

Localisation using the Appearance of Prior Structure

Alexander D. Stewart
New College



Supervisor:
Professor Paul Newman

Mobile Robotics Group
Department of Engineering Science
University of Oxford

October 2014

Localisation using the Appearance of Prior Structure

Abstract

Accurate and robust localisation is a fundamental aspect of any autonomous mobile robot. However, if these are to become widespread, it must also be available at low-cost. In this thesis, we develop a new approach to localisation using monocular cameras by leveraging a coloured 3D pointcloud prior of the environment, captured previously by a survey vehicle. We make no assumptions about the external conditions during the robot’s traversal relative to those experienced by the survey vehicle, nor do we make any assumptions about their relative sensor configurations.

Our method uses no extracted image features. Instead, it explicitly optimises for the pose which harmonises the information, in a Shannon sense, about the appearance of the scene from the captured images conditioned on the pose, with that of the prior. We use as our objective the Normalised Information Distance (NID), a true metric for information, and demonstrate as a consequence the robustness of our localisation formulation to illumination changes, occlusions and colourspace transformations. We present how, by construction of the joint distribution of the appearance of the scene from the prior and the live imagery, the gradients of the NID can be computed and how these can be used to efficiently solve our formulation using Quasi-Newton methods.

In order to reliably identify any localisation failures, we present a new classifier using the local shape of the NID about the candidate pose and demonstrate the performance gains of the complete system from its use. Finally, we detail the development of a real-time capable implementation of our approach using commodity GPUs and demonstrate that it outperforms a high-grade, commercial GPS-aided INS on 57km of driving in central Oxford, over a range of different conditions, times of day and year.

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.

Alex Stewart, New College

Funding

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

Acknowledgements

Although this thesis bears my name, it would not have been possible without the support and encouragement of a great many people. Foremost amongst them is my advisor Paul Newman. Paul's seemingly boundless energy, optimism and enthusiasm for robotics and life in general have made the last four years a fantastic adventure. To have been a part of the Mobile Robotics Group (MRG) has been a privilege. Paul, now aided by Ingmar, has built a truly amazing place to do robotics, filled with brilliant and friendly people and a vast array of robots & resources.

Despite having grown almost threefold since I started, MRG has maintained the same great atmosphere, outlook and focus, which is a testament to Paul and Ingmar's leadership and the attitudes of all of the members of the group, past and present. I wish to particularly thank, in no particular order: Winston Churchill, Geoff Hester, Will Maddern, Colin 'Bear' McManus, Alastair Harrison and Dan Withers for everything from developing the infrastructure on which the group, and thus this thesis, is built; to debugging recalcitrant robots in horrible weather at 2am, collecting apparently nuisance-free 24 hour datasets and lots of cycling.

Last, but by no means least, I wish to thank my family and friends for all of their love, support and encouragement over the years. This thesis is dedicated to my parents, for their unconditional love and support, and for teaching me the wonderful things that come from hard work and the pursuit of knowledge.

Contents

1	Introduction	2
1.1	Motivation	2
1.2	Contributions	5
1.3	Structure of the Thesis	6
1.4	Publications	6
2	Preliminaries	7
2.1	Transformations in $\mathbb{SE}(3)$	7
2.1.1	Euler Angle Parameterisation	8
2.1.2	Lie Algebra Parameterisation	10
2.2	Cubic B-Splines	12
2.2.1	Cardinal Cubic B-Splines	12
2.2.2	Cumulative form Cardinal Cubic B-Splines	15
2.2.3	Cubic B-Spline Image Interpolation	15
2.3	BFGS Optimisation	17
2.3.1	Wolfe Line Search	17
2.3.2	BFGS Method	20
3	Localisation using the Appearance of Prior Structure	23
3.1	Problem Formulation	23
3.1.1	Assumptions	26

3.1.2	Optimisation Objective	27
3.1.3	Properties of f_{distance}	29
3.2	Related Work	30
3.3	Conclusions	37
4	Information Metrics	38
4.1	Shannon Information	38
4.1.1	Entropy Fundamentals	38
4.1.2	Information Quantities	40
4.1.3	Entropy Equivalence	42
4.1.4	Metric Spaces for Entropy Measures	44
4.2	The Variation of Information	45
4.2.1	Properties of the Variation of Information	45
4.2.2	Equivalence Problem	48
4.3	Normalised Information Distance	49
4.3.1	Normalised Information Distance Variants	50
4.3.2	Properties of the Normalised Information Distance	50
4.3.3	Importance of Normalisation	52
4.3.4	Relationship to Normalised Mutual Information	53
4.4	Application of the NID to LAPS	54
4.4.1	Choice of Colourspace Dimensionality	54
4.4.2	Combining NID from Multiple Cameras	56
4.4.3	Results	57
4.5	Conclusions	64
5	Formulation of Optimisation	66
5.1	Derivatives of NID	70
5.1.1	Gradient of NID	70

5.1.2	Hessian of NID	75
5.2	Differentiable form of Discrete PDFs	78
5.3	Joint Appearance Distribution	80
5.3.1	Spatial Weighting	81
5.3.2	Intensity Weighting	86
5.3.3	Point Projection Jacobians	91
5.3.4	Results	93
5.3.5	Choice of Weighting Method	97
5.4	Optimising NID	100
5.4.1	Levenberg-Marquardt	101
5.4.2	Newton's Method	102
5.4.3	Quasi-Newton	103
5.4.4	Sampling Methods	104
5.4.5	Choice of Optimisation Method	105
5.5	Point Subset Selection	106
5.6	Point Importance Weighting	108
5.7	Conclusions	114
6	Building the Scene Prior	115
6.1	Scene Prior Pipeline	115
6.2	Experimental Platform	115
6.2.1	Sensor Calibration	120
6.3	Construction of Scene Priors	121
6.4	Segmentation of Dynamic Obstacles	126
6.5	Conclusions	127
7	Implementation	128
7.1	System Overview	128

7.2	Calculation of NID & Derivatives on GPUs	132
7.2.1	One Point per Pixel Subsampling of $\mathcal{S}$	135
7.2.2	Building Joint Appearance Distributions on a GPU	138
7.3	Optimising NID	141
7.3.1	Convergence Results	145
7.4	Identifying Localisation Failures	150
7.4.1	Classification Dataset	150
7.4.2	Classification Feature Space	152
7.4.3	SVM Classification of Optimiser Solution	155
7.5	Fusing LAPS with Odometry	163
7.5.1	Spline Fusion	166
7.5.2	Fusing LAPS with Odometry using Spline Fusion	172
7.6	Conclusions	175
8	Localisation using LAPS	176
8.1	Oxford Datasets	176
8.2	Evaluation Metrics	181
8.2.1	Optimised Ground Truth	182
8.2.2	LAPS Initialisation and Restarts	184
8.2.3	Explanation of Figures	185
8.3	Results	188
8.3.1	Summary Results	189
8.3.2	Traversal: Oxford A1 $\rightarrow$ Prior: Oxford A1	194
8.3.3	Traversal: Oxford A2 $\rightarrow$ Prior: Oxford A1	197
8.3.4	Traversal: Oxford B2 $\rightarrow$ Prior: Oxford B1	200
8.3.5	Traversal: Oxford B1 $\rightarrow$ Prior: Oxford B2	203
8.3.6	Traversal: Oxford B3 $\rightarrow$ Prior: Oxford B1	206

8.3.7	Traversal: Oxford B1 $\rightarrow$ Prior: Oxford B3	209
8.3.8	Traversal: Oxford B3 $\rightarrow$ Prior: Oxford B3	212
8.3.9	Traversal: Oxford A1 $\rightarrow$ Prior: Oxford B1	215
8.4	Discussion	218
8.4.1	LAPS Localiser Performance	218
8.4.2	Solution Classifier Performance	220
8.4.3	Failure Cases	220
8.5	Conclusions	224
9	Conclusions	225
9.1	Summary of Contributions	225
9.2	Future Work	226
9.3	Concluding Remark	230
A	Levenberg-Marquardt Equivalence to Gradient Descent for a Single Residual	241

Chapter 1

Introduction

1.1 Motivation

The guiding aim of this thesis is to contribute to the development of a robust, low-cost localisation system for autonomous mobile robots operating outdoors. Localisation is a fundamental aspect of any autonomous mobile robot. Without an accurate estimate of where it is, a robot cannot successfully navigate in its environment and is thus by definition not mobile, at least not in any controlled sense. Furthermore, without an accurate localisation system a robot cannot index into any prior information available about the environment, whether captured by itself or by others, greatly increasing the online complexity of tasks and so limiting the practical capabilities of the robot.

Whilst localisation is a fundamental task, it is not an easy one; particularly outdoors at large-scales. Outdoors, illumination is uncontrolled and highly variable. Appearance changes at all temporal scales, from dynamic shadows, the day-night cycle and local weather conditions through to longer term seasonal and structural changes, for example the construction of a new building, or the destruction of an old one. Operating at large scales necessitates a robust solution. In order that the robot

can move freely and safely around the environment, the system must also operate online in real-time.

The Global Positioning System (GPS) satellite network represents an impressive feat of engineering, providing an approximate georeferenced position with low-cost hardware. However, its accuracy and repeatability, whilst sufficient for a wide variety of tasks, are insufficient in isolation for accurately localising a mobile robot. This is particularly true in dense urban environments where urban canyons and dense foliage / clutter can either block or significantly degrade satellite coverage and of course indoors, where GPS is not available. Unfortunately, these are typically also the areas that require the most accurate localisation to ensure safe operation and to make effective use of any prior information.

A wide variety of approaches have been explored in the literature to tackle the problem of robust outdoor localisation. All of them use locally sampled information about the scene, albeit from a variety of different sensors, instead of globally referenced information as in the case of GPS. Approaches using lidars such as [46, 47, 65, 96, 3] sidestep the issue of changing appearance due to illumination, as lidars operate in the near infrared, and are typically only weakly affected by changes in solar illumination and have no problems operating at night. However, lidars are typically expensive sensors, a Velodyne HDL-64E lidar as used in [46, 47], is \$85k, although [3] uses a cheaper 2D lidar, these are still considerably more expensive than a camera.

Traditionally, state of the art vision-based techniques such as [66, 124, 31, 49, 29, 43, 11] have been feature based. As shown in [29, 11], image features are often brittle with respect to viewpoint and illumination changes, in addition to changes in the specific type of camera used. Whilst more robust image feature descriptors are an active research area, an alternative approach now becoming practical due to increases in compute power are dense methods which operate directly on the image

intensities, and recent work in [27] seeks to combine the two approaches.

An alternative approach used in [14, 63, 57], is to transform the image colour space to compensate for changing illumination, particularly by removing shadows. Although this transform can be very effective, it can also introduce image noise and so is best used in concert with standard techniques as shown in [63]. Another approach developed in [11], is to accept and store variation in the appearance of the scene as different ‘Experiences’ of the environment, thus sidestepping the problem of data association across large differences in appearance. Indeed, the metric for when a new Experience is required is explicitly when localisation to the currently saved set of Experiences fails. Thus the complexity of representation of a place scales automatically based on the degree of visual change observed.

In this thesis we develop a new approach to low-cost localisation using only monocular cameras, by leveraging a coloured 3D pointcloud prior of the operating environment captured previously by a survey vehicle. We assume that the vehicle to be localised is equipped only with one or more calibrated monocular cameras, with complimentary fields of view, and the previously captured coloured 3D pointcloud prior. Our method uses no extracted image features. Instead, it explicitly optimises for the pose which harmonises the information, in a Shannon sense, about the appearance of the 3D scene prior from the captured images conditioned on the pose, with that of the prior.

This model, akin to that used by Google Street View¹, whereby a small number of survey vehicles are equipped with expensive sensor arrays and then used extensively to map the environment seems significantly more efficient. The scene priors are captured once, but used many times, by many different clients which can leverage the benefits of the knowledge obtained from an expensive sensor suite that they themselves do not possess to improve their performance. The success and expansion

¹www.google.com/maps/views/streetview

in coverage of services such as Google Street View have demonstrated that mapping the world in this manner is not only feasible, but practical. In this work, we seek to demonstrate that leveraging these priors for robust, low-cost monocular localisation is also thus.

