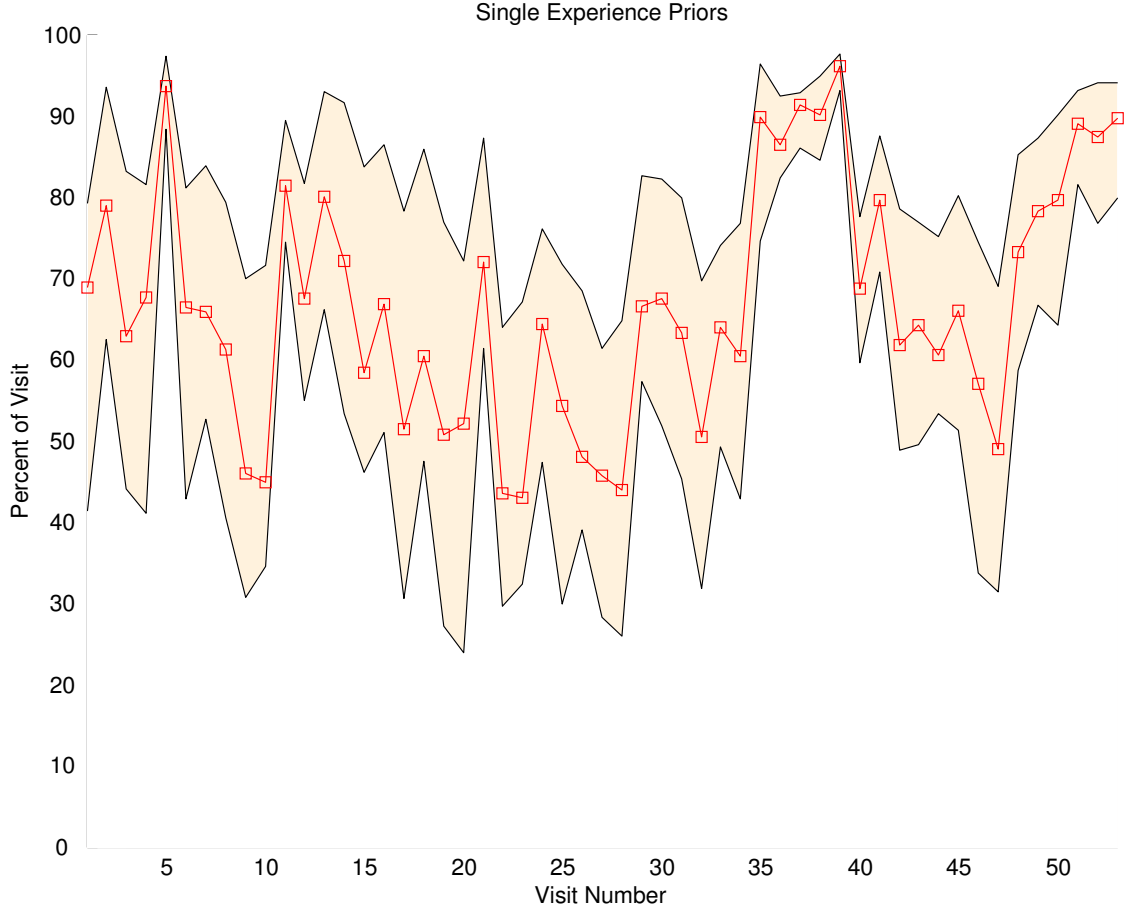


Figure 5.17: This graph shows the performance of single experience priors. For each visit we used that as the single experience available. Localisation was then performed on all the other visits using just the single available prior and the time spent lost - i.e. localisation had failed - was recorded. Here we plot the median performance of each visit when used as the single prior experience along with the 25th and 75th percentile bounds. We note that no visit performs well as a prior with the average time spent lost being 65.8%. This motivates the use of multiple experiences.

5.6.5 Long-Term Trends

Over time, barring new routes, our approach produces an inversely proportional decay as the system captures the typical variation of the route. The order in which the data were collected is a privileged sequence. Once we saw a signal of the system working we intentionally collected data we knew would cause problems to see how it would respond - such as driving at night or new routes. However the data could be ordered in $53!$ ways to produce different performance graphs. To make this point and demonstrate we can achieve an inversely proportional decay we performed a greedy re-ordering. We moved the 10 most surprising traverses of the outer route and the 6 inner traverses to the beginning and re-ran the system. We measured surprisal by the amount of traversal that was saved in the original configuration.

The resulting performance graphs for $N = 1$ and $N = 3$ are shown in Figure 5.20. By moving the most “interesting” and “surprising” visits to the beginning of the plastic map creation we see that accumulation of experiences is high before dropping off significantly. For $N = 1$ the system is still getting lost occasionally - this is due to the fragility of single localisation failures. When increasing N to 3 we see an improved robustness with many visits achieving no localisation failures.

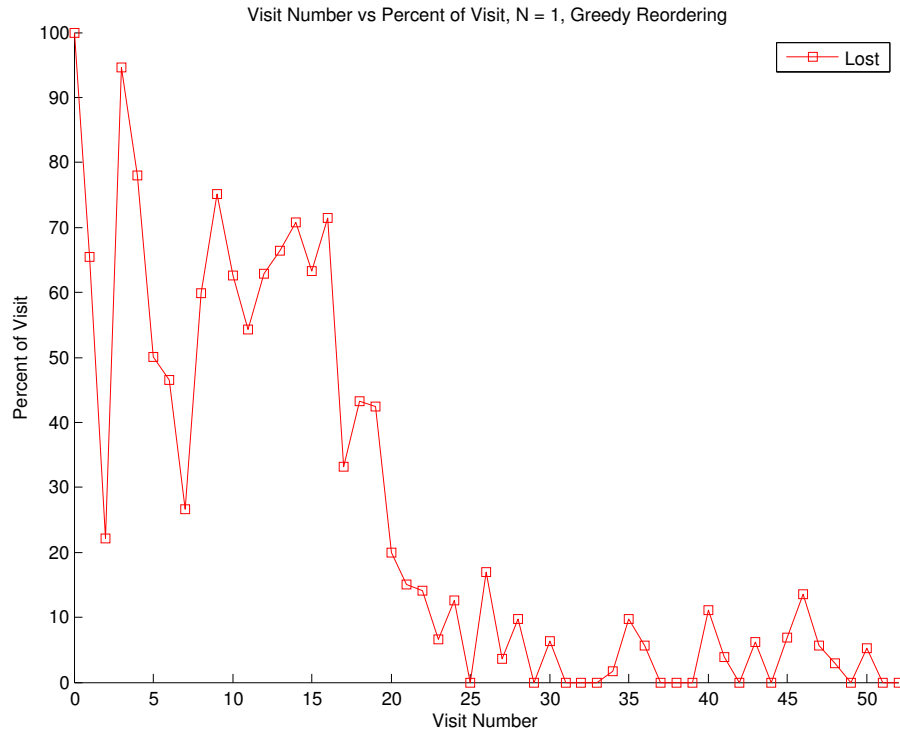
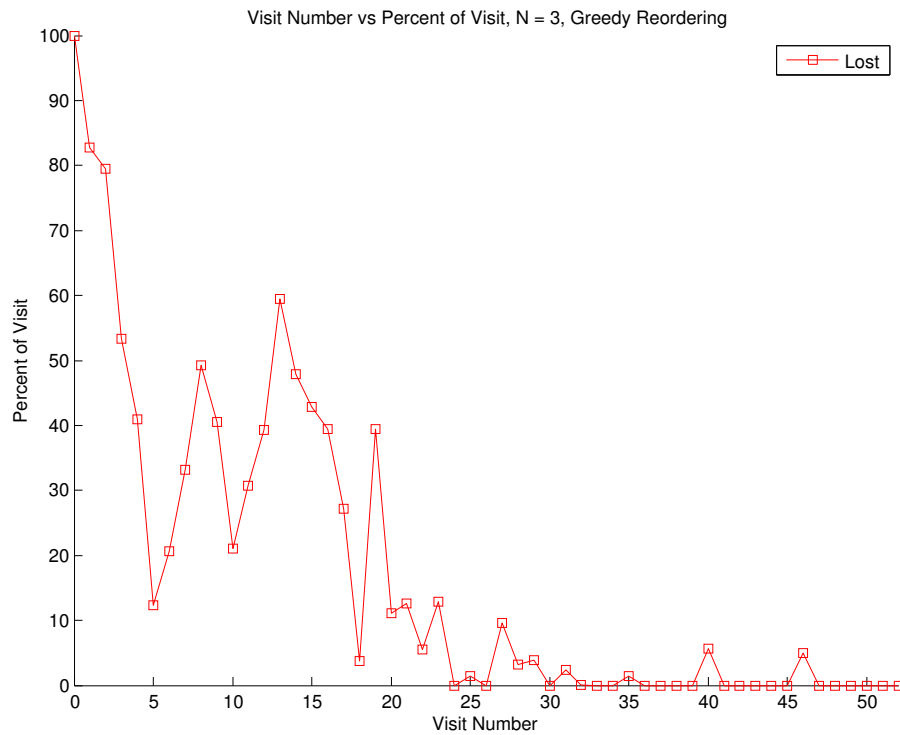
(a) Greedy reordering - $N = 1$ (b) Greedy reordering - $N = 3$

Figure 5.18: The system performance when the traverse order has been greedily re-arranged to present the system with the most surprising and diverse set of experiences first.