1.2 Contributions

This thesis makes the following contributions:

- An information theoretic approach to monocular, metric 6-DoF localisation over large scales outdoors, leveraging a 3D coloured pointcloud scene prior. Our approach explicitly optimises for the pose which harmonises the information in a Shannon sense, about the appearance of the scene from the live images conditioned on the pose, with that of the prior.
- A solution classifier to independently identify localisation failures by classifying not only the value of the information metric objective at termination of the optimisation, but also its true shape obtained from sampling. This exploits the bounded metric property of our objective, the Normalised Information Distance (NID) and the characteristic response of information theoretic alignment tasks.
- We developed a real-time capable implementation of our approach using commodity GPUs and demonstrated that its performance exceeds that of a high-grade commercial INS system on 57km of driving in central Oxford over a range of different conditions, times of day and year. As part of which, we demonstrate the performance of the solution classifier and the resulting performance gains for the overall system when it is used.

1.3 Structure of the Thesis

In Chapter 2 we summarise relevant background material used later in the thesis. Chapter 3 introduces the problem formulation of Localisation using the Appearance of Prior Structure (LAPS) and discusses related work. In Chapter 4 we review some information theory fundamentals, the concept of metric spaces for entropy measures and our subsequent objective function, the Normalised Information Distance (NID). We also demonstrate in this chapter how the NID can be applied to the LAPS problem. Chapter 5 develops the detailed formulation of the optimisation, how the gradients of NID can be computed and the choice of optimiser. In Chapter 6 we discuss how the 3D coloured pointcloud Scene Priors used throughout this work are generated. The development of the solution classifier and the real-time capable implementation on commodity GPUs are discussed in Chapter 7 and the results demonstrating their performance on 57km of driving in central Oxford are presented in Chapter 8. Finally, conclusions and some possible avenues for future work are presented in Chapter 9.

1.4 Publications

The concept described in Chapter 3 and preliminary work from Chapter 5 were published at ICRA 2012 in [102]. Preliminary details of the implementation developed in Chapter 7, and an extension using the illumination invariant image transform, were presented at IV 2014 in [55] and in [57]. The point importance weighting from Chapter 5 and the solution classifier developed in Chapter 7 along with the results presented in Chapter 8 have not been previously published. Finally, the core of the work presented here has been filed as a patent.

Chapter 2

Preliminaries

In this chapter we review some important background theory used throughout this work, including transformations in $\mathbb{SE}(3)$, cubic B-Splines for interpolation and BFGS optimisation. For readers who are familiar with these areas, this chapter could be skipped on first reading, or referred to only as necessary throughout the rest of the work.

2.1 Transformations in $\mathbb{SE}(3)$

Throughout this work we make extensive use of rigid body transformations in $\mathbb{SE}(3)$. We use $G_{A,B} \in \mathbb{SE}(3)$ to denote the 4×4 homogeneous transformation matrix that maps 3D points $\mathbf{q}_B$ defined in frame B into frame A: $\mathbf{q}_A \equiv G_{A,B}\mathbf{q}_B$ [71], where $\mathbf{q}$ is in homogeneous form: $\mathbf{q} \equiv [x, y, z, 1]^T$. $G_{A,B}$ has the general form given in Equation (2.1)

$$G_{A,B} \equiv \begin{bmatrix} R_{A,B} & \mathbf{t}_{A,B} \\ \mathbf{0} & 1 \end{bmatrix} \quad (2.1)$$

where $R_{A,B} \in \mathbb{SO}(3)$ is a 3×3 rotation matrix and $\mathbf{t}_{A,B} \in \mathbb{R}^3$ is a translation vector.

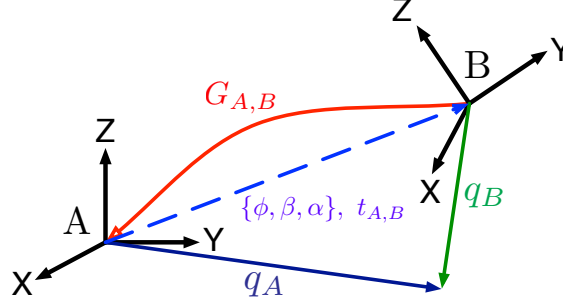


Figure 2.1: Definition of rigid body transformation in $\mathbb{SE}(3)$ using the conventions used in this work. The $\mathbb{SE}(3)$ transform $G_{A,B}$ is defined as that which maps a point $\mathbf{q}_B$ defined in frame B into frame A: $\mathbf{q}_A \equiv G_{A,B}\mathbf{q}_B$. The Euler angle parameterisation of $G_{A,B}$ is defined by the position $t_{A,B}$ and orientation $\{\phi, \beta, \alpha\}$ (roll, pitch and yaw, respectively rotations about the X, Y & Z axes) of the source frame B measured in the destination frame A as detailed in Section 2.1.1.

This is represented graphically in Figure 2.1. All coordinate frames in this work are assumed to be right-handed and orthogonal, and we use the common NASA / aerospace convention by which the base frame for every robot and sensor is aligned such that X is forward, Y is right and Z is down.

We use two methods to construct $G_{A,B}$: Euler angles, for which we use the ‘ZYX’ form; and the Lie Algebra. Euler angles provide a more intuitive minimal representation for a human, and are thus useful when plotting quantities and for quoting an error, whilst the Lie Algebra provides a more compact, cleaner and more easily manipulated algebraic form. In Sections 2.1.1 and 2.1.2 we state the key results & identities used later in this thesis for both methods, for a more detailed discussion of both we direct the reader to [103, 71].

2.1.1 Euler Angle Parameterisation

When using Euler angles to parameterise a transformation, we use the ‘ZYX’ parameterisation [71], and construct the the homogeneous transformation $G_{A,B}$ from the position and orientation of the *origin* frame B *in* the *destination* frame A. Specifically, we define the orientation of the origin frame B in the destination frame

A through three consecutive rotations, starting from frame A in the following order: first yaw (α), then pitch (β), then roll (ϕ). The rotation matrix $R_{A,B} \in \mathbb{SO}(3)$ is then constructed from the product of the rotation matrices for each axis in the *opposite* order to which they were measured, ie: first roll, then pitch then yaw as shown in Equation (2.2),

$$\begin{aligned}
 R_{A,B} &\equiv R_z(\alpha) \cdot R_y(\beta) \cdot R_x(\phi) & (2.2) \\
 &= \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \\
 &= \begin{bmatrix} c_\alpha c_\beta & (c_\alpha s_\beta s_\phi - s_\alpha c_\phi) & (c_\alpha s_\beta c_\phi + s_\alpha s_\phi) \\ s_\alpha c_\beta & (s_\alpha s_\beta s_\phi + c_\alpha c_\phi) & (s_\alpha s_\beta c_\phi - s_\alpha c_\phi) \\ -s_\beta & c_\beta s_\phi & c_\beta c_\phi \end{bmatrix} & (2.3)
 \end{aligned}$$

where in the final form given in Equation (2.3), we have used the abbreviations: $c_\theta \equiv \cos(\theta)$ and $s_\theta \equiv \sin(\theta)$.

Finally, we define the valid ranges for roll (ϕ), pitch (β) and yaw (α) to be: $\{\phi, \alpha\} \in [-\pi, \pi]$, $\beta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. The inverse mapping from $R_{A,B}$ to Euler angles can be obtained from Equation (2.3), albeit as with all 3 dimensional representation of $\mathbb{SO}(3)$, it has a singularity [71]. Thus in any implementation, Quaternions which have 4 parameters and thus no singularity are often preferred for storing the orientation components of a transform. The homogeneous matrix representation is typically preferred when the transform is actually used.

2.1.2 Lie Algebra Parameterisation

Every Lie Group such as $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$ have an associated Lie algebra, which is a tangential vector space defined about the identity element of the group that is both minimal and continuously differentiable [103, 71], and is thus an ideal space in which to perform an optimisation. We denote the Lie algebra representations of $\mathbb{SE}(3)$ and $\mathbb{SO}(3)$ by $\mathfrak{se}_3$ and $\mathfrak{so}_3$ respectively, such that: $G \in \mathbb{SE}(3)$ can be equivalently represented by $\mathbf{g} \in \mathfrak{se}_3$.

The matrix exponential maps $\mathfrak{se}_3 \mapsto \mathbb{SE}(3)$ using the ‘generator’ matrices for the $\mathbb{SE}(3)$ group given in Equation (2.4) as shown in Equation (2.5),

$$\begin{aligned}
 \text{gen}_0 =_{\mathbb{SE}(3)} & \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{gen}_1 =_{\mathbb{SE}(3)} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{gen}_2 =_{\mathbb{SE}(3)} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
 \text{gen}_3 =_{\mathbb{SE}(3)} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{gen}_4 =_{\mathbb{SE}(3)} & \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{gen}_5 =_{\mathbb{SE}(3)} & \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
 \end{aligned} \tag{2.4}$$

$$G(\mathbf{g}) = \exp \left(\sum_{i=0}^5 \mathbf{g}_i \cdot \text{gen}_i_{\mathbb{SE}(3)} \right) \tag{2.5}$$

where: gen_i denotes the i -th generator of $\mathbb{SE}(3)$, with $\mathbb{SE}(3) \in \mathbb{R}^{4 \times 4}$ and $\mathfrak{se}_3 \in \mathbb{R}^{4 \times 4}$.

The $\mathfrak{se}_3 \in \mathbb{R}^{4 \times 4}$ representation can also be transformed from/to a minimal $\mathbb{R}^{6 \times 1}$

form using the hat: $(\cdot)^\wedge : \mathbb{R}^{6 \times 1} \mapsto \mathbb{R}^{4 \times 4}$ and inverse hat: $(\cdot)^\vee : \mathbb{R}^{4 \times 4} \mapsto \mathbb{R}^{6 \times 1}$ operators [103, 71]. The matrix logarithm performs the inverse mapping to that of the matrix exponential: $\mathbb{SE}(3) \mapsto \mathfrak{se}_3$, as shown in Equation (2.6).

$$\mathbf{g}(G) = \log(G) \tag{2.6}$$

Using the definition of the derivative of the matrix exponential given in Equation (2.7),

$$\frac{\partial}{\partial \alpha} \exp(\alpha X) = X \cdot \exp(\alpha X) = \exp(\alpha X) \cdot X \tag{2.7}$$

the partial derivative of an $\mathbb{SE}(3)$ transform with respect to the i -th element of its Lie algebra about $\mathbf{0}$ is exactly the i -th generator of $\mathbb{SE}(3)$. Thus, we can compute quantities such as $\frac{\partial \mathbf{q}_A}{\partial \Delta G}$ for $\mathbf{q}_A = \Delta G \cdot G_{A,B} \mathbf{q}_B$ using Equation (2.8).

$$\begin{aligned} \frac{\partial \mathbf{q}_A}{\partial \Delta G} &= \left(\frac{\partial}{\partial \Delta G} \Delta G \cdot G_{A,B} \mathbf{q}_B \right) \\ \Rightarrow \frac{\partial \mathbf{q}_A}{\partial \Delta \mathbf{g}_i} &= \text{gen}_i^{\mathbb{SE}(3)} \cdot G_{A,B} \mathbf{q}_B \end{aligned} \tag{2.8}$$

Such terms appear frequently throughout the work in this thesis, as we seek to optimise the pose of the robot in $\mathbb{SE}(3)$ based on information about 3D points in the world. This involves iterative updates of a pose estimate of the form: $G_{k+1} = \Delta G \cdot G_k$ at each iteration of the optimisation, for which the Jacobians must be computed.

2.2 Cubic B-Splines

In a number of areas throughout the work described in this thesis we make use of the theory of B-Splines, specifically cubic B-Splines. In Section 5.3 we use the pixel weighting from B-Spline image interpolation as the observation weights when building differentiable histograms, and in Section 7.5.1 we use B-Spline interpolation in $\mathfrak{se}_3$ to fuse LAPS pose estimates with odometry updates in a continuous space.

It is beyond the scope of this work to provide an in-depth overview of the theory of B-Splines as they encompass a very large theoretical area and there are many texts which focus exclusively upon them. Instead, in this section our aim is to state the key aspects of B-Splines necessary to understand this work. For a detailed discussion, we refer the reader to [88, 112].

2.2.1 Cardinal Cubic B-Splines

A B-Spline curve of degree m ($m = 3$ for cubic B-Splines) is a piecewise polynomial defined by Equation (2.9)

$$\mathbf{p}(u) = \sum_{k=0}^n \beta_{k,m}(u) \cdot \mathbf{p}_k \quad (2.9)$$

where $\mathbf{p}_k \in \mathbb{R}^w$, where w is the dimensionality of the space, are called the control points and are the coefficients that parameterise the curve. The control points are weighted by the B-Spline basis functions: $\beta_{k,m}(u)$ which are univariate functions defined over the parametric coordinate of the spline: u . The extent of u defines the extent of the spline, with $\min(u)$ defining the start and $\max(u)$ the end. The value of the spline at any valid u is denoted by $\mathbf{p}(u) \in \mathbb{R}^w$ and exists in the same space as the control points. Note that in general, and in the all cases used in this work, the spline does *not* interpolate the control points.

The points defined in u at which consecutive polynomial pieces meet are called

the knots, and in general the separation between the knots can be arbitrarily chosen within the bounds of valid values of u , including zero (coincident knots). However, in this work we consider only Cardinal B-Splines, where the knots are uniformly spaced across the spline as this is usual when using B-Splines for interpolation. Specifically, we shall assume that the knots are located at integer coordinates.

For the case of cardinal B-Splines, the basis functions $\beta_{k,m}(u)$ weighting the control points are simply offset versions of the same functions for each polynomial piece. Although in Equation (2.9) there is a basis function for every control point, in practice the local modification property of B-Splines means that $\beta_{k,m}(u)$ is only non-zero for degree $+1$ control points. In the case of cubic B-Splines ($m = 3$) there are therefore only $3 + 1 = 4$ control points for which $\beta_{k,3}(u) > 0$. This is a key property, as it means that within a window the spline can be modified in isolation, and makes any resulting system sparse.

A critical property of the B-Spline basis functions used later in this work is the partition of unity given in Equation (2.10).

$$\sum_k \beta_{k,m}(u) = 1 \quad (2.10)$$

Formally, for any parametric coordinate on the spline u , the summation of the weights for all control points is equal to one. This is true irrespective of the degree of the spline.

Taking advantage of the repeating nature of $\beta_{k,m}(u)$ for cardinal cubic B-Splines, and defining the control points with non-zero weights for a given sample point u to be those at: $[u_{i-1}, u_i, u_{i+1}, u_{i+2}]$, with $u_i \leq u < u_{i+1}$, we can parameterise $\beta_{k,m}(u)$ instead by a fractional coordinate $s \in [0, 1]$ which is defined as: $s \equiv u - u_i$. This is shown graphically in Figure 2.2.

Using this form, the non-zero weights assigned to the control points with indices:

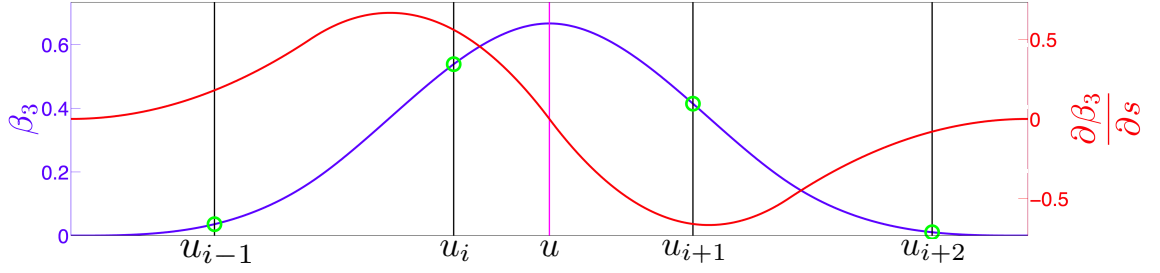


Figure 2.2: Cardinal cubic B-Spline basis function $\beta_3(s)$ (shown in blue) and its derivative (shown in red) with respect to the fractional coordinate s . The weight $\beta_{k,3}(u)$ is only non-zero for the control points with indices $k \in [i-1, i, i+1, i+2]$ defined relative to the sample point in the spline parametric space: u , shown by a vertical pink line, by the points at which they influence the spline: $[u_{i-1}, u_i, u_{i+1}, u_{i+2}]$, shown by vertical black lines, where $u_i \leq u < u_{i+1}$.

$[i-1, i, i+1, i+2]$ are given by Equation (2.11), obtained from the iterative Cox-deBoor formula [88],

$$\begin{aligned}\beta_{i-1,3}(s) &= \frac{1}{6} \cdot (1 - 3s + 3s^2 - s^3) \\ \beta_{i,3}(s) &= \frac{1}{6} \cdot (4 - 6s^2 + 3s^3) \\ \beta_{i+1,3}(s) &= \frac{1}{6} \cdot (1 + 3s + 3s^2 - 3s^3) \\ \beta_{i+2,3}(s) &= \frac{1}{6} \cdot s^3\end{aligned}\tag{2.11}$$

the weights for all other control points are zero. The corresponding derivatives with respect to the fractional coordinate s are given in Equation (2.12).

$$\begin{aligned}\frac{\partial \beta_{i-1,3}(s)}{\partial s} &= \frac{1}{2} \cdot (-1 + 2s - s^2) \\ \frac{\partial \beta_{i,3}(s)}{\partial s} &= \frac{1}{2} \cdot (-4s + 3s^2) \\ \frac{\partial \beta_{i+1,3}(s)}{\partial s} &= \frac{1}{2} \cdot (1 + 2s - 3s^2) \\ \frac{\partial \beta_{i+2,3}(s)}{\partial s} &= \frac{1}{2} \cdot s^2\end{aligned}\tag{2.12}$$

2.2.2 Cumulative form Cardinal Cubic B-Splines

Although the standard form for B-Spline curves is that given in Equation (2.9), there is an equivalent alternative form called the cumulative form [38, 90], given in Equation (2.13).

$$\mathbf{p}(u) = \mathbf{p}_0 \cdot \beta_{0,m} + \sum_{k=1}^n \tilde{\beta}_{k,m}(u) \cdot (\mathbf{p}_k - \mathbf{p}_{k-1}) \quad (2.13)$$

This form is the one used in the Spline Fusion algorithm [53] discussed in Section 7.5.1. The cumulative form basis functions are obtained from the standard basis functions using the relation: $\tilde{\beta}_{k,m}(u) = \sum_{j=k}^n \beta_{j,m}(u)$ [38]. This form is preferable when the control points represent poses in $\mathbb{SE}(3)$ which could be far from the origin, but typically not from their adjacent control points. Thus making Equation (2.13) the combination of small delta transforms, with a single (possibly large) offset. Starting from Equation (2.11) this results in the updated weight functions given in Equation (2.14).

$$\begin{aligned} \tilde{\beta}_{i-1,3}(s) &= 1 \\ \tilde{\beta}_{i,3}(s) &= \frac{1}{6} \cdot (5 + 3s - 3s^2 + s^3) \\ \tilde{\beta}_{i+1,3}(s) &= \frac{1}{6} \cdot (1 + 3s + 3s^2 - 2s^3) \\ \tilde{\beta}_{i+2,3}(s) &= \frac{1}{6} \cdot s^3 \end{aligned} \quad (2.14)$$

2.2.3 Cubic B-Spline Image Interpolation

Although previously we have only discussed B-Spline curves, it is also possible to define a B-Spline surface. These have a mesh of control points, which are weighted by the product of two univariate B-Spline curves, one defined in each of the two

parametric coordinates of the surface [88].

An important use of B-Spline surfaces is subpixel image interpolation, as it results in a C^2 interpolated value, whereas bilinear interpolation is only C^0 . The general form for using B-Splines to interpolate an image is given in Equation (2.15),

$$\mathcal{I}(\mathbf{x}) = \sum_r \sum_c \mathcal{I}^\beta(r, c) \cdot \beta(x - c) \cdot \beta(y - r) \quad (2.15)$$

where as before β denotes the cardinal B-Spline basis functions from Section 2.2.1 and r & c are indices for the rows & columns in the image respectively.

The control points in Equation (2.15) are specified by $\mathcal{I}^\beta(r, c)$. It is important to understand that these are *not* the pixel values of the image to be interpolated. Rather they are coefficients that must be computed using the constraints that Equation (2.15) must evaluate to the values of the pixels in the original image when evaluated at their centers. If this were not true, then the result would correspond to image interpolation. To see this, consider again Equation (2.11) and note that only one term: $\beta_{i+2,3}(s)$ is zero at $s = 0$, which would correspond to sampling at a pixel center in image coordinates in Equation (2.15). Thus if $\mathcal{I}^\beta(r, c)$ were equal to the pixel values from the original image, then Equation (2.15) would *not* (in general) interpolate the original image as it would imply that the value of the pixel is equal to a weighted sum of its neighbours and itself.

Although $\mathcal{I}^\beta(r, c)$ can be obtained by solving a large linear system formed from the interpolation constraints from $\mathcal{I}(r, c)$ and the weights from Equation (2.11), in practice this would be very expensive for an even moderately sized image. Instead, it can be performed much more cheaply with causal & anti-causal Z-transform filters [112] which can be efficiently implemented on a GPU [91, 92]. Note that for a general signal an anti-causal filter would be problematic, but for an image it simply corresponds to traversing the row/column in the opposite direction.

2.3 BFGS Optimisation

For the general problem of minimising an objective function $f : \mathbb{R}^n \mapsto \mathbb{R}$, line search methods [79] compute a descent direction: $\mathbf{p}_k \in \mathbb{R}^n$ at the k -th iteration and then use it to update the current position: $\mathbf{x}_k \in \mathbb{R}^n$, by moving in the direction specified by $\mathbf{p}_k$ by some amount specified by $\alpha_k \in \mathbb{R} > 0$, as shown in Equation (2.16).

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \tag{2.16}$$

The principal differentiator for these methods is how the descent direction $\mathbf{p}_k$ is chosen. However, it is also important to understand how α_k should be chosen at each iteration in order to achieve good performance. In this section, we first introduce the Wolfe conditions which are used to choose α_k . Then we discuss the BFGS method, the best currently known general Quasi-Newton algorithm for computing $\mathbf{p}_k$ by iteratively building an approximation of the Hessian. Our discussion of these methods follows closely that of [79] using their notation, but with a tighter focus on the areas of interest for this work. For a more complete discussion we refer the reader to that text and also [5]. As part of the work described in this thesis, the algorithms in this section were implemented in C++ as part of the Google Ceres Solver [1]¹.

2.3.1 Wolfe Line Search

Consider again Equation (2.16). We now consider how α_k should be chosen, assuming that a valid descent direction $\mathbf{p}_k$ has already been computed. One strategy would be to perform a search along $\mathbf{p}_k$ for the $\alpha_k > 0$ which is the global minimiser. However, this is not an efficient strategy, particularly because $\mathbf{p}_k$ will be recomputed again at the next iteration. Instead, so-called inexact line searches are performed,

¹<http://ceres-solver.org/>

where we require only an α_k at each iteration that reduces f ‘enough’. The key question is now how ‘enough’ should be defined to trade-off the cost of performing the line search at each iteration against the total number of iterations required, in order to obtain good performance.

The Strong Wolfe conditions given in Equations (2.17) and (2.18) [79]

$$f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla \mathbf{f}_k^T \mathbf{p}_k \quad (2.17)$$

$$|\nabla \mathbf{f}(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k| \leq c_2 |\nabla \mathbf{f}_k^T \mathbf{p}_k| \quad (2.18)$$

$$0 < c_1 < c_2 < 1$$

are a principled definition of what constitutes an acceptable α_k , that is, one which sufficiently reduces the univariate function: $\phi(\alpha) \equiv f(\mathbf{x}_k + \alpha \mathbf{p}_k)$. Note that $\phi(\alpha)$ simply abbreviates f moving along the descent direction specified by $\mathbf{p}_k$ from the current position $\mathbf{x}_k$.

The first condition given in Equation (2.17), also known as the Armijo condition, states that the value of the objective must decrease. As $\alpha_k > 0$, $0 < c_1 < 1$ and $\mathbf{p}_k$ is by definition a descent direction, any arbitrarily small α_k will satisfy this condition, represented by all α for which $\phi(\alpha)$ is below the blue line in Figure 2.3.

The second condition, given in Equation (2.18) and known as the curvature condition specifies that the magnitude of the gradient: $\phi'(\alpha)$ must also decrease. This is shown in Figure 2.3 by red tangent lines at the points where: $|\phi'(\alpha)| = c_2 |\nabla \mathbf{f}_k^T \mathbf{p}_k|$. The values of α_k which satisfy both of the strong Wolfe conditions in Figure 2.3 are highlighted in green, note that these values are not unique and are not all local minima; however, they are all closer to local minima than the start point.