5.6.6 Descriptor Choice

In Chapter 4 we saw that varying the descriptor length of BRIEF did not overly effect the results. This was due to the fact that sequential images from the camera are largely similar in appearance - the camera is unlikely do have undergone large translations or rotations between captures. When attempting to navigate against a prior experience, this assumption no longer holds true. To this end we investigate the performance results for different choices of BRIEF descriptor - 16, 32 and 64. The longer the descriptor the more discriminative it is.

We present a plot of both the saving and lost performance of our system when using the different BRIEF lengths, for varying values of N in Figure 5.19. We see that BRIEF-16 is significantly worse than BRIEF-32 and BRIEF-64 due to its shorter and less discriminative nature. It ends up saving more experiences and is localised for less time. Overall BRIEF-64 is best, it always saves the least, and apart from $N = 2$ is lost less than BRIEF-32. We also show the system performance for $N = 3$ with BRIEF-16 and BRIEF-64 in Figure 5.20. Clearly the BRIEF-64 graph is more suppressed (better) with several traverses achieving no localisation failures.

These results make sense. The data association problem when attempting to localise in an experience is more difficult due to changing environments, larger and more persistent spatial deviations, and having to find matches in a larger pool of candidates. In this setting the more discriminative descriptors perform better.

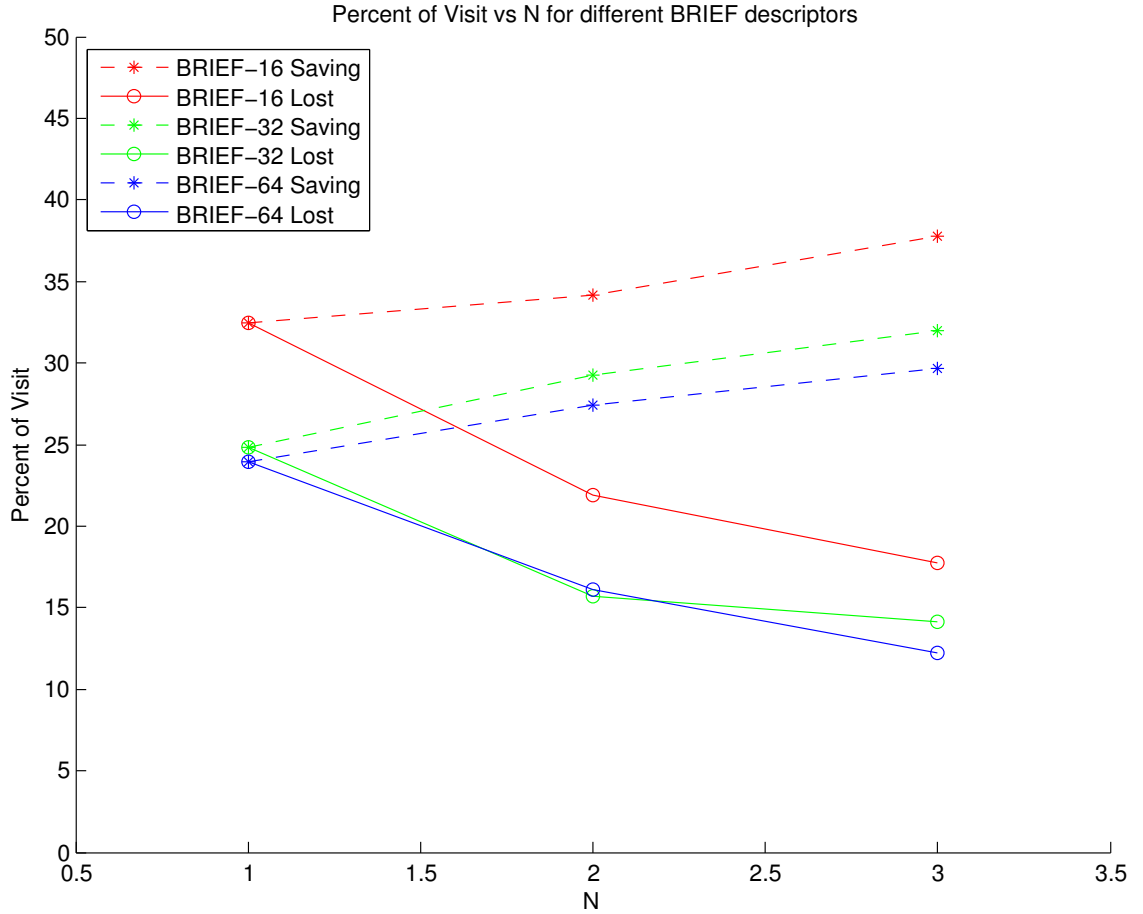
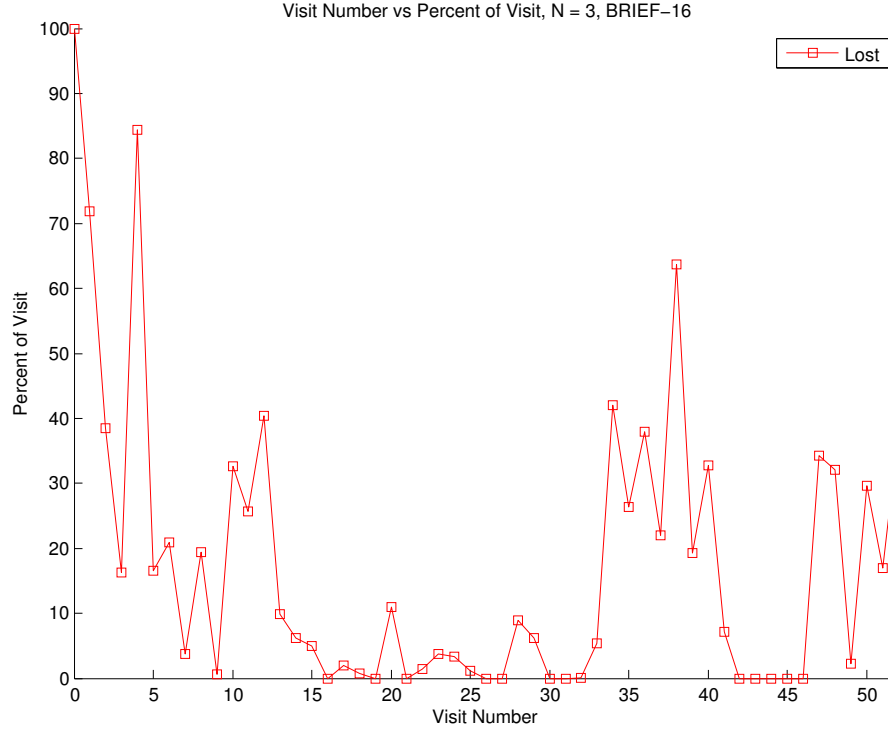
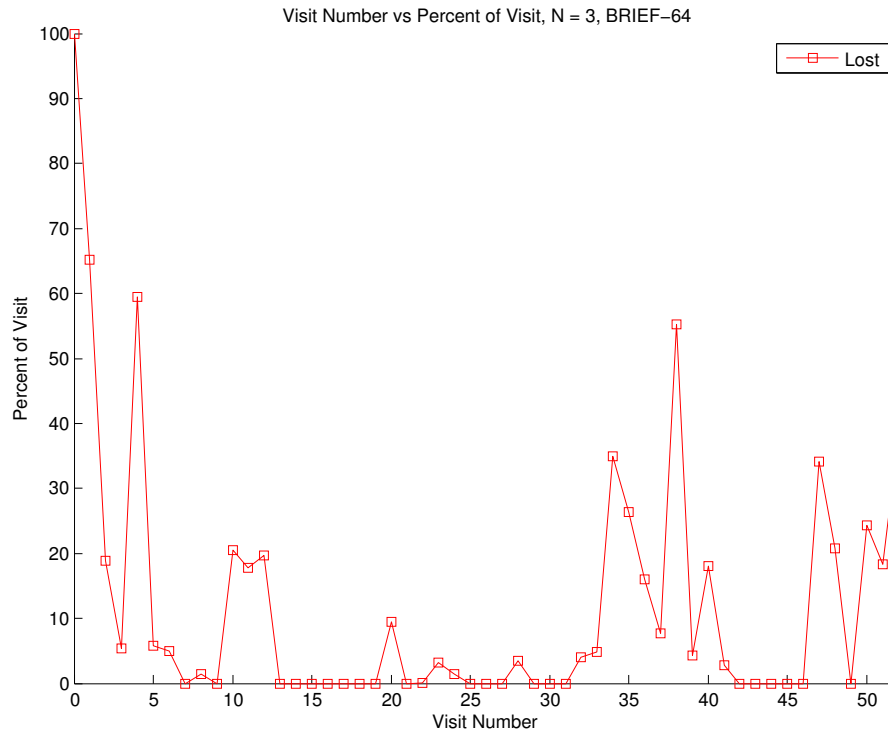


Figure 5.19: Here we plot both the amount of time spent saving and lost for different values of N , when using different BRIEF descriptor lengths. The least discriminative is BRIEF-16 (red) which spends the most time both saving and lost. BRIEF-32 (green) and BRIEF-64 (blue) - the more discriminative descriptors - perform better, with BRIEF-64 achieving the best results.



(a) BRIEF-16 Lost Results



(b) BRIEF-64 Lost Results

Figure 5.20: Performance results for two different BRIEF descriptors - 16 and 64. Unlike the frame to frame visual odometry system, here we see increased performance when using more discriminative descriptors.

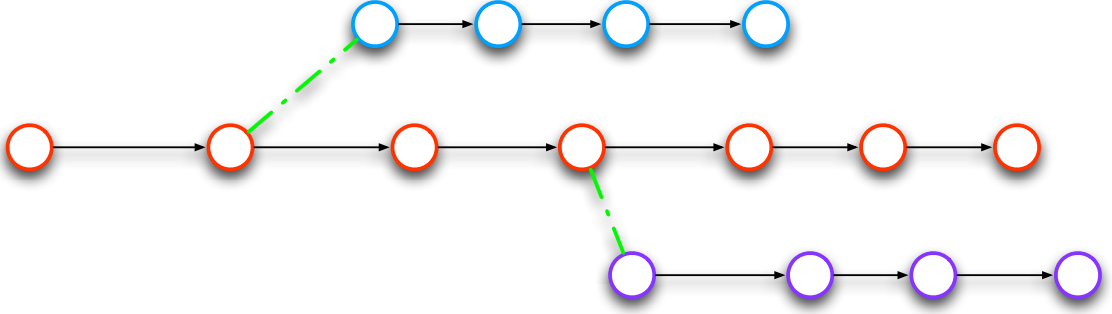


Figure 5.21: An example graph constructed with the No Discovery setting. Here there blue and purple experiences are only linked to other experiences when they are created, otherwise they share no links.

5.6.7 Connectivity of the Experience Graph

To evaluate how the connectivity of G affects the localisation performance we describe four approaches to discovering these connections and test each. The four approaches are No Discovery, Live Discovery, GPS Discovery and Refined Discovery. The first two approaches do not require any extra computation. The final two are introspective processes which are run between vehicle sorties. We now describe each in more detail.

No Discovery

To demonstrate the importance of G 's connectivity levels, in this approach we only allow inter-experience edges to be created which include the start of an experience. This enables experiences to be started if they are not active (otherwise they would never be used), but prevents the re-initialisation of lost experiences. An example graph showing this method is shown in Figure 5.21.

Live Discovery

Here edges are created when the live stereo stream is localised in more than one experience. The resulting links are likely to be accurate due to the strong geometry

required for successful localisation. However by only using the live stream, G can only be changed when the vehicle is driving, and only in the regions covered by the vehicle on that trip. For example if the previous sortie included the car park, but it is not included on the next outing, the car park section cannot be connected to other experiences of that area. This dependancy on the live system to create edges is undesirable.

GPS Discovery

In this variant, for every experience created we also store the GPS position of every node. When an experience is created, for each new node, we find the nearest position in all other experiences via GPS. Given a new node, for every other experience, if the nearest node is close according to GPS, we introduce an edge connecting them.

Refined Discovery

In this final version we attempt to recover from the potential variations that GPS can suffer from when recorded over long periods of time. As our experiences are collected over several months, our data are affected by the potentially wandering nature of the GPS signal. We achieve an initial match to other experience nodes using the same approach as GPS Discovery. We then look to refine this estimate by matching the new node against a sequence of nodes surrounding the GPS suggestion, based on the same stereo frame to frame estimation techniques used in the VO system.

Given the new node n^* and a candidate node proposed from GPS, n_c , we take a small window of nodes either side of n_c , denoted $\{n_c\}_w$ and compute the transform from n^* to each node in the window. Next, assuming all transforms are valid, we compute the translation from n^* to each $\{n_c\}_w$. If we find a local minimum which is not on the bounds, i.e. n^* really did pass by the window $\{n_c\}_w$, we take the candidate node in $\{n_c\}_w$ with the smallest translation to n^* to be the same place.

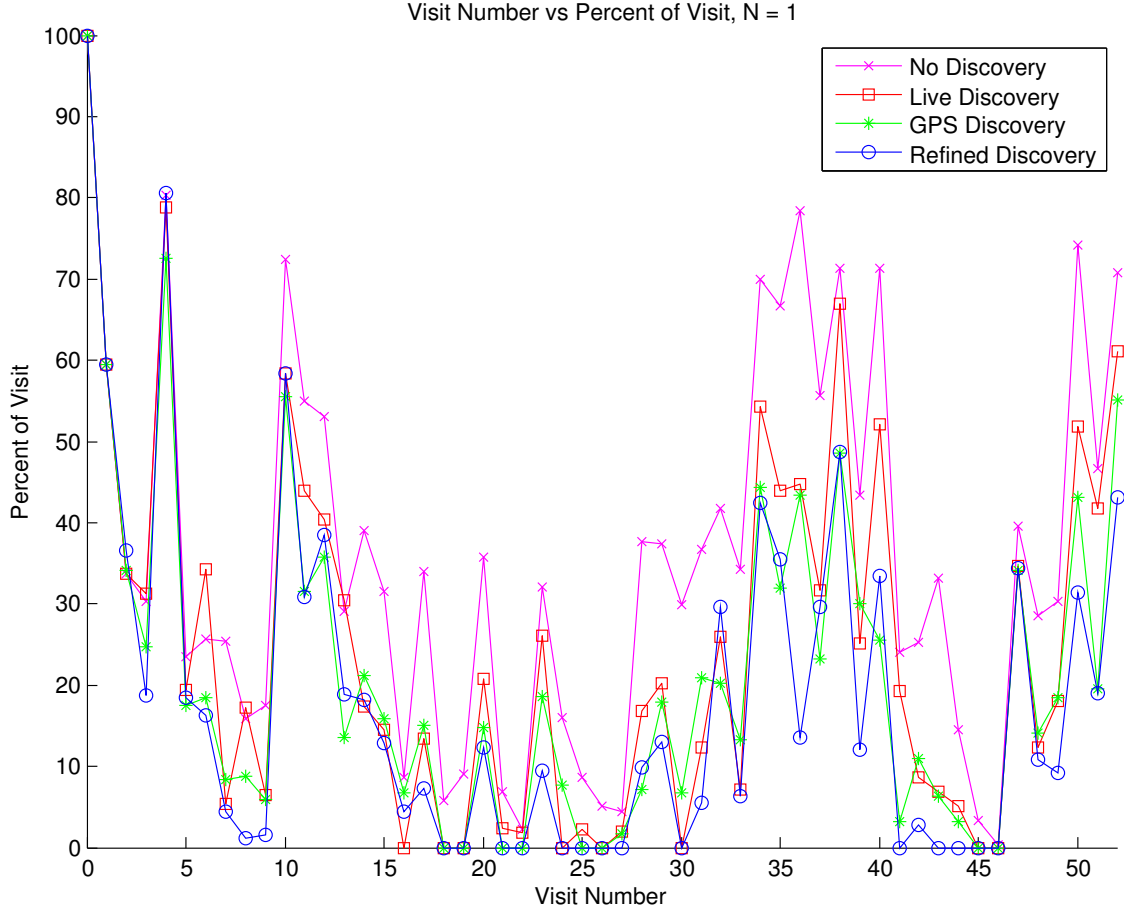


Figure 5.22: System performance when using different graph structure discovery approaches. $N = 1$ for all runs.

The appropriate edge is then created in G .

Impact of Discovery Choice

We present results for $N = 1$ and 3 in Figure 5.22 and Figure 5.23. The saved and lost statistics for each variant, along with varying N is shown in Table 5.2. Performing Refined Discovery results in a 26%, 17% and 10% improvement when N is 1, 2 and 3 respectively, when compared to Live Discovery. This decrease in improvement makes sense. When operating at $N = 3$, the Live Discovery approach is able to create a greater number of links, reducing the number of new ones that the introspective approaches can introduce. Interestingly GPS Discovery does not