The choice of the Wolfe conditions is important, and many line search optimi-

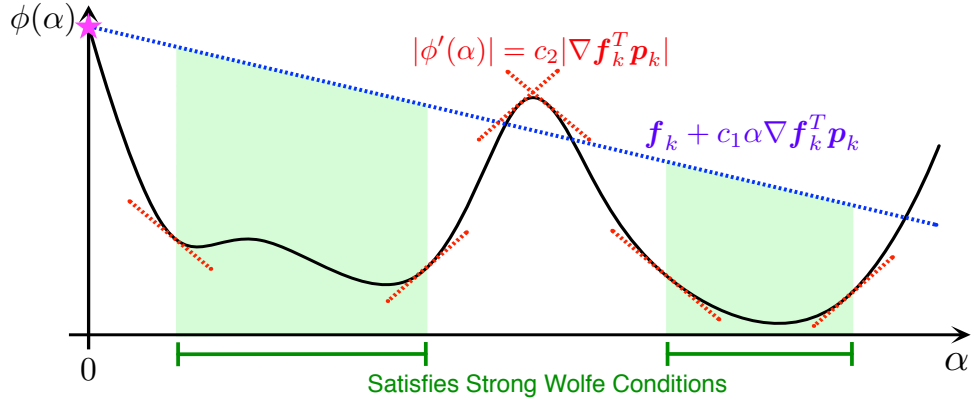


Figure 2.3: Example line search for α_k , showing the graphically the areas satisfying each, and both of the Wolfe conditions; reproduced and updated from [79]. An example univariate function $\phi(\alpha) \equiv f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)$ is shown in black, with the start point: $\mathbf{x}_k$ ($\alpha_k = 0$) highlighted with a pink star. The blue line represents the Armijo condition given by Equation (2.17), all points below it satisfy the constraint. The red lines show the points at which the inequality in the curvature condition given by Equation (2.18) is an equality, i.e. the boundaries at which it is satisfied. The areas highlighted in green are the values of α_k which satisfy both of the strong Wolfe conditions.

sation algorithms, including BFGS, require that the line search performed at each iteration satisfies them. Furthermore, it can be shown that for any continuously differentiable f that is bounded from below along $\mathbf{x}_k + \alpha_k \mathbf{p}_k$ with $\alpha_k > 0$, then there exist intervals of α_k which satisfy the Wolfe conditions.

The implementation of an efficient search algorithm which finds an α_k that satisfies the strong Wolfe conditions is nontrivial. In essence it involves a bracketing phase which searches for a coarse bracket in which a valid α_k must exist, followed by a zoom phase which repeatedly shrinks the bracket until a valid α_k is found [79]. This process involves repeated polynomial interpolation and in order to be efficient, tries to minimise the number of evaluations of both the function and gradient. However, it should be stressed that such a line search can be remarkably effective; for the work presented in this thesis there are typically on average only 1 - 1.2 function & gradient evaluations per optimiser iteration.

2.3.2 BFGS Method

In the general line search update step given earlier in Equation (2.16), it was assumed that $\mathbf{p}_k$ was a descent direction. There are a wide family of methods for choosing $\mathbf{p}_k$, here we focus on the Quasi-Newton BFGS algorithm [79], currently the best known general method [79, 5]. Quasi-Newton methods require only the value and gradient of the objective to be evaluated at each iteration, and then aggregate this information over multiple iterations to iteratively update an approximation of the Hessian: B_k . Using which, the update step from Equation (2.16) becomes Equation (2.19), which is analogous to Newton's method.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \quad (2.16)$$

$$= \mathbf{x}_k - \alpha_k B_k^{-1} \nabla \mathbf{f}_k \quad (2.19)$$

The choice of $\mathbf{p}_k = -B_k^{-1} \nabla \mathbf{f}_k$ in Equation (2.19), comes from modelling the objective as a quadratic at each iteration as shown in Equation (2.20).

$$m_k(\mathbf{p}_k) = f_k + \nabla \mathbf{f}_k^T \mathbf{p}_k + \frac{1}{2} \mathbf{p}_k^T B_k \mathbf{p}_k \quad (2.20)$$

$$\nabla_{\mathbf{p}} m_k = \nabla \mathbf{f}_k + B_k \mathbf{p}_k \quad (2.21)$$

$$\Rightarrow \nabla_{\mathbf{p}} m_k = 0 : \mathbf{p}_k = -B_k^{-1} \nabla \mathbf{f}_k \quad (2.22)$$

Differentiating yields Equation (2.21), and setting equal to zero results in the relation for the minimising descent direction given in Equation (2.22).

Now that a descent direction $\mathbf{p}_k$ has been computed, and assuming that a Wolfe line search has been used to select the corresponding α_k , and thus $\mathbf{x}_{k+1}$, the key question is how B_k should be updated to form B_{k+1} . In BFGS, B_{k+1} is chosen such

that the derivative of the quadratic model of the objective given in Equation (2.21) at the *next* iteration: $k+1$, matches the evaluated gradients of the objective $\nabla \mathbf{f}$ at both the next: $\mathbf{x}_{k+1}$ and current: $\mathbf{x}_k$ positions. As by definition: $\nabla_{\mathbf{p}} m_{k+1}(0) = \nabla \mathbf{f}_{k+1}$, the second condition, given in Equation (2.23), is the one that must be satisfied, resulting in the constraint given in Equation (2.24), known as the secant equation,

$$\begin{aligned} \nabla m_{k+1}(\mathbf{x}_k) &= \nabla m_{k+1}(-\alpha_k \mathbf{p}_k) \\ &= \nabla \mathbf{f}_{k+1} - B_{k+1} \alpha_k \mathbf{p}_k \\ &\equiv \nabla \mathbf{f}_k \end{aligned} \tag{2.23}$$

$$\Rightarrow \underbrace{\nabla \mathbf{f}_{k+1} - \nabla \mathbf{f}_k}_{\mathbf{y}_k} = B_{k+1} \underbrace{\alpha_k \mathbf{p}_k}_{\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k}$$

$$B_{k+1} \mathbf{s}_k = \mathbf{y}_k \tag{2.24}$$

where in order that B_{k+1} remain symmetric and positive definite, it is also required that: $\mathbf{s}_k^T B_k \mathbf{s}_k = \mathbf{s}_k^T \mathbf{y}_k > 0$. However, provided that α_k is chosen by a Wolfe line search, this is guaranteed to be satisfied [79].

The reason Equation (2.24) is referred to as the (multi-dimensional) secant equation is its relation to the update step in the secant optimisation method in 1D given in Equation (2.25) [120].

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{f'(\mathbf{x}_k) - f'(\mathbf{x}_{k-1})} f'(\mathbf{x}_k) \tag{2.25}$$

Indeed, Quasi-Newton methods are generalisations of the 1D secant method to multiple dimensions, as can be seen by comparing Equation (2.19) with Equation (2.24) to Equation (2.25).

The constraints supplied by the secant equation are not on their own sufficient to uniquely determine B_{k+1} , as a symmetric positive definite matrix has $\frac{n \cdot (n+1)}{2}$

degrees of freedom, but Equation (2.24) imposes only n constraints. The requirement that all principal minors, the determinants of the upper-left $m \times m$ sub-matrices $\forall m \in [1, \dots, n]$, be > 0 for positive definite matrices imposes another n constraints (Sylvester's condition). To constrain the remaining degrees of freedom and thus obtain a unique solution, B_{k+1} is chosen to be that 'closest' to B_k , how this is defined is the key differentiator between different Quasi-Newton methods. For BFGS, it is defined using the Frobenius norm: $\|\cdot\|_F$ [118] on the *inverse* of B_k given by $H_k \equiv B_k^{-1}$ and weighted by H_k (although any matrix satisfying the secant equation would be admissible), as shown in Equation (2.26).

$$\begin{aligned} \min_{H_{k+1}} \|H_{k+1} - H_k\|_F & \quad (2.26) \\ \text{subject to: } H_{k+1} &= H_{k+1}^T, \quad H_{k+1} \mathbf{y}_k = \mathbf{s}_k \end{aligned}$$

Equation (2.26) results in a unique solution for H_{k+1} given by Equation (2.27). Note that this equates to a rank-2 update (two rank-1 updates) of H_k . Typically, the identity, or a scaled version of it according to various heuristics, is taken as the initial value: H_0 , which Equation (2.27) is then used to update after each iteration.

$$H_{k+1} = \left(I - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \right) H_k \left(I - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \right) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \quad (2.27)$$

Chapter 3

Localisation using the Appearance of Prior Structure

In this chapter, we first introduce the general form of our problem formulation for localising monocular cameras against a coloured 3D pointcloud prior using appearance, the intuition behind it and state our core underlying assumptions. Then we review related work in vision-based localisation, focussing on that which uses the concept of a prior map, and also the related area of 3D object alignment and tracking using mutual information.

3.1 Problem Formulation

Consider the case of a robot with a camera capturing images as it moves through an arbitrary, known 3D scene $\mathcal{S}$ shown in Figure 3.1. Our goal is to estimate the $\text{SE}(3)$ pose of the robot in $\mathcal{S}$ parameterised by $G_{R,W}$: the transform from the prior frame W in which $\mathcal{S}$ is defined into the robot frame R .

As the structure of the scene is known, the subset of the scene which would reproject into the camera, and where it would reproject, can be calculated for any value

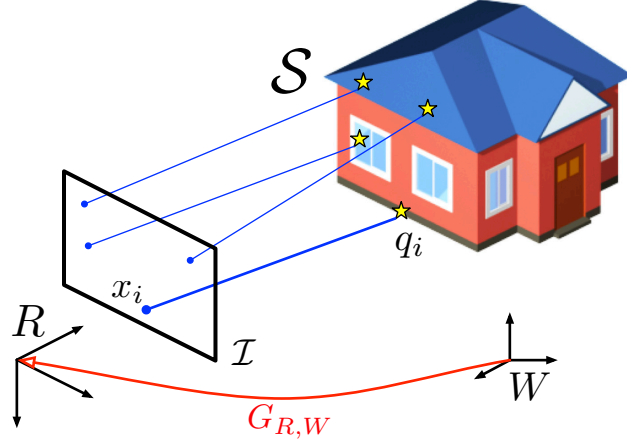


Figure 3.1: A robot equipped with a camera captures images $\mathcal{I}$ as it moves through a previously surveyed scene $\mathcal{S}$ containing 3D points $\mathbf{q}_i$ defined in frame W . For any value of the transform which maps points in W into the robot frame R : $G_{R,W}$, a subset of the points $\mathbf{q}_i$ will reproject into image $\mathcal{I}$ at coordinates $\mathbf{x}_i$.

of $G_{R,W}$. By interpolating in the captured image $\mathcal{I}$ at the reprojected coordinates $\mathbf{x}$, we obtain for every point that reprojects into the image an observation of its appearance under $G_{R,W}$. Assuming that the prior contains an exemplar appearance for every point in addition to its 3D pose, then for every point which reprojected into $\mathcal{I}$ under $G_{R,W}$, we now have two observations of its appearance: one from the prior and one from the image.

The hypothesis for our approach is that the true estimate of the camera pose should harmonise the information about the appearance of the scene from the prior and the live image. This intuition is shown graphically in Figure 3.2.

Figure 3.3 demonstrates the concept from Figure 3.2 using histograms of the reprojected point intensities and real data from [106]. When the pose of the camera is correctly estimated, the appearance histogram of the reprojected points at B closely matches that of the prior viewed from A . Conversely, when the pose is poorly estimated, the resulting intensity statistics have little in common with the prior.