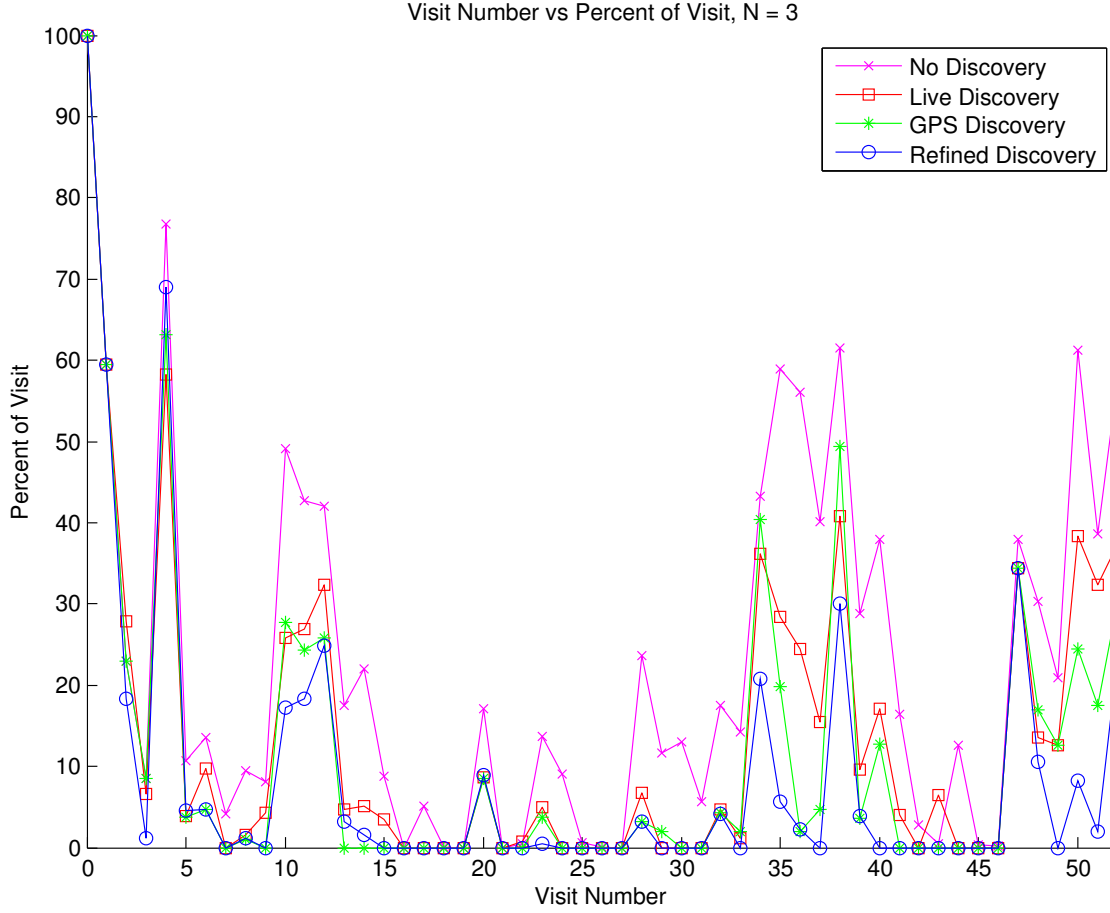


Figure 5.23: System performance when using different graph structure discovery approaches. $N = 3$ for all runs.

perform as well as Refined Discovery. This is likely to be caused by the quality of the inter-experience edges in G being substandard due to the slight inconsistent nature of the GPS signal.

Note how in both Figure 5.22 and Figure 5.23, GPS and Refined Discovery do not always out perform the original Live Discovery variant. This is because performance on a particular run is directly tied to what has been saved previously. As the Live Discovery stores more in earlier runs, on some later outings it has more experiences to draw from and sometimes stays localised for longer than the other

Table 5.2: Percentage of saved output (lower is better). GPS and Refined Discovery methods work best.

	Discovery Type			
	Saving			
	No Discovery	Live	GPS	Refined
$N = 1$	36.33%	24.80%	21.32%	18.47%
$N = 2$	46.07%	29.25%	27.19%	24.13%
$N = 3$	50.53%	31.95%	30.32%	28.51%
	Lost			
	No Discovery	Live	GPS	Refined
$N = 1$	36.33%	24.80%	21.32%	18.47%
$N = 2$	26.55%	15.73%	12.83%	12.01%
$N = 3$	23.28%	14.12%	11.96%	9.09%

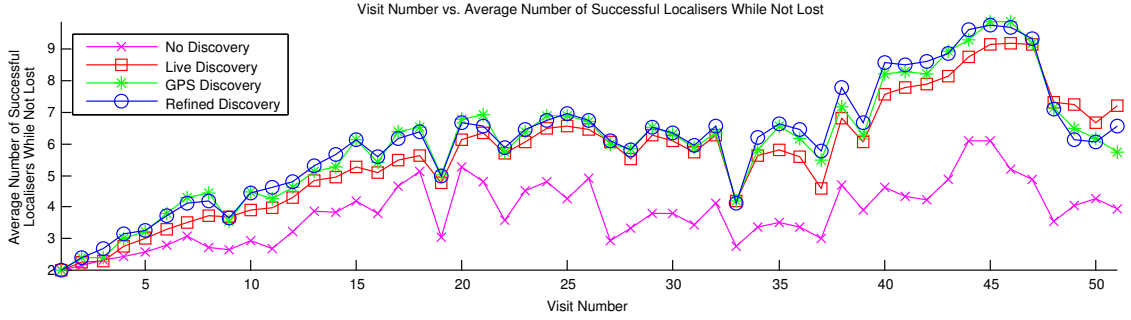


Figure 5.24: For each run, the average number of successful localisers while the system is not lost is shown for each system variant. Note that GPS and Refined Discovery, both variants that spend time after a run to perform better matching of newly saved experiences have a higher number of successful localisers. Given both of these variants save less, it shows they are leveraging their experiences better. Results shown for $N = 2$.

systems. However Table 5.2 shows that Refined Discovery performs best overall, followed by GPS Discovery and then Live Discovery. No Discovery always performs worst and saves significantly more experiences than the other variants. This of course is expected as the system becomes brittle to localisation failures and is wholly dependent on the external loop closer to put it back on the right path.

In Figure 5.24 we show, for each visit, the average number of successful localisers while the system is not lost. We see that both variants that process newly saved experiences between outings generally record a higher number of successful localisers for each run. Given they also store less experiences, this implies they are making

much better use of the information they already have stored. By increasing the quantity and quality of edges in G , we have shown that we can stay localised for longer and are required to save less, as we are making better use of the information we currently have.

Table 5.3: Sensitivity to Lost Threshold, $N = 2$.

	Motion Similarity Threshold			
	5%	10%	15%	20%
Saving	38.11%	31.71%	29.25%	27.62%
Lost	25.53%	19.76%	15.73%	15.08%

5.6.8 Sensitivity to Lost Threshold

The experience based navigation approach requires some metric to measure if a localiser is lost. We chose to compare the localiser’s motion estimate with that of the VO. Given this, we must then decide on some threshold at which the two disagree. To investigate our systems sensitivity to this parameter we ran it with thresholds requiring 5%, 10%, 15% and 20% dissimilarity. The saving and lost percentages for $N = 2$ is given in Table 5.3. We also plot the saving and lost graphs in Figure 5.25.

We see that for more stringent (lower) thresholds the general performance degrades. This is because the system is demanding that the inter-frame motion estimates of the localiser and the VO agree more closely. In reality this is difficult to achieve as we are trying to match image features to different sets of landmarks, each of which will have their own errors. We have seen in the previous chapter how the average inter-frame error of the VO system is around 5% - and that is attempting to just match two adjacent frames. For thresholds between 10% and 20% we see the difference in performances are less. We also note that the general shape of the graphs in Figure 5.25 are the same, with the stricter 5% threshold generally being slightly higher than the rest. This suggests that while the choice of this threshold will impact the performance of the system, it should not affect the overall functionality provided it is reasonable, i.e. the sensitivity threshold is not smaller than nor significantly larger than the typical error found in the VO system.

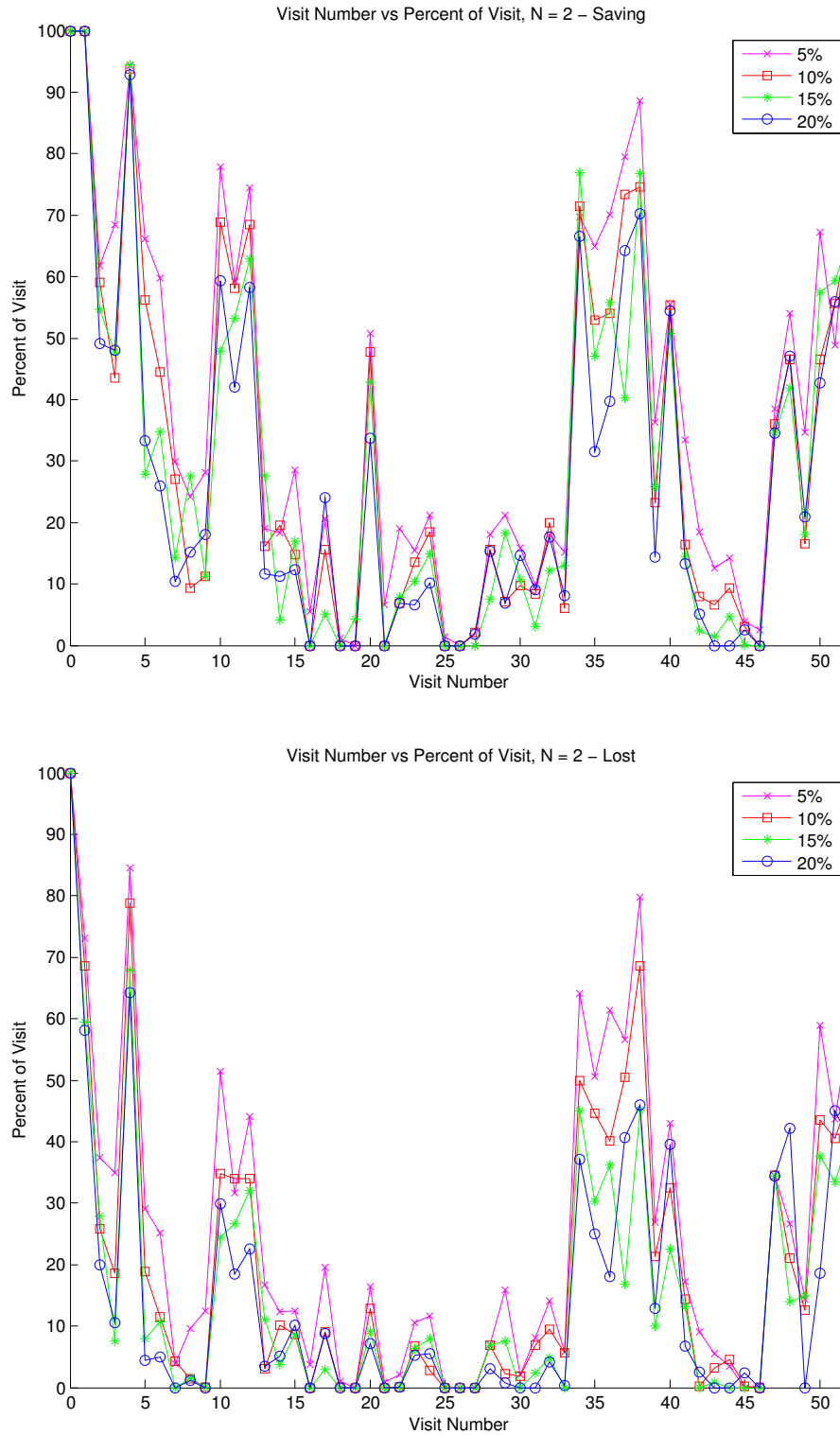


Figure 5.25: Here we plot the saving (upper) and lost (lower) performance graphs for varying lost thresholds. We see that the general shape of the graphs are largely similar.

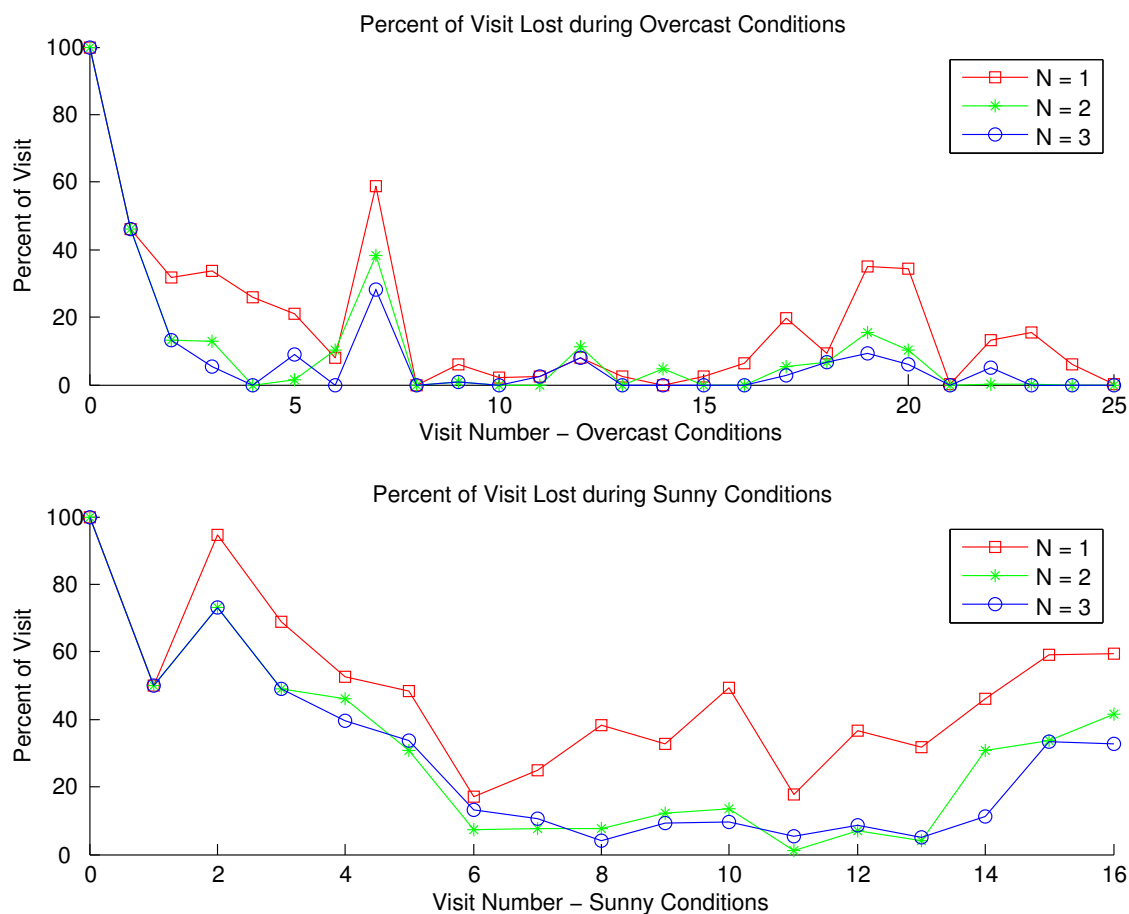


Figure 5.26: Performances of only overcast (top) vs. only sunny (bottom) traverses. Notice the constant offset for sunny conditions compared to overcast conditions - a whole visit with no localisation failures is not completed in sunny conditions. This is probably caused by strong shadows making localisation difficult in previous experiences.

5.6.9 Weather Conditions

We also classified each traverse as either overcast or sunny (the weather for each visit is shown in Figure 5.12). We ran the system using only overcast or sunny experiences, the results of which are shown in Figure 5.26. (We removed the 4 dusk and 6 inner loop traverses.) Interestingly, when provided with only overcast visits, the system quickly accumulates sufficient experiences to navigate successfully. Conversely when presented with the purely sunny traverses there is a slight constant offset - the system never quite completes the route without getting lost at some point.

We believe this is happening because the direct sunlight casts different shadowing effects which make localisation against previous experiences difficult.

5.6.10 Timing Performance

Finally we show the performance of the system running on the Wildcat hardware in Figure 5.27. The Wildcat houses 2 Intel Xeon X5570 2.93GHz CPUs, offering 16 cores (with hyper-threading). Shown is the number of successful localisers and timing performance for each frame on visit 47, which is the first traverse of the inner loop. Localisation is successful until frame 1296, at which point the vehicle turns onto the inner loop. When this happens a new experience begins as localisation fails in all previous experiences. At frame 2239 the external loop closer fires and results in successful localisation, so saving of the new experience stops. Despite varying numbers of active localisers the timing per frame typically stays under 100ms, while the average for the successful localisation part of the sequence (i.e. not including frames 1296-2239) is 53ms. This is possible due to the ability to parallelise the localisation processing across the CPU cores.

5.7 Discussions

5.7.1 Sensor Choice

As noted previously, our approach has been implemented using a stereo camera, but would also be applicable to other sensor modalities, assuming that motion estimation and localisation modules are available. One obvious choice would be lidar sensors. Choosing a different sensor may provide more robustness to certain types of scene change. For example lidar sensors will clearly perform better than vision sensors when scene change occurs from varying lighting effects. This would allow it to

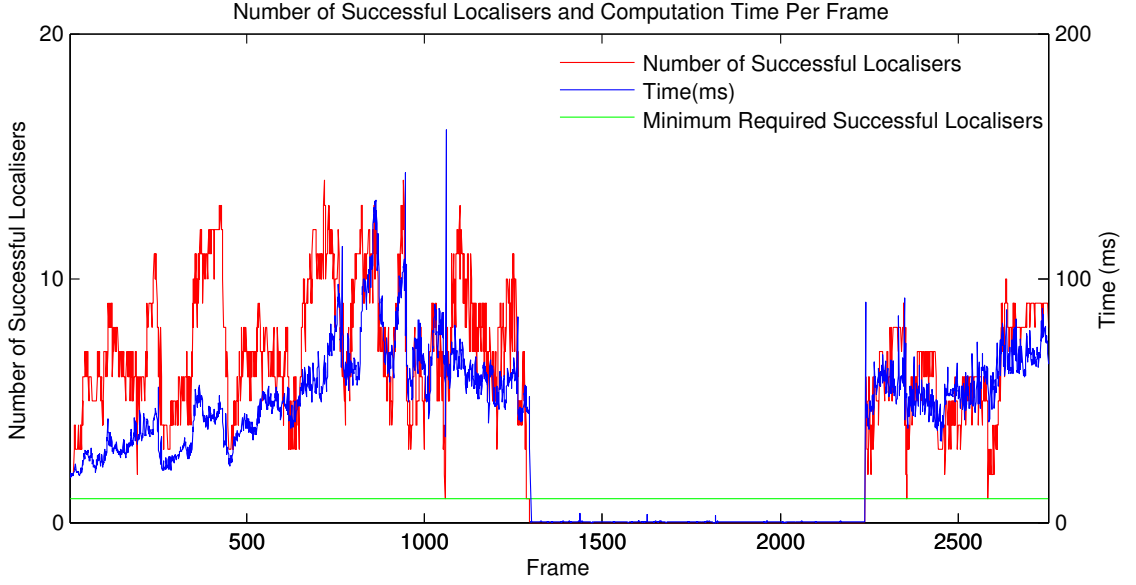


Figure 5.27: Timing performance (using the Wildcat hardware) and number of successful localisers for each frame on visit 47, the first traverse of the inner loop. Only when the robot is driving on the previously unseen road is the VO saved, otherwise localisation is always successful. In the regions of previously seen road the average number of successful localisers is 7, and the average localisation time per frame during this period is 53ms.

deal better with the sunny traverses that the vision based system struggled with in Figure 5.26. This change in sensor modality is exactly the approach McManus *et al.* [69] took to extend their original teach and repeat system [36]. By using a lidar based system they were able to reduce the impact of lighting on the performance of the system. However lidar does not completely solve this issue. McManus *et al.* notes dips in performance caused by the high noon sun, and rain soaked ground producing different reflectance values compared to when dry. Others have also noticed lidar systems suffer when surfaces become wet or icy [6]. Further, such a system would still need to lay down multiple experiences in regions which undergo gross structural change, such as the car park shown in Figure 5.9. It should also be noted the lidar sensor used by McManus *et al.* can only operate at 2Hz and due to its scanning nature, if used while the robot is moving produces warped images.