In the context of a ground vehicle operating in busy, mixed environments the

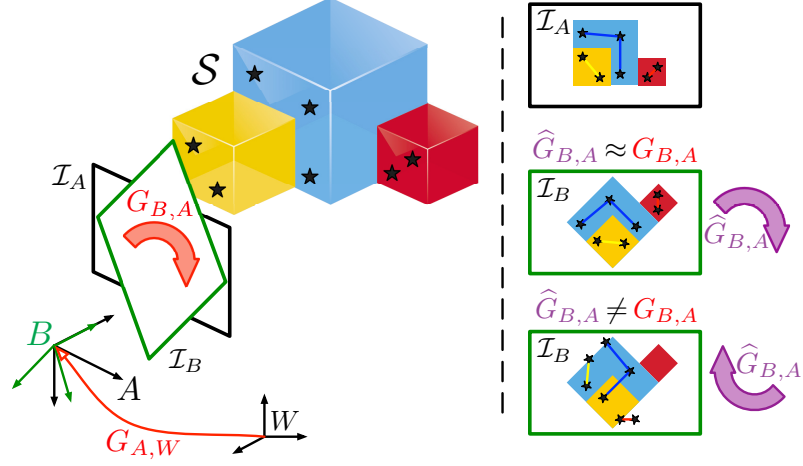


Figure 3.2: Problem intuition. The robot starts at A at which point its pose parameterised by $G_{A,W}$ is known. It then undergoes a clockwise roll after which it captures its second image at B . When the robot’s estimated motion between A & B represented by $\hat{G}_{B,A}$ is accurate ($\hat{G}_{B,A} \approx G_{B,A}$) the reprojection into $\mathcal{I}_B$ of the points in $\mathcal{S}$ align with the image and thus their appearance is consistent. When the estimate is wrong ($\hat{G}_{B,A} \neq G_{B,A}$), the reprojected points do not align with $\mathcal{I}_B$, and so their appearance is inconsistent.

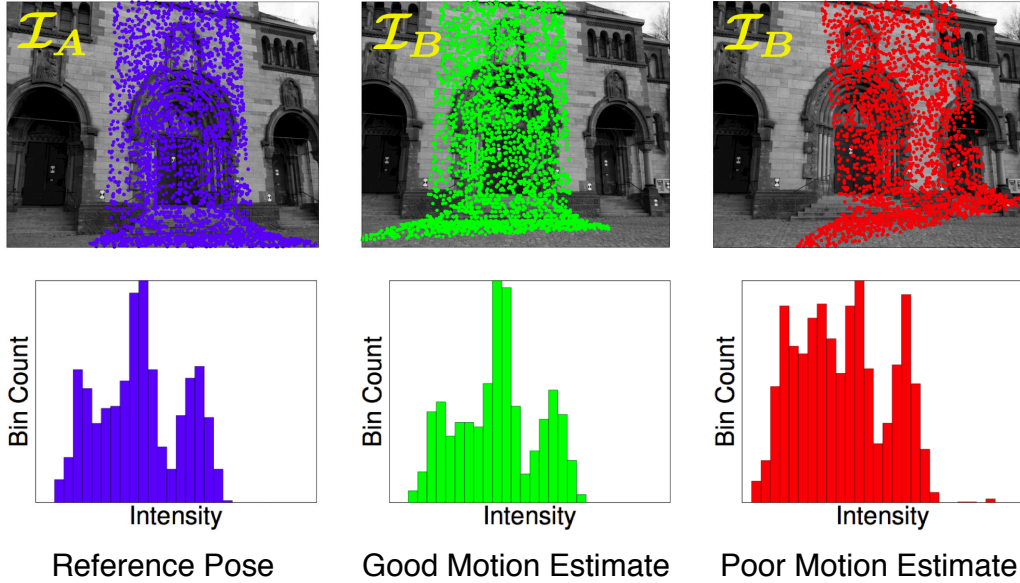


Figure 3.3: Example of Figure 3.2, using real data from [106]. The reprojected points in the prior as viewed from the initial robot position A , whose pose $G_{A,W}$ is accurate are shown in blue. The camera then translates right to capture an image at B . The case when this motion is correctly estimated, and thus the pose of the robot at B is accurate is shown in green, the converse case when the motion is incorrect is shown in red. For each image, the histogram of the intensities (appearance) of the reprojected points is also shown. Data from EPFL Multi-View Stereo Herz-Jesu-P8 benchmarking dataset [106].

geometric structure of the scene is typically complex. Furthermore, representing scenes using meshes, the principal base representation in computer graphics [78] is attractive because GPU hardware is optimised for these primitives. However, whilst purely imagined 3D environments can be constructed natively in this form, systems for capturing the 3D structure of real outdoor scenes do not natively output meshes. Instead 3D lidar systems such as [97, 113] output pointclouds sampled from the environment. The question of how best to construct a mesh from a pointcloud sampled by such a system is still an active research area and storage of the textures associated with the meshes can significantly increase the storage requirements over that of the original pointcloud. Thus we consider $\mathcal{S}$ to represent a coloured pointcloud sampled from the scene, the base representation of the data.

3.1.1 Assumptions

Following the intuition discussed above, we seek a new monocular visual localisation system to localise a robot in a previously surveyed environment under the following assumptions:

- A1. The operating environment has been previously surveyed by a vehicle equipped with a 3D lidar, cameras and an INS.
- A2. A coloured pointcloud prior $\mathcal{S}$, has been built for the operating environment, which captures its 3D structure and an exemplar appearance, that is, a typical example of the scene appearance captured under favourable conditions.
- A3. $\mathcal{S}$ has been preprocessed offline to improve its suitability for localisation, e.g. removal of dynamic objects.
- A4. The robot to be localised is equipped with one or more calibrated monocular cameras with complementary fields of view which capture images syn-

chronously (in time).

- A5. The robot to be localised has a purely relative odometry source, such as wheel odometry, which can be used to provide an estimate of the robot's motion between consecutive image frames as a seed for localisation.

We make no assumptions about the external conditions during the robot's traversal relative to those of the survey vehicle's, for example the weather or illumination conditions. Nor do we make any assumptions about the specific sensor configuration used by the robot relative to that used by the survey vehicle.

An implicit corollary to A4, assumed in most work using vision for metric localisation and also in this thesis, is that the cameras have global, not rolling, shutters and thus all parts of the image are captured at the same instant in time. Although currently global shutter cameras are more expensive, we stress that there is no conceptual limitation preventing the approach developed in this thesis from being applied to rolling shutter cameras. However, it complicates the reprojection of points in the world into the image, a matter we will return to later in the discussion of future work in Chapter 9.

3.1.2 Optimisation Objective

The appearance of a point $\mathbf{q}$ in $\mathcal{S}$ as viewed in $\mathcal{I}$: $\mathcal{I}(\mathbf{x})$ is the value of the image (colour) at the sub-pixel coordinates $\mathbf{x}$ in the image plane to which $\mathbf{q}$ reprojects, where $\mathbf{x}$ is a function of the intrinsic camera calibration and the camera pose. Using the pinhole camera model [33] and denoting the camera parameters by $\boldsymbol{\kappa}$, we can define the reprojection operator $\mathcal{P}$ defined in Equation (3.1)

$$\mathbf{x} \equiv \mathcal{P}(\mathbf{q}, G_{R,W}, \boldsymbol{\kappa}) \quad (3.1)$$

which maps a point $\mathbf{q} \in \mathbb{R}^3$ defined in W , the same reference coordinate system as

the camera, to its image $\mathbf{x} \in \mathbb{R}^2$ in the image plane. We denote the value of the image (colour) at $\mathbf{x}$ by $\mathcal{I}(\mathbf{x}) \in \mathbb{R}^m$ for an image with m channels ($m = 3$ for RGB).

For any given camera and pose, only a subset of $\mathcal{S}$ will reproject into the camera. We represent this subset of $\mathcal{S}$ with: $\overline{\mathcal{S}} \subseteq \mathcal{S}$.

We can now write our problem shown graphically in Figure 3.1 for a single camera as an optimisation over $G_{R,W}$ to minimise an objective function $f_{\text{distance}} : \mathbb{R}^{2 \times (|\overline{\mathcal{S}}| \times m)} \mapsto \mathbb{R}^1$, which measures the discrepancy between the visual appearance of the visible subset of points as shown in Equation (3.3),

$$\arg \min_{G_{R,W}} f_{\text{distance}} \left(\overbrace{\mathcal{I}(\mathcal{P}(\overline{\mathcal{S}}, G_{R,W}, \boldsymbol{\kappa}))}^{\text{Appearance from live image}}, \overbrace{\mathcal{I}_{\mathcal{S}}(\overline{\mathcal{S}})}^{\text{Appearance from Prior}} \right) \quad (3.2)$$

$$= \arg \min_{G_{R,W}} f_{\text{distance}} \left(\mathcal{I}(\mathbf{x}), \mathcal{I}_{\mathcal{S}}(\mathbf{q}) \mid \overbrace{\forall \mathbf{q} \in \overline{\mathcal{S}}}^{\text{Visible subset of } \mathcal{S}} \right) : \mathbb{R}^{2 \times (|\overline{\mathcal{S}}| \times m)} \mapsto \mathbb{R}^1 \quad (3.3)$$

where we use $\mathcal{I}_{\mathcal{S}}(\mathbf{q})$ to denote the appearance of the point $\mathbf{q}$ stored in the prior. This is assumed to have the same dimension as $\mathcal{I}(\mathbf{x})$, the colour of the point from the live image, although the colourspace need not be the same. It is important to note that f_{distance} is computed over the *set of appearance tuples*: $\langle \mathcal{I}(\mathcal{P}(\overline{\mathcal{S}}, G_{R,W}, \boldsymbol{\kappa})), \mathcal{I}_{\mathcal{S}}(\overline{\mathcal{S}}) \rangle$ as shown in Equation (3.2), *not* separately tuple-by-tuple. In the interests of clarity, to emphasise the information required per point (i.e. the components of each tuple), we use the equivalent notation given in Equation (3.3), where the components of the tuple for each point, and the definition of the set over which f_{distance} is computed, are written explicitly within the function signature.

It is clear from Equation (3.3) that $\mathcal{I}(\mathbf{x})$ is dependent upon $G_{R,W}$. But it is important to note that so is $\overline{\mathcal{S}}$, as the visible subset of the scene must be dependent upon the position of the camera and so $\overline{\mathcal{S}}$ should be recomputed for each candidate $G_{R,W}$ during any optimisation.

Although Equation (3.3) is for a single camera, as noted in A4, we are assuming that the vehicle to be localised will have multiple synchronised monocular cameras with complementary fields of view. This is because we do not expect that a single camera would be able to provide good confinement across all parameters of $G_{R,W}$ for all realistic $\mathcal{S}$ and certainly not across all possible $\mathcal{S}$. Thus we extend Equation (3.3) to form Equation (3.4),

$$\begin{aligned} & \arg \min_{G_{R,W}} \sum_c^{\text{Cameras}} f_{\text{distance}} \left(\mathcal{I}_c(\overline{\mathcal{S}}_c), \mathcal{I}_{\mathcal{S}}(\overline{\mathcal{S}}_c) \right) \\ &= \arg \min_{G_{R,W}} \sum_c^{\text{Cameras}} f_{\text{distance}} \left(\mathcal{I}_c(\mathbf{x}), \mathcal{I}_{\mathcal{S}}(\mathbf{q}) \mid \forall \mathbf{q} \in \overline{\mathcal{S}}_c \right) \end{aligned} \quad (3.4)$$

which sums f_{distance} across all cameras. Note that this requires that the concept of summation for f_{distance} exists and that it is theoretically valid.

Two important distinctions between this formulation and feature-based localisation techniques are the issues of feature selection and correspondence. In feature-based formulations such as [98, 43], the features and their correspondences have to be explicitly identified in each new frame with care taken to ensure correct data associations. In Equation (3.4), correspondences are implicitly known through $\mathcal{S}$. The method of point selection is simply that of identifying the visible subset of points for a given camera position, a typically inexpensive operation that can be aided by offline preprocessing of the pointcloud.

3.1.3 Properties of f_{distance}

Numerous potential choices exist for f_{distance} given in Equation (3.4). We desire a function that is robust to different appearance profiles due to different cameras, different illumination conditions and also unmapped dynamic objects within the scene.

We also want f_{distance} to be differentiable, and to have a wide basin of convergence to a clear, ideally unique, minima so that it can be optimised via standard numerical optimisation techniques. Finally, we also require that the concept of addition exists for f_{distance} such that we can meaningfully sum over all cameras in Equation (3.4), formally this means that we want f_{distance} to be a metric [119].

In this work, we choose f_{distance} to be the Normalised Information Distance [50, 51] (NID), a metric analogue of the Mutual Information [95], which we will discuss in detail in Chapter 4. The use of a true information metric is attractive because it allows for an intuitive and concise formulation with a strong theoretical basis. Furthermore, it is robust, and allows for the alignment of data from different modalities, which is in this work we use to account for differences in the appearance of the scene over time, between different camera configurations and different colourspaces.

3.2 Related Work

The problem of metric localisation against a map using locally observed sensor data is inherent to the Simultaneous Localisation and Mapping (SLAM) problem [25, 2] and has been a very active area of research in the robotics community over the last two decades. The use of 2D parallel plane lidars was, and remains, a popular sensor choice for $\mathbb{SE}(2)$ SLAM. Historically, this was largely due to the limited computational resources which could be carried by a mobile robot given the desire for real-time autonomous robot exploration. More recently, exploiting the steady increase in portable compute power, visual SLAM using monocular [22, 23, 26, 105] and stereo cameras [42, 67] has become widespread.

Until recently, the primary focus for visual navigation was feature-based methods. These first extract sparse interest points and image features from the ‘dense’ input images, and then perform localisation and mapping using just these points,

discarding the rest of the information in the image. These methods are attractive, and historically necessary, as they are efficient, in that they reduce the scale of the problem whilst still permitting a robust solution. For a typical image there may be millions of pixels, but only 200-300 sparse feature points. This approach also decouples the problem of data association between the live image and the map from the rest of the system. There is a wide array of different image features, with new variants frequently appearing; currently popular choices include SIFT [54], SURF [4] and more recently BRIEF [8].

Although the development of new more robust image features is still an active research area, a common problem is that they are brittle, in that they have very limited robustness to changes in viewpoint, illumination and the specific model of camera used [29]. However, despite these issues, real-time systems for visual SLAM and the simplified problem of visual odometry, where the ‘map’ of features exists solely from frame to frame, have been successfully demonstrated at vast scales [98, 43, 11, 29, 124, 15] in challenging environments.

An alternative to feature-based approaches are so called ‘direct’ methods. These do not use extract any sparse image features, instead they operate directly on the raw pixel intensities across the whole image, thus making use of all of the information available, although some earlier work in this area used image patches [99]. The popularity of these approaches has been driven in large part by the rise of the use of Graphics Processing Units (GPUs) for general compute tasks (GPGPU), because their inherently massively parallel hardware design means the real-time evaluation of dense objective functions is tractable. The first complete real-time system in this area was DTAM introduced in [77]. The method uses a monocular RGB camera to, in parallel, iteratively build a dense, 3D textured mesh of a small scale workspace and localise against it. The former uses a regularised L_1 variational objective, whilst the latter uses the L_2 norm of the photometric error between the live image and a syn-

thesised image built from the current dense estimate of the workspace. The results demonstrate superior performance to the feature based PTAM method from [39] and strong robustness to dynamic occlusions and defocussing.

The L_2 norm of the photometric pixel error has proven a very popular choice for dense localisation methods. It has been used for $\mathbb{SE}(2)$ visual odometry at large scales using a standard reversing camera on a car in [52] and recently in a semi-dense context for monocular SLAM in [27]. It has also been used for visual odometry in conjunction with depth in an RGB-D context in [101] and using stereovision in [13]. The use of the L_2 norm implies that the appearance of the scene is assumed to be approximately constant, although SLAM systems such as DTAM can handle gradual illumination changes by iteratively updating the texture of the model. However, large and sudden illumination changes will cause them to fail; although for the small scale, indoor workspaces considered this is not a major concern. For visual odometry systems such as [52, 13], the photoconsistency assumption is only made frame to frame, and thus even in the presence of uncontrolled outdoor illumination is still reasonable.

Although SLAM, by definition, involves localisation against a map, in general it is assumed that the map was constructed by the robot being localised and that the sensor configuration has not changed between the point at which the map was constructed and when it is used for localisation. Thus, it is also implicitly assumed that the robot to be localised *can* build and update the map as it has the sensors to do so. This simplifies the problem of data association, as it removes the additional requirement to be robust to different sensor configurations. It should be stressed that this is definitively *not* to say that the data association problem is easy when the sensor configuration is fixed, it is not. Accurate data association is arguably the most significant challenge for any robust large scale system.

In contrast to this approach, there is now increasing interest in separating the

tasks of mapping and localisation and specifically using cheaper, more limited sensor configurations on the robot to be localised than were used on the survey vehicle to build the prior map. As noted in Section 1.1, we feel this approach is very attractive and it is the basis for the the problem formulation described in this chapter.

In [80], a robot with wide-angle cameras was localised outdoors against a sparse, wireframe 3D edge prior map obtained from a manual survey and which contains purely geometric information. Building on the approach from [40] for small scale indoor object tracking, for every image to be localised the wireframe edge map is reprojected into the image plane using a candidate robot pose. Then for every point in the image along each reprojected edge, the closest edge in the image measured perpendicular to the direction of the reprojected edge is identified. A likelihood is then computed by Gaussian weighting this distance and then aggregating for all points on the reprojected edge and across all edges from the map that reproject into the image using the candidate pose. The complete system uses a particle filter to estimate the pose, with the likelihood used as the weight for each particle and a wheel odometry propagation model. The results presented demonstrate the system performs robustly with a performance comparable to a lidar based system in a variety of harsh outdoor conditions including significant sun flare and overexposed images when combined with an intelligent autoexposure method described in the work that maintains clarity in the areas of the image which contain the edges currently being used for localisation.

In the context of road vehicle localisation, [124] describe a system demonstrated over 100km of autonomous driving which uses a combination of a point-feature system using a monocular camera against a prior feature map built using stereo, and a lane-feature localisation system. The latter matches lane markings from a prior map extracted from orthographic images, built using stereo with manual verification, and OpenStreetMap. During localisation, the prior lane map is reprojected into

the areas of the image identified as free-space using stereo and a Kalman filter is used to estimate the $\mathbb{SE}(2)$ pose which aligns the reprojected map with lane features extracted from the live imagery. In [74, 73], the authors demonstrate an $\mathbb{SE}(2)$ correction source for stereo visual odometry using a prior map consisting of orthographic views of the road, which are then matched to the live images using ESM in [74] and the mutual information in [73] over a couple of kilometres of driving, although the approaches described are not real-time.

In [61], a survey grade Riegl scanner is used offline to build a high resolution 3D coloured pointcloud of a moderate scale indoor office environment, from which a coarse voxel octree map is constructed capturing the 3D structure and coloured by averaging the appearance of all points from the survey in the voxel. Online, localisation is performed using a particle filter, with the likelihood for each particle computed by first creating a synthetic image at the candidate pose from the textured prior map using OpenGL. This synthetic image and the live image are then independently renormalised to have zero mean and a standard deviation of one, after which the pixelwise L_2 norm between the images is computed and used as the likelihood weight for the particle. The renormalisation provides some robustness to changes in illumination between the prior and the live image and successful localisation is demonstrated 3 months after the prior was captured, although as results are for an indoor environment, the lighting changes are not significantly affected by the season.

Whilst the L_2 norm is a popular image alignment metric, it is inherently unsuited to the alignment of data from different modalities, as it assumes that the target and reference are embedded in the same space. In medical imaging, cross-modal alignment is required to compose fMRI, X-ray and other modalities into a single representation of a patient. No simple transformation exists between these modalities, and although some features of the patient may be present in both, for

example bones, others such as soft tissue are not. The Mutual Information [95], reviewed later in Chapter 4, has become a popular and robust objective function for performing such cross-modal alignment [59, 89, 24]. For any given alignment, corresponding samples from both modalities are used to construct a joint histogram approximation of the joint distribution of the two sources, from which the mutual information can be computed. The true alignment is then taken to be that which maximises the mutual information between the sources.

Using the mutual information as the objective for alignment of a textured 3D model to an image was first proposed in [115] and then later further developed in [58, 108, 60]. The principal distinctions being how the joint histogram is constructed and the manner in which the optimisation is performed, a matter we shall discuss extensively later in Chapter 5. Building on this work, [86] applied the technique of [108] to the problem of 3D object tracking, in essence a sequential alignment problem. The resulting system achieved improved performance over a feature-based system when demonstrated at small scale indoors, running at 1-2 frames per second.

In [17] it is shown how the concepts from the aforementioned work on mutual information alignment can be applied to visual servoing, where the objective is to compute the control inputs that align the live camera view with a static reference image. In particular, the work also introduces the use of the complete Newton’s method for optimisation with the full Hessian, estimated once at convergence by evaluating the Hessian analytically for the case when the reference is equal to the target and then fixing it throughout the optimisation. The resulting system is real-time and demonstrated to have good robustness to occlusions and differences in illumination between the target and reference images. The authors also applied their visual-servoing technique to teach and repeat navigation of a small vehicle outdoors using a sequence of images in [21] and planar object tracking in [16].

Mutual information has also been used to perform target-free lidar-camera extrin-

sic calibration in [84, 85], where it is computed between the lidar intensity returns and the reprojected grayscale intensity in the camera for a candidate calibration transform. The method is demonstrated to outperform another state-of-the-art method given in [48] using scans and images captured from natural scenes indoors and outdoors, with the mutual information providing robustness to the uncontrolled and varied illumination experienced outdoors evident in the camera imagery, but not the lidar intensity.

Although the focus of this work is metric localisation, in the related field of topological localisation, significant progress has been made in the development of systems that are robust to illumination and appearance changes and can operate at large scales. In [15], topological localisation is demonstrated over a 1000km dataset using FABMAP. In [70, 69], SeqSLAM, which uses whole image matching after patch normalisation on heavily downsampled images, is demonstrated to achieve good performance across large appearance changes by matching the signature obtained from a sequence of images, as first shown in [36]. In [76], SeqSLAM was extended by learning a model for seasonal appearance changes. In [75], matching sequences are found using an alternative cost-flow method that is robust to non-matching sections due to temporal occlusions and where the use of HOG descriptors also provides robustness to seasonal changes.

Since the first publication of our work in [102], two recent publications have taken an approach very similar to that described in this work. In [9], building on the work of [17] and using a method first suggested in [20], a monocular camera is successfully localised outdoors against a textured planar 3D model using the maximisation of the mutual information between the reprojected texture from the model and the live image. However, their approach is far from real-time, taking 4 seconds per frame and is, as a consequence, only demonstrated over a short distance.

In [122], the normalised mutual information is used to compute the $\text{SE}(2)$ pose

of a vehicle by aligning a road surface prior map showing lane markings, defined in lidar intensity space to the live image from a monocular camera. The system runs in real-time at 10 Hz providing corrections for an IMU based EKF and demonstrates comparable performance to a lidar based localisation system using a novel sampling strategy accelerated on a GPU in place of gradient-based optimisation.

Inevitably due to space constraints, this section cannot contain a completely exhaustive exposition of all related work given the scope of the field of vision-based localisation. However, it has been chosen to provide a rich context for the work developed in this thesis. In particular, the use of information theoretic approaches to localisation using vision. Furthermore the references selected have been chosen to provide valuable seeds, via their own references, into the wider corpus.

3.3 Conclusions

In this chapter we introduced the problem formulation used in this work to metrically localise a number of monocular cameras against a 3D coloured pointcloud scene prior. We developed the intuition for our approach and the core underlying assumptions behind it. Finally, we discussed the related work in this area, highlighting the key distinctions from the separate problem of visual SLAM and detail more closely related work on localisation against a prior map using vision. We also reviewed the development of the use of mutual information for image alignment tasks, both in medical imaging and more recently in robotics for a variety of tasks including tracking and extrinsic calibration of sensors for different modalities. In Chapter 4, we discuss information metrics and how they can be applied to our problem formulation and in Chapter 5 we will introduce the specific formulation of the optimisation to be solved.

Chapter 4

Information Metrics

In this chapter, we first review the fundamentals of, and key results from, information theory used in this work, including the concept of metric spaces for information. Then we review two metrics for information, the Variation of Information and the Normalised Information Distance (NID) and explain and demonstrate why the NID is preferable, as it is unbiased. Finally, we show how the NID can be used as the objective in the problem formulation presented in Chapter 3.

4.1 Shannon Information

4.1.1 Entropy Fundamentals

The notion of the information content of a discrete random variable was developed by Claude Shannon in [95] in the context of communication channels. Shannon defined the concept of the entropy of a discrete random variable X , given by Equation (4.1),

$$\begin{aligned}
 H(X) &\equiv \sum_{x \in X} p_X(x) \log \left(\frac{1}{p_X(x)} \right) \\
 &= - \sum_{x \in X} p_X(x) \log(p_X(x))
 \end{aligned} \tag{4.1}$$

Where: $0 \log(0) \equiv 0$

where: $p_X(x) = \Pr(X = x)$. It is important to note that the base of the logarithm is not specified in Equation (4.1), the choice of base defines the resulting units of information. If Base-2 is used, the units of information are bits, if the natural logarithm is used, the units are nats.