5.7.2 Forgetting Experiences

We find for the majority of the route used in our experiments, the number of experiences saved tends to a constant, as the low number of visual modes are quickly discovered. Significant sections of Figure 5.6 are captured with just a handful of experiences. However there are notable areas that have highly unstable appearance, resulting in significant numbers of experiences being saved in these regions. The prime example being shown in Figure 5.10, overhanging trees casting a unique set of shadows each time the sun is shining. After 53 runs, over 30 experiences have been saved in the most difficult region. Another troublesome region previously discussed is when passing the car park shown in Figure 5.9. To prevent new experiences being created on almost every visit to these areas, these regions could be marked as unmappable after a certain number of attempts. When entering this region in future, the system could switch over to purely relying on the VO module, which is very accurate over shorter distances, and the external loop closure could be used when returning from longer problematic sections.

In this work we have not considered deletion policies for experiences. The policies covered in the previous work that focus on maintaining diversity and recency would be obvious choices if it was required to keep computation costs down. Another approach would be to select only *appropriate* experiences. If we know we are navigating in sunny conditions in the middle of summer, there is little point attempting to localise in an experience captured on a snowy winter morning. By judiciously selecting experiences, computational savings could easily be made.

5.8 Conclusions

In this chapter we have demonstrated continuous localisation of a road vehicle in drastically changing lighting and weather conditions over a 3 month period. This

was possible because we adopted the notion of plastic mapping. We focussed not on building a single monolithic map or inferring a latent underlying state which explains all observations of the workspace but on creating a composite representation constructed from multiple overlapping experiences. This representation is only as complex and rich as it needs to be. It handles both drastic and creeping changes in the same way - as soon as prior experiences fail to adequately describe the present a new experience is saved for future reference. We have shown our system working in real time embedded on a vehicle. We have shown the advantages of plastic maps in localisation performance (robustness) and have demonstrated the asymptotic behaviour of plastic map maintenance. Starting with a core competency (in our case VO) day on day, week on week, we are extending our vehicle's operating envelope; gradually making the extraordinary ordinary.

Chapter 6

Conclusions

This chapter gives a summary of the work described in this thesis and discusses some possible avenues for future research.

6.1 Summary of Contributions

We began with the idea that for robots to truly achieve lifelong autonomy they require the ability to accurately navigate in changing workspaces. To this end we first presented our stereo visual odometry implementation and provided a detailed analysis of several components. We then introduced a new approach to the problem of long-term navigation.

In Chapter 2 we gave a general overview on the field of stereo visual odometry. The complete visual odometry system is the union of many components. We began by covering prior complete systems and noted key choices previous researchers have made to improve the quality of their implementations. We then described in more detail each block in the overall system pipeline, covering the related background work at each stage. By combining these components we were able to build a highly accurate ego-motion estimation system. One key insight was the use of the recently

published BRIEF descriptor which combines the speed of simpler patched-based methods with the matching performance of more discriminative descriptors such as SIFT and SURF. This removed the need to provide an initial guess of the camera motion, the choice of a search window size (which has a significant impact on the speed/performance of the system) and the necessity to handle failure if the window size is not sufficiently large or the initial motion guess is incorrect.

Chapter 3 introduced several datasets that were subsequently used for testing. Care was taken to ensure the datasets used in the analysis of the visual odometry system came with highly accurate ground truth. This was provided by either computer simulation or DGPS. We also presented the New College Dataset, which, since its publication, has seen significant use in the robotics community. Finally we gave a brief overview of the systems engineering that was required to build and test both the stereo visual odometry and lifelong navigation systems.

To justify the design choices in our stereo visual odometry approach, and enable parameter selection we presented extensive performance results in Chapter 4. We tested on four datasets - two of which are publicly available from a third party - to demonstrate the versatility of the system. We believe that the quality of our system is the product of many well engineered components working in harmony and there is no silver bullet for producing good results. By removing each component individually we immediately see a degradation in performance, indicating that each part contributes to the overall quality of the system. Interestingly we find that for simple pose to pose estimation, due to the high similarity of temporally adjacent frames, a relatively short BRIEF descriptor can be used which brings with it a computation saving. We also present qualitative results on data provided by SCISYS, which was collected during their field trials of our application, and discuss some of the practical issues of running a navigation system for extended periods of time.

With the ability to navigate accurately covered in the previous chapters, we

introduced our new lifelong approach to the problem in Chapter 5. We move away from the notion of single, globally metric monolithic maps containing an averaged version of everything ever seen, and side step the difficult issue of data fusion in the long-term. Instead we model the world with a set of independent experiences, each one allowed to live in its own relative frame of reference, and linked to others only with topological connections. By making new experience acquisition a function of localisation failure we naturally capture the complexity of the workspace with a sufficient number of representations. Results are presented on the same route which was repeated 53 times over a three month period. The data were collected at different times of day under varying weather conditions. We see that visually staid regions can be modelled by relatively low numbers of experiences, while areas that exhibit high visual diversity accrue more. By varying a system parameter - N - we are able to trade off between experience size and localisation performance. Matching is required over larger visual changes and so the system performs better when more discriminative descriptors (such as longer BRIEF variants) are used. We also discuss the connectivity of the topological graph G which links experiences. By using offline introspective step we can further improve the performance of our system, we can stay localised longer and need to save less information to achieve it. Finally we show the system can perform in real time on the vehicle.

6.2 Future Work

There are many possible areas of future research for the work covered in this thesis, especially related to the fledgling lifelong navigation framework. Some options are:

1. **Distraction suppression with richer priors.** While the localisation system has a certain degree of robustness to dynamic objects, eventually it will fail. This is especially problematic when travelling in heavy traffic, where neigh-

bouring vehicles are likely to fill the majority of the camera view. By making use of a rich 3D prior of the environment the idea is to detect and ignore transient objects in the localisation step. The system would track persistent landmarks such as road markings and buildings, but ignore temporary ones like people and other cars. This is currently being investigated by colleagues in the lab.

2. **Lighting invariance.** We find that the long-term navigation approach is prone to massing large numbers of largely unhelpful experiences in regions affected by strong shadows. The ephemeral nature of shadows means the resulting landmark configurations are unlikely to be seen again. This is undesirable and it would be better if the system had some degree of resilience to lighting changes. We have recently begun looking into this problem using a fast image transformation [31, 32] that maps the original image into an illumination invariant space.
3. **Quantifying the utility of an experience.** Taking a more general view on the lighting problem discussed in the previous point, it would be interesting to quantify the usefulness of an experience. One application of this information would be for deletion policies. This could be done at two levels. If features within an individual experience are never seen again, pruning could be performed. At the global level, if whole experiences are not useful - either they become irrelevant through changing environments or they were never useful in the first place - can we remove them without affecting our performance?
4. **Including semantic information.** Folding in semantic information with our experiences would open up a variety of interesting options for future research. Firstly, we could annotate with place identifiers such as office or junction. This could be used in higher level tasks such as path planning and route finding.

Secondly, by folding in numeric or semantic information about the conditions at the time of experience capture - weather conditions, time of day, time of year etc. experiences could be pre-emptively loaded before exploration. Is an experience captured on a damp winter evening useful at midday in the height of summer?

5. **Investigating the acceptable viewing frustum.** This issue is more a function of the visual odometry system rather than the experience based navigation framework. The concern is that the localisation capability of the VO is strongly influenced by the translational and rotational deviation from the prior. While this is not necessarily a problem in relatively constrained environments like road networks it is more pertinent in open spaces such as offices. Investigating this upper bound and how it can be extended would increase the utility of both systems.
6. **Different sensor modalities.** While we have demonstrated extensive results on vision based data, the idea of experienced based navigation is not restricted to vision. Implementation with a laser based navigation system would prove interesting and it would neatly side step some of the lighting issues. A fusion of laser and vision could also be used, this combination has previously been shown to improve pure vision based systems, e.g. FAB-MAP 3D [84].
7. **Multi-robot experiences.** All of our results were collected with a single robot - but why should a robot only benefit from its own experiences? By sharing experiences, a multi robot set up would more quickly map the world and when the first one experiences changes, if this were transmitted to other robots, when they encounter the same situation they would already be equipped to deal with it despite never having experienced it first hand.

6.3 Concluding Remark

The main contribution of this thesis is a new approach to the problem of long-term navigation, a key component of an autonomous robotic system. We hope that our approach, insights and experiences will prove useful for future researchers in this area, and ultimately aid the development of truly lifelong autonomous robots.

Appendix A

Vanilla VO Results on the VO Test Datasets

As discussed in Chapter 4, our error metric is bias for small ground truth values. When the motion of the camera is very small, for example when the robot is stationary, even a small reported motion causes a large *percentage* error. We mark these misleading values with a superscript asterisk (*).

Table A.1: Vanilla VO results on the Synthetic dataset.