The entropy can be thought of as a measure of the uncertainty or randomness of X , or equivalently, the information provided by an observation of X . Formally, the entropy is the expected value of I , the information content of an observation of X . This is referred to as the ‘self-information’ of X and is itself a random variable defined by Equation (4.2).

$$I(X = x) \equiv \log \left(\frac{1}{p_X(x)} \right) \tag{4.2}$$

Intuitively, the more unlikely an event is, the more information its observation conveys. An outcome which is almost certain to occur conveys little information, whilst one that is very unlikely conveys a lot of information. This can be seen graphically in Figure 4.1a, for the case of obtaining heads from a biased coin toss as a function of the probability of heads.

The entropy, or information content, of the biased coin toss as a function of the probability of heads is shown in Figure 4.1b. The entropy is maximised for a fair coin (probability of heads = probability of tails), which is intuitive. As if the coin is biased, irrespective of whether it is biased towards heads or tails, then the

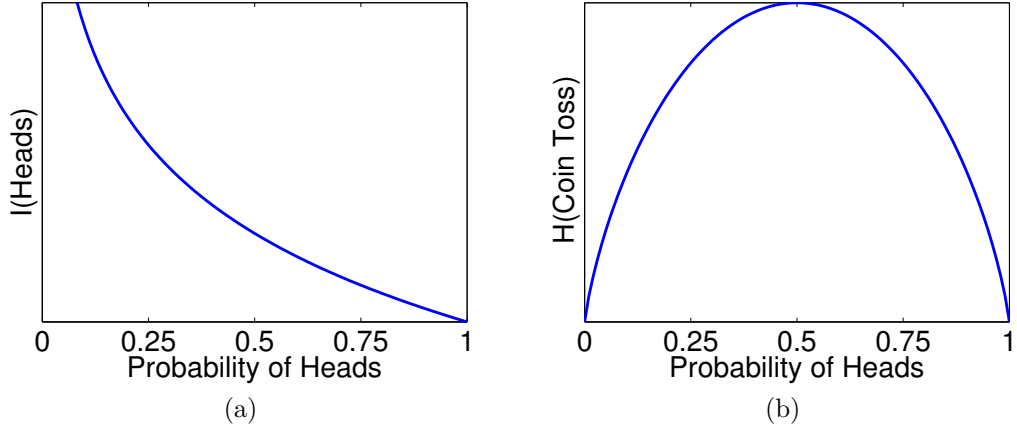


Figure 4.1: Self-information 4.1a and entropy 4.1b of heads for a biased coin toss as a function of the probability of heads. Note that in the limit of 4.1a: $p_{\text{heads}} \rightarrow 0$, $I(\text{heads}) \rightarrow \infty$.

uncertainty of the outcome of a toss is decreased.

4.1.2 Information Quantities

The concept of entropy introduced in Section 4.1.1 can be naturally extended to the bivariate case. Whilst this can be viewed simply by considering the original random variable to represent some subset of the union of outcomes of two new random variables. It is useful to represent these bivariate entropy terms graphically to understand their relations, as shown in Figure 4.2. Where each circle represents a random variable, and area represents entropy, or information.

The joint entropy of X & Y denoted by $H(X, Y)$ (Equation (4.3)),

$$H(X, Y) \equiv \sum_{x \in X, y \in Y} -p_{XY}(x, y) \log(p_{XY}(x, y)) \quad (4.3)$$

$$H(X | Y) \equiv H(X, Y) - H(Y) \quad (4.4)$$

$$I(X; Y) \equiv H(X) + H(Y) - H(X, Y) \quad (4.5)$$

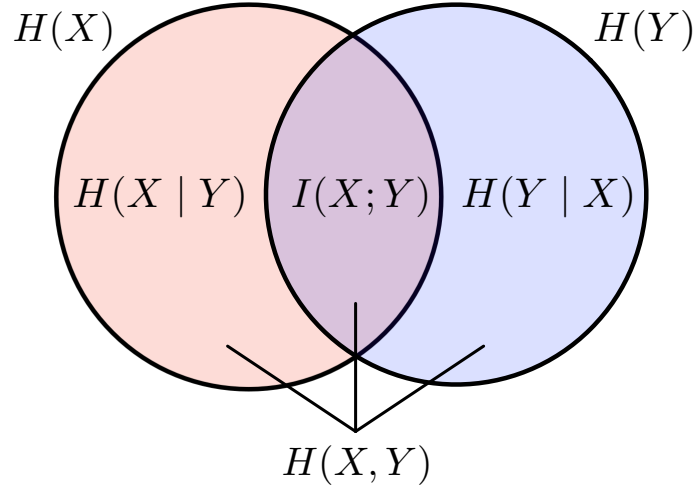


Figure 4.2: Venn diagram representation of the relationship between common multivariate entropy quantities.

is the total information of $\{X \cup Y\}$. Thus, if X and Y were independent, then as X provides no information about Y , the total information of $X \cup Y$ is additive: $H(X, Y) = H(X) + H(Y)$. However, if they are not independent, as shown in Figure 4.2, then $H(X, Y) < H(X) + H(Y)$, as there is information in Y that is common to X and it should not be double counted.

The information common to X & Y is called the mutual information and is denoted by $I(X; Y)$ (Equation (4.6)),

$$I(X; Y) \equiv H(X) + H(Y) - H(X, Y) \quad (4.6)$$

$$H(X | Y) \equiv H(X, Y) - H(Y) \quad (4.7)$$

and is an important measure of the similarity of X and Y . Correspondingly, the conditional entropy of X given Y , denoted by $H(X | Y)$ (Equation (4.7)) is the remaining information in, or uncertainty about, X after Y is known.

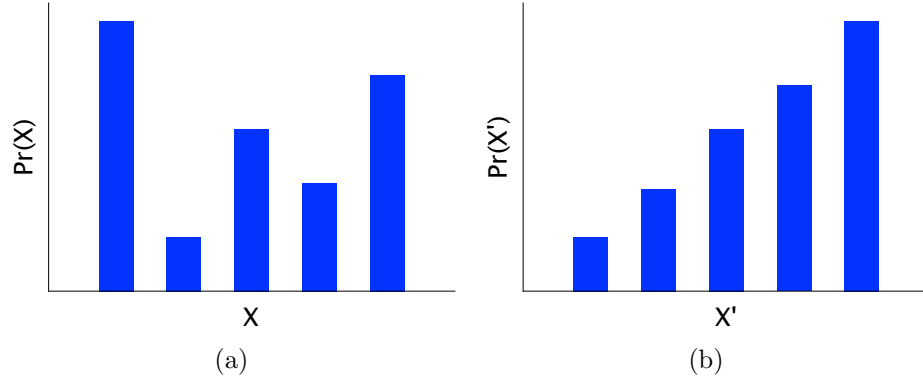


Figure 4.3: Entropy preserving permutations of a univariate distribution. The values associated with the events of X have been arbitrarily permuted from 4.3a to form X' in 4.3b, but the entropy is unchanged: $H(X) = H(X')$.

4.1.3 Entropy Equivalence

Before we discuss the concepts of metrics in an information space, it is important to understand a fundamental aspect of entropy techniques: the entropy is not a function of the *values* the events represent, it is solely a function of their probabilities. Thus we can always arbitrarily change the value or symbol each event represents and as long as the probabilities of all events are unchanged, any entropy quantity involving the random variable will also be unchanged.

For example, in the coin toss considered previously, we can swap which event represents heads or tails and the entropy will be unchanged, as shown by the symmetry about the fair coin case in Figure 4.1b. Graphically, for any univariate discrete distribution, this means that the ordering of the columns of the distribution can be arbitrarily permuted and the entropy will be unchanged, as shown in Figure 4.3.

When considering the joint distribution of two discrete random variables represented as a matrix of probabilities, the joint entropy $H(X, Y)$ is unchanged under an arbitrary permutation of the individual values in the matrix. However, this would *not* in general preserve the marginal entropies $H(X)$ & $H(Y)$, the sums over the rows & columns of the matrix respectively. However, any arbitrary permutation of

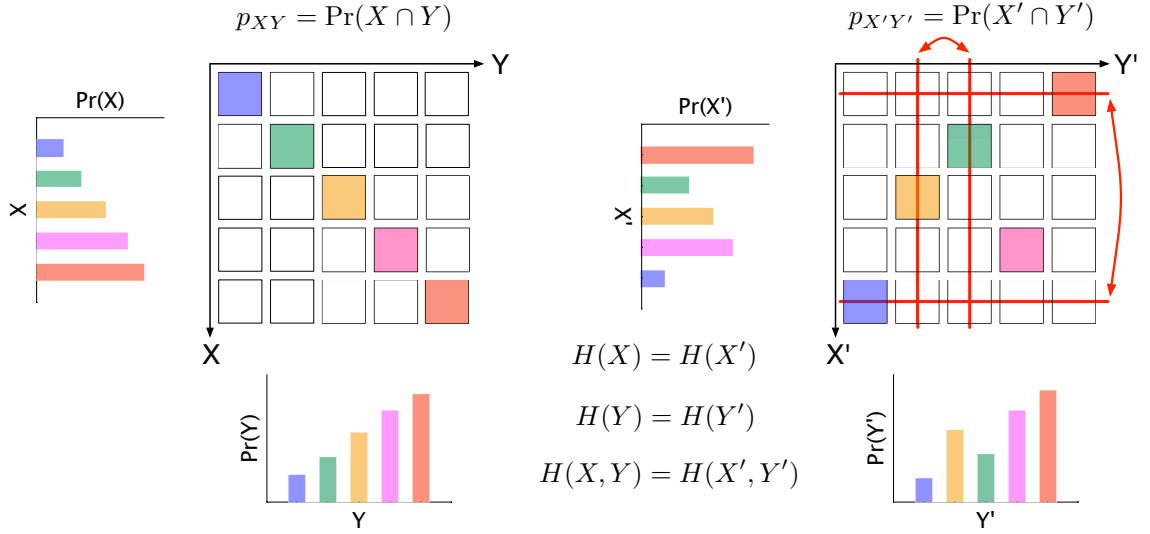


Figure 4.4: Entropy preserving permutations of the bivariate joint distribution. The values associated with the events of X & Y have been arbitrarily permuted, which corresponds to permutations of the rows and columns in the joint distribution p_{XY} , to form $p_{X'Y'}$. This results in no change in any of the entropy quantities, as shown.

the rows and/or columns of the joint distribution will preserve both the joint and the marginal entropies. This is shown graphically in Figure 4.4.

This fundamental independence of the values associated with events is what allows entropy methods to be used for alignment of data in different modalities, where no clear mapping between the modalities may exist. If the random variables represent colour, then this is equivalent to saying that we can arbitrarily permute the discrete quantisation of the colourspaces of X and Y , e.g. inversion, without changing any of the entropy quantities. We will return to this property in the context of our problem in Section 4.4 and present sample results in Section 4.4.3.

A corollary to this property is that if the discrete distributions are estimated from data, the entropy is independent of how many points were used to construct the estimate of the distribution.

4.1.4 Metric Spaces for Entropy Measures

We will now review using entropy quantities as measures of the distance between two different distributions. Formally, a metric function $d(x, y)$ is a measure of distance for a space $\mathcal{M}$ which must satisfy four requirements [119]:

- (a) Nonnegativity: $d(x, y) \geq 0 \quad \forall \{x, y\} \in \mathcal{M}$.
- (b) Equivalence: $d(x, y) = 0 \iff x = y$.
- (c) Symmetry: $d(x, y) = d(y, x)$.
- (d) Triangle Inequality: $d(x, y) + d(y, z) \geq d(x, z)$.

When considering metric entropy measures, it is important to remember that these metric requirements are all made up to an arbitrary permutation of the probabilities associated with the events for each random variable as discussed in Section 4.1.3. Thus a metric entropy measure would satisfy: $d(X, X) = d(X, X') = 0$ from Figure 4.3. This is an important distinction from common metric spaces such as the Euclidean space, where in general: $L_2(\mathbf{x}, \mathbf{x}) = 0 \neq L_2(\mathbf{x}, \mathbf{x}')$ where $\mathbf{x}'$ is an arbitrary permutation of the vector $\mathbf{x}$.

The mutual information $I(X; Y)$ can be used as a measure of the similarity between two distributions and is maximised when one variable wholly contains the information about the second. Assuming that X & Y are drawn from the same space, in that they have the same number of discrete events, this occurs when $X = Y$.