	Mean	Median	StdDev	Min	Max
x_e	0.0005	0.0004	0.0005	0.0000	0.0044
y_e	0.0008	0.0005	0.0016	0.0000	0.0294
z_e	0.0008	0.0005	0.0013	0.0000	0.0211
$\theta_{r,e}$	0.0001	0.0001	0.0001	0.0000	0.0011
$\theta_{p,e}$	0.0002	0.0001	0.0002	0.0000	0.0022
$\theta_{q,e}$	0.0001	0.0001	0.0003	0.0000	0.0035
ρ_e	0.0014	0.0011	0.0020	0.0000	0.0365
ϕ_e	0.0001	0.0001	0.0002	0.0000	0.0035
$\Psi_e(\rho_e)$	4.6120	0.7067	38.6381*	0.0423	558.4550*
$\Psi_e(\phi_e)$	1.0201	0.3716	2.1877	0.0009	27.1489*

Table A.2: Vanilla VO results on the Kitti06 dataset.

	Mean	Median	StdDev	Min	Max
x_e	0.0551	0.0548	0.0247	0.0002	0.3516
y_e	0.0112	0.0106	0.0068	0.0000	0.0779
z_e	0.0044	0.0040	0.0031	0.0000	0.0177
$\theta_{r,e}$	0.0003	0.0002	0.0002	0.0000	0.0021
$\theta_{p,e}$	0.0003	0.0002	0.0002	0.0000	0.0018
$\theta_{q,e}$	0.0005	0.0002	0.0008	0.0000	0.0050
ρ_e	0.0570	0.0570	0.0243	0.0027	0.3520
ϕ_e	0.0005	0.0003	0.0008	0.0000	0.0050
$\Psi_e(\rho_e)$	5.1204	4.9006	1.7763	0.3335	31.9838*
$\Psi_e(\phi_e)$	8.5175	5.8002	8.2810	0.0071	91.6476*

Table A.3: Vanilla VO results on the Begbroke dataset.

	Mean	Median	StdDev	Min	Max
x_e	0.0015	0.0011	0.0020	0.0000	0.0610
y_e	0.0011	0.0009	0.0009	0.0000	0.0149
z_e	0.0010	0.0009	0.0007	0.0000	0.0135
$\theta_{r,e}$	0.0002	0.0002	0.0002	0.0000	0.0012
$\theta_{p,e}$	0.0004	0.0003	0.0004	0.0000	0.0033
$\theta_{q,e}$	0.0002	0.0001	0.0001	0.0000	0.0010
ρ_e	0.0023	0.0020	0.0021	0.0001	0.0624
ϕ_e	0.0002	0.0002	0.0002	0.0000	0.0019
$\Psi_e(\rho_e)$	2.0374	1.0249	5.4791	0.0605	74.5880*
$\Psi_e(\phi_e)$	7.1202	2.1909	12.5606	0.0015	116.4150*

Table A.4: Vanilla VO results on the Kitti00 dataset.

	Mean	Median	StdDev	Min	Max
x_e	0.0369	0.0321	0.0216	0.0000	0.1934
y_e	0.0101	0.0078	0.0152	0.0000	0.2736
z_e	0.0064	0.0052	0.0057	0.0000	0.0721
$\theta_{r,e}$	0.0005	0.0003	0.0006	0.0000	0.0086
$\theta_{p,e}$	0.0005	0.0003	0.0007	0.0000	0.0102
$\theta_{q,e}$	0.0008	0.0003	0.0017	0.0000	0.0331
ρ_e	0.0407	0.0350	0.0241	0.0010	0.2754
ϕ_e	0.0009	0.0005	0.0017	0.0000	0.0332
$\Psi_e(\rho_e)$	5.2585	4.6325	4.4673	0.5099	103.2340*
$\Psi_e(\phi_e)$	8.3027	5.2213	12.8836	0.0008	212.7180*

Bibliography

- [1] M. Agrawal and K. Konolige. Rough Terrain Visual Odometry. In *Proceedings of IEEE International Conference on Advanced Robotics (ICAR)*, August 2007.
- [2] T. Bailey and H. Durrant-Whyte. Simultaneous Localisation and Mapping (SLAM): Part II State of the Art. *Robotics and Automation Magazine*, September 2006.
- [3] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, 110:346–359, 2008.
- [4] S. Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 943 – 948, October 2004.
- [5] P. Biber and T. Duckett. Dynamic Maps for Long-Term Operation of Mobile Service Robots. In *Proceedings of IEEE Robotics: Science and Systems*, Cambridge, USA, June 2005.
- [6] D. Borrmann, J. Elseberg, S. S. Rauniar, and A. Nuchter. Lifelong 3D Mapping - Monitoring with a 3D Scanner. In *IEEE International Conference on*

- Intelligent Robots and Systems (IROS) - Workshop on Robotics for Environmental Monitoring*, 2010.
- [7] M. Bosse and R. Zlot. Map Matching and Data Association for Large-Scale Two-dimensional Laser Scan-based SLAM. *International Journal of Robotics Research*, 27:667–691, 2008.
- [8] M. Bosse and R. Zlot. Keypoint design and evaluation for place recognition in 2D lidar maps. *Robotics and Autonomous Systems*, 57(12):1211–1224, December 2009.
- [9] J.-Y. Bouguet. Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/, 2010.
- [10] W. Burgard, C. Stachniss, and D. Haehnel. Mobile Robot Map Learning from Range Data in Dynamic Environments. *Autonomous Navigation in Dynamic Environments, Springer Tracts in Advanced Robotics*, 35:27, 2007.
- [11] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzinski, C. Strecha, and P. Fua. BRIEF: Computing a Local Binary Descriptor Very Fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.
- [12] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *Proceedings of IEEE European Conference on Computer Vision (ECCV)*, September 2010.
- [13] Y. Cheng, M. Maimone, and L. Matthies. Visual odometry on the Mars Exploration Rovers. *IEEE Robotics and Automation Magazine*, vol. 13, June 2006.
- [14] M. Chli and A. J. Davison. Active Matching. *Proceedings of IEEE European Conference on Computer Vision (ECCV)*, pages 72 – 85, 2008.

- [15] W. Churchill and P. Newman. Continually Improving Large Scale Long Term Visual Navigation of a Vehicle in Dynamic Urban Environments. In *Proceedings of IEEE Intelligent Transportation Systems Conference (ITSC)*, 2012.
- [16] W. Churchill and P. Newman. Practice Makes Perfect? Managing and Leveraging Visual Experiences for Lifelong Navigation. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Minnesota, USA, May 2012.
- [17] A. Comport, E. Malis, and P. Rives. Real-time Quadrifocal Visual Odometry. *International Journal of Robotics Research, Special Issue on Robot Vision*, 29(2-3):245–266, 2010.
- [18] P. I. Corke, D. Strelow, and S. Singh. Omnidirectional Visual Odometry for a Planetary Rover. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2004.
- [19] N. Cornelis and L. Van Gool. Fast scale invariant feature detection and matching on programmable graphics hardware. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1 – 8, June 2008.
- [20] M. Cummins and P. Newman. Probabilistic Appearance Based Navigation and Loop Closing. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Rome, April 2007.
- [21] M. Cummins and P. Newman. Highly Scalable Appearance-Only SLAM FAB-MAP 2.0. In *Robotics Science and Systems*, 2009.
- [22] M. Cummins and P. Newman. Appearance-only SLAM at Large Scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 30:1100-1123, November 2010.

- [23] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2003.
- [24] F. Dayoub and T. Duckett. An Adaptive Appearance-based Map for Long-Term Topological Localization of Mobile Robots. *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3364–3369, 2008.
- [25] G. Dubbelman, P. Hansen, and B. Browning. Bias Compensation in Visual Odometry. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2012.
- [26] M. Dunbabin, J. Roberts, K. Usher, G. Winstanley, and P. Corke. A Hybrid AUV Design for Shallow Water Reef Navigation. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2105 – 2110, April 2005.
- [27] H. Durrant-Whyte and T. Bailey. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *IEEE Robotics & Automation Magazine*, 2:99–110, 2006.
- [28] E. Eade and T. Drummond. Scalable Monocular SLAM. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 469–476, New York, NY, USA, June 2006. IEEE Computer Society.
- [29] C. Engels, H. Stewnius, and D. Nistér. Bundle adjustment rules. In *In Photogrammetric Computer Vision*, 2006.
- [30] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974. 10.1007/BF00288933.

- [31] G. Finlayson, M. Drew, and C. Lu. Intrinsic Images by Entropy Minimization. *Proceedings of IEEE European Conference on Computer Vision (ECCV)*, pages 582–595, 2004.
- [32] G. Finlayson, S. Hordley, C. Lu, and M. Drew. On the removal of shadows from images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(1):59–68, 2006.
- [33] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24:381395, 1981.
- [34] F. Fraundorfer and D. Scaramuzza. Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications. *IEEE Robotics Automation Magazine*, 19(2):78 –90, June 2012.
- [35] P. Furgale and T. Barfoot. Stereo mapping and localization for long-range path following on rough terrain. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4410 –4416, May 2010.
- [36] P. Furgale and T. D. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534–560, 2010.
- [37] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930 – 943, August 2003.
- [38] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR)*, Providence, USA, June 2012.

- [39] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent. *Proceedings of IEEE Robotics Science and Systems*, 2007.
- [40] V. Guizilini and F. Ramos. Visual Odometry Learning for Unmanned Aerial Vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.
- [41] C. Harris and M. Stephens. A Combined Corner and Edge Detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, Manchester, 1988.
- [42] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [43] K. L. Ho and P. Newman. Loop Closure Detection in SLAM by Combining Visual and Spatial Appearance. *Robotics and Autonomous Systems*, 54:740–749, 2006.
- [44] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.
- [45] A. Howard. Multi-robot mapping using manifold representations. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 4198 – 4203, May 2004.
- [46] A. Howard, G. Sukhatme, and M. Mataric. Multi-Robot Simultaneous Localization and Mapping using Manifold Representations. *IEEE: Special Issue on Multi-Robot Systems*, October 2006.
- [47] P. Huber. *Robust Statistics*. John Wiley & Sons, New York, 1981.

- [48] Intel. SSE4 Programming Reference. <http://software.intel.com/>, April 2007.
- [49] E. S. Jones and S. Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. In *International Journal of Robotics Research*, volume 30, pages 56–79, January 2011.
- [50] J. Kelly and G. S. Sukhatme. An Experimental Study of Aerial Stereo Visual Odometry. In *Symposium Intelligent Autonomous Vehicles (IAV)*, Toulouse, France, Sep 2007.
- [51] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, Nara, Japan, November 2007.
- [52] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 2969 –2976, June 2011.
- [53] K. Konolige, M. Agrawal, and J. Solà. Large Scale Visual Odometry for Rough Terrain. In *International Symposium on Research in Robotics (ISRR)*, November 2007.
- [54] K. Konolige and J. Bowman. Towards lifelong visual maps. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1156–1163, 2009.
- [55] K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-based maps. *International Journal of Robotics Research (IJRR)*, 29(10), 2010.

- [56] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A General Framework for Graph Optimization. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.
- [57] A. Lambert. Visual odometry aided by a sun sensor and an inclinometer. Master’s thesis, Department of Aerospace Science and Engineering University of Toronto, 2011.
- [58] A. Lambert, P. Furgale, T. Barfoot, and J. Enright. Visual odometry aided by a sun sensor and inclinometer. In *IEEE Aerospace Conference*, pages 1 –14, March 2011.
- [59] H. Lategahn and C. Stiller. City GPS using Stereo Vision. *Proceedings of IEEE International Conference on Vehicular Electronics and Safety*, July 2012.
- [60] 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, Feb. 2009.
- [61] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 2548 –2555, November 2011.
- [62] D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, 2004.
- [63] W. Maddern, M. Milford, and G. Wyeth. Continuous appearance-based trajectory SLAM. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 3595 –3600, May 2011.

- [64] M. Maimone, Y. Cheng, and L. Matthies. Two years of Visual Odometry on the Mars Exploration Rovers: Field Reports. *Journal of Field Robotics*, 24(3):169 – 186, March 2007.
- [65] E. Malis. Improving vision-based control using efficient second-order minimization techniques. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1843 – 1848, May 2004.
- [66] J. Mason, S. Ricco, and R. Parr. Textured Occupancy Grids for Monocular Localization Without Features. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.
- [67] L. Matthies and S. Shafer. Error modeling in stereo navigation. *IEEE Journal of Robotics and Automation*, 1987.
- [68] L. H. Matthies. *Dynamic stereo vision*. PhD thesis, Carnegie-Mellon University, 1989.
- [69] C. McManus, P. Furgale, B. Stenning, and T. D. Barfoot. Visual Teach and Repeat Using Appearance-Based Lidar. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2012.
- [70] C. Mei, S. Benhimane, E. Malis, and P. Rives. Efficient Homography-Based Tracking and 3-D Reconstruction for Single-Viewpoint Sensors. *IEEE Transactions on Robotics*, 24(6):1352–1364, December 2008.
- [71] 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. Special issue of BMVC.

- [72] M. Milford and G. Wyeth. Persistent Navigation and Mapping using a Biologically Inspired SLAM System. *The International Journal of Robotics Research*, 2009.
- [73] H. Moravec. Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. In *Tech Report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & Doctoral Dissertation, Stanford University*, number CMU-RI-TR-80-03. September 1980.
- [74] NASA. Rover Leaves Tracks in Morse Code. http://www.nasa.gov/mission_pages/msl/news/msl20120829f.html, August 2012.
- [75] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010.
- [76] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, Washington, DC, USA, 2011. IEEE Computer Society.
- [77] P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schroeter, L. Murphy, W. Churchill, D. Cole, and I. Reid. Navigating, Recognising and Describing Urban Spaces With Vision and Laser. In *The International Journal of Robotics Research*, 2009.
- [78] P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schroeter, L. Murphy, W. Churchill, D. Cole, and I. Reid. Navigating, Recognising and Describing Urban Spaces With Vision and Laser. *The International Journal of Robotics Research*, 28, October 2009.

- [79] D. Nistér. Preemptive RANSAC for live structure and motion estimation. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 199–206, October 2003.
- [80] D. Nistér. An Efficient Solution to the Five-Point Relative Pose Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–777, 2004.
- [81] D. Nistér, O. Naroditsky, and J. Bergen. Visual Odometry. *Proceedings of IEEE Computer Vision and Pattern Recognition*, 1:652–659, 2004.
- [82] G. Nutzi, S. Weiss, D. Scaramuzza, and R. Siegwart. Fusion of IMU and Vision for Absolute Scale Estimation in Monocular SLAM. *Journal of Intelligent and Robotic Systems*, 61(1-4):287–299, January 2011.
- [83] Oxford Technical Solutions. <http://www.oxts.com/>, 2012.
- [84] R. Paul and P. Newman. FAB-MAP 3D: Topological Mapping with Spatial and Visual Appearance. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 2010.
- [85] G. S. Paul Furgale, Timothy D Barfoot. Continuous-time batch estimation using temporal basis functions. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 2012.
- [86] J. Rehder, K. Gupta, S. T. Nuske, and S. Singh. Global pose estimation with limited GPS and long range visual odometry. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 2012.
- [87] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proceedings of IEEE European Conference on Computer Vision (ECCV)*, May 2006.

- [88] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, October 2008.
- [89] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, November 2011.
- [90] D. Scaramuzza and F. Fraundorfer. Visual Odometry [Tutorial]. *IEEE Robotics Automation Magazine*, 18(4):80–92, December 2011.
- [91] D. Scaramuzza, F. Fraundorfer, and R. Siegwart. Real-Time Monocular Visual Odometry for On-Road Vehicles with 1-Point RANSAC. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 2009.
- [92] K. Schindler, A. Essb, B. Leibe, and L. V. Gool. Automatic detection and tracking of pedestrians from a moving stereo rig. *ISPRS International Journal of Photogrammetry and Remote Sensing*, 65:523–537, 2010.
- [93] G. Sibley, L. Matthies, and G. Sukhatme. Bias reduction filter convergence for long range stereo. In *12th International Symposium of Robotics Research*, 2005.
- [94] G. Sibley, C. Mei, P. Newman, and I. Reid. A System for Large-Scale Mapping in Constant-Time using Stereo. *International Journal of Robotics Research*, 2010.
- [95] G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive Relative Bundle Adjustment. *Proceedings of IEEE Robotics Science and Systems*, 2009.

- [96] G. Sibley, C. Mei, I. Reid, and P. Newman. Planes, Trains and Automobiles - Autonomy for the Modern Robot. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 285–292, Anchorage, Alaska, USA, May 2010. 05.
- [97] G. Sibley, C. Mei, I. Reid, and P. Newman. Vast Scale Outdoor Navigation Using Adaptive Relative Bundle Adjustment. In *International Journal of Robotics Research*, volume 29, pages 958–980, July 2010.
- [98] G. Sibley, G. S. Sukhatme, and L. Matthies. Constant Time Sliding Window Filter SLAM as a Basis for Metric Visual Perception. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA) - Workshop*, 2007.
- [99] 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, May 2009. Data Papers, Peer Reviewed Publication of High Quality Data Sets.
- [100] H. Strasdat, A. Davison, J. Montiel, and K. Konolige. Double Window Optimisation for Constant Time Visual SLAM. *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2011.
- [101] N. Sunderhauf and P. Protzel. Brief-gist - closing the loop by simple means. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1234 –1241, September 2011.
- [102] A. Taneja, L. Ballan, and M. Pollefeys. Image Based Detection of Geometric Changes in Urban Environments. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2011.

- [103] P. H. S. Torr and A. Zisserman. Robust Computation and Parameterization of Multiple View Relations. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 727–732, 1998.
- [104] P. H. S. Torr and A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, 78:138 – 156, 2000.
- [105] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment a modern synthesis. In *Vision Algorithms: Theory and Practice, LNCS*, pages 298–375. Springer Verlag, 2000.
- [106] F. von Hundelshausen and R. Sukthankar. D-nets: Beyond patch-based image descriptors. *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 2941 – 2948, 2012.
- [107] Weather Online UK. <http://www.weatheronline.co.uk>, 2012.
- [108] S. Weiss, D. Scaramuzza, and R. Siegwart. Monocular-SLAM-Based Navigation for Autonomous Micro Helicopters in GPS-Denied Environments. *Journal of Field Robotics*, 28(6):854–874, November 2011.
- [109] B. Williams, N. Hudson, B. Tweddle, R. Brockers, and L. Matthies. Feature and Pose Constrained Visual Aided Inertial Navigation for Computationally Constrained Aerial Vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2011.
- [110] D. Wolf and G. S. Sukhatme. Online simultaneous localization and mapping in dynamic environments. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

- [111] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA) - Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.
- [112] Z. Zhang. Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting. *Image and Vision Computing Journal*, 15:59–76, 1997.