The $I(X; Y)$ is symmetric and has a well defined maximum value: $I_{\max}(X; Y) = \min(H_{\max}(X), H_{\max}(Y))$, as can be see graphically in Figure 4.2 and so could via a linear operation satisfy equivalence. However, it does *not* satisfy the triangle inequality, and is thus not a true metric.

We will now discuss two entropy quantities that are true metrics, the Variation of Information and the Normalised Information Distance.

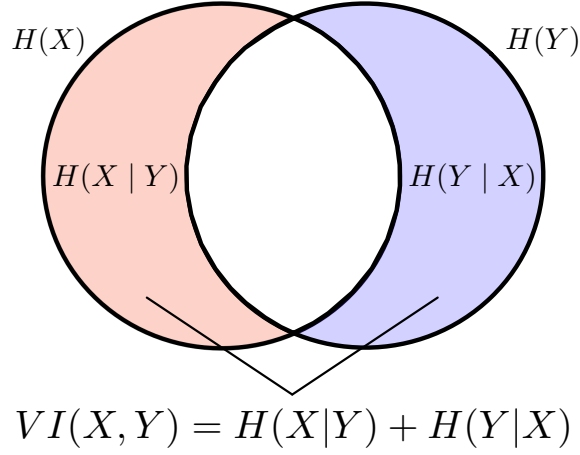


Figure 4.5: Graphical representation of the Variation of Information: $VI(X, Y)$.

4.2 The Variation of Information

The Variation of Information (VI) [68] is defined in Equation (4.8)

$$\begin{aligned}
 VI(X, Y) &\equiv H(X | Y) + H(Y | X) \\
 &= H(X) + H(Y) - 2 \cdot I(X, Y) \\
 &= H(X) + H(Y) - 2 \cdot (H(X) + H(Y) - H(X, Y)) \\
 &= 2 \cdot H(X, Y) - H(X) - H(Y)
 \end{aligned} \tag{4.8}$$

and shown graphically in Figure 4.5 and is a true metric for comparing random variables using their distributions. We will now prove this, and also demonstrate that the VI is bounded.

4.2.1 Properties of the Variation of Information

1 Property:

The VI is a metric.

PROOF In order to be a metric for a space $\mathcal{M}$, a function $d(x, y)$ must satisfy the four properties from Section 4.1.4:

(a) Nonnegativity: $\text{VI}(X, Y) \geq 0 \quad \forall \{X, Y\}$.

By definition: $H(X) \geq 0$ and $\min(H(X), H(Y)) \geq I(X; Y) \geq 0$. Thus $\text{VI}(X, Y) = H(X) + H(Y) - 2 \cdot I(X, Y) \geq 0$ is satisfied.

(b) Equivalence: $\text{VI}(X, Y) = 0 \iff X = Y$.

$\text{VI}(X, Y) = 0$ requires $2 \cdot H(X, Y) - H(X) - H(Y) = 0$, which is satisfied by $H(X, Y) = H(X) = H(Y)$, which implies: $X = Y$.

(c) Symmetry: $\text{VI}(X, Y) = \text{VI}(Y, X)$.

Satisfied by definition, as the addition of entropy terms is commutative, thus:

$$H(X | Y) + H(Y | X) = H(Y | X) + H(X | Y).$$

(d) Triangle Inequality: $\text{VI}(X, Y) + \text{VI}(Y, Z) \geq \text{VI}(X, Z)$.

Following the approach of [68], we first derive Equation (4.10) for discrete random variables $\{X, Y, Z\}$ using the entropy identities in Equation (4.9).

$$\left. \begin{aligned} H(A | B) &\geq H(A | B, C) \\ H(A | B) &\equiv H(A, B) - H(B) \\ H(A, B) &\geq H(A | B) \end{aligned} \right\} \quad \forall \text{ random variables } \{A, B, C\} \quad (4.9)$$

$$\begin{aligned}
& H(Y | X) + H(Z | Y) - H(Z | X) \\
& \geq H(Y | X) + H(Z | Y, X) - H(Z | X) \\
& = \cancel{H(X, Y)} - H(X) + H(X, Y, Z) - \cancel{H(X, Y)} - H(Z | X) \\
& = H(X, Y, Z) - H(X) - H(Z | X) \\
& = H(Y, Z | X) - H(Z | X) \\
\therefore H(Y | X) + H(Z | Y) - H(Z | X) & \geq 0
\end{aligned} \tag{4.10}$$

Now taking Equation (4.10), swapping X & Z and summing with Equation (4.10) and using the definition of VI from Equation (4.8), we can simplify the result to the triangle inequality for the VI.

$$\begin{aligned}
& H(Y | X) + H(Z | Y) - H(Z | X) \\
& \quad + H(Y | Z) + H(X | Y) - H(X | Z) \geq 0 \\
& \text{VI}(X, Y) + \text{VI}(Y, Z) - \text{VI}(X, Z) \geq 0 \\
& \text{VI}(X, Y) + \text{VI}(Y, Z) \geq \text{VI}(X, Z)
\end{aligned} \tag{4.11}$$

2 Property:

The VI is bounded in both directions.

PROOF The lower-bound of zero was proven in VI Property 1 and occurs for $X = Y$. For the upper bound considering Equation (4.8) it is clear that $\text{VI}(X, Y)$ is maximised when the conditional entropies for X & Y are maximised, as conditioning never increases the entropy (Equation (4.9)).

The conditional entropies are maximised under independence $X \perp\!\!\!\perp Y$ when

$H(X | Y) = H(X)$ and $H(Y | X) = H(Y)$. Finally for discrete random variables, the uniform distribution has the largest entropy which equals $\log(k_X)$ and $\log(a) + \log(b) = \log(a \cdot b)$. Thus the VI is bounded in both directions: $\log(k_X \cdot k_Y) \geq \text{VI}(X, Y) \geq 0$.

4.2.2 Equivalence Problem

Whilst the VI appears to be a desirable distance measure between two distributions, there is a significant issue which has not been addressed. Namely that the VI is *biased* by the total information content of the distributions [51, 44].

Consider again Figure 4.5, the VI is represented by the total shaded area, thus any arrangement of resized circles which had the same non-overlapping area would have the same value of VI. Thus it is possible that even if the information content of one of the variables falls dramatically, if the mutual information also shrinks then the VI can be unchanged. This case is shown graphically in Figure 4.6, where $H(Y_2) \ll H(Y_1)$, $I(X_2; Y_2) \ll I(X_1; Y_1)$ and thus $\text{VI}(X_1, Y_1) \approx \text{VI}(X_2, Y_2)$. It is this issue that motivates our introduction of the Normalised Information Distance as an alternative information measure.

This problem also affects the pure mutual information, which whilst not a metric is still used for data alignment tasks. In [107], this problem is described for an alignment problem in medical imaging, where the bias occurs as a result of different amounts of image overlap, and thus different ratios of foreground to background and resulting distributions. The solution presented in [107] introduces the Normalised Mutual Information, which we shall discuss further in Section 4.3.4.

This problem may seem abstract, but in fact this problem results in poor performance if VI were used as f_{distance} in our problem formulation, as will be shown in Figure 4.15 in Section 4.4.3.

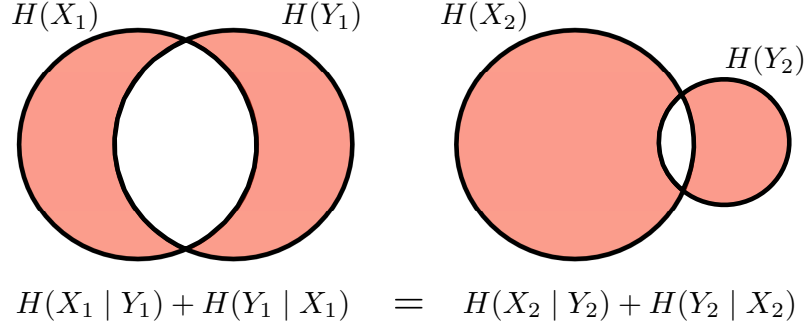


Figure 4.6: Variation of Information equivalence problem. The shaded area representing the VI is equal in both cases. But when comparing distributions, we wish to be able disambiguate between these cases. As the equivalent marginal entropies and greater mutual information in the left hand case means that we would like a distance such that: $d(X_1, Y_1) < d(X_2, Y_2)$.

4.3 Normalised Information Distance

The Normalised Information Distance (NID) [50, 51] for two discrete random variables $\{X, Y\}$ is a relative distance measure obtained by normalising the Variation of Information by the total (joint) entropy. Formally, we use the definition of $\text{NID}(X, Y)$ given Equation (4.12).

$$\begin{aligned} \text{NID}(X, Y) &\equiv \frac{\text{VI}(X, Y)}{H(X, Y)} \\ &= \frac{H(X | Y) + H(Y | X)}{H(X, Y)} \\ &= \frac{H(X, Y) - I(X; Y)}{H(X, Y)} \end{aligned} \tag{4.12}$$

Given their close relationship, the NID inherits variations of the properties given previously for the VI. It is also a true metric and is also bounded in both directions for discrete random variables, which we shall show in Section 4.3.2. In addition, by normalising the VI by the joint entropy $H(X, Y)$, the NID addresses the issue of bias with respect to the entropy of the distributions noted in Section 4.2.2.

4.3.1 Normalised Information Distance Variants

Although the form of NID in Equation (4.12) is the earlier definition [50], there is an alternative, later variant [51] given by Equation (4.13) which is also referred to as the Normalised Information Distance and is also a metric $\in [0, 1]$.

Ambiguously as NID is really a description, in [51] Equation (4.12) is referred to as *a* normalised information distance and Equation (4.13)

$$\text{NID}'(X, Y) \equiv \frac{\max(H(X | Y), H(Y | X))}{\max(H(X), H(Y))} \quad (4.13)$$

as *the* normalised information distance. Other recent work separately refers to Equation (4.12) as the Normalised Variation of Information (NVI) [114].

In [51], the authors argue that Equation (4.13) should be a superior definition to Equation (4.12), as whilst both are universal metrics, the universality bound for Equation (4.13) is tighter. However, as noted in [44], the results presented in [51] do not show a clear benefit.

Furthermore, Equation (4.12) has a critically significant benefit over Equation (4.13) in that it is differentiable. As we wish to use gradient based optimisation techniques for their typically significantly improved performance, any objective must be differentiable. Thus, throughout this work we will always use Equation (4.12) as the definition of NID.

4.3.2 Properties of the Normalised Information Distance

1 Property:

The NID is a metric.

PROOF In order to be a metric for a space $\mathcal{M}$, a function $d(x, y)$ must satisfy the four properties from Section 4.1.4:

- (a) Nonnegativity: $\text{NID}(X, Y) \geq 0 \quad \forall \{X, Y\}$.

By definition: $VI(X, Y) \geq 0$ and $H(X, Y) \geq 0$, thus $NID(X, Y) \geq 0$.

(b) Equivalence: $NID(X, Y) = 0 \iff X = Y$.

As VI is a metric: $VI(X, Y) = 0 \iff X = Y$, thus $NID(X, Y) = 0 \iff X = Y$.

(c) Symmetry: $NID(X, Y) = NID(Y, X)$.

Satisfied as VI is symmetric, and so is the joint entropy $H(X, Y)$.

(d) Triangle Inequality: $NID(X, Y) + NID(Y, Z) \geq NID(X, Z)$.

Following the approach of [44, 114], first we derive Equation (4.15) for discrete random variables $\{X, Y, Z\}$.

Recalling Equation (4.10) from VI Property 1, rearranging yields Equation (4.14), which without loss of generality yields Equation (4.15) after permuting $\{X, Y, Z\}$ ($X \rightarrow Y, Y \rightarrow Z, Z \rightarrow X$).

$$H(Y | X) + H(Z | Y) - H(Z | X) \geq 0 \quad (4.10)$$

$$H(Z | Y) + H(Y | X) \geq H(Z | X) \quad (4.14)$$

$$H(X | Z) + H(Z | Y) \geq H(X | Y) \quad (4.15)$$

Now considering Equation (4.12), we can rewrite the NID as Equation (4.16) and thus to prove the triangle inequality we need to prove the triangle inequality for each of the terms on the r.h.s. However, by virtue of symmetry, we need only show it for one of the two terms:

$$NID(X, Y) = \frac{H(X | Y)}{H(X, Y)} + \frac{H(Y | X)}{H(X, Y)} \quad (4.16)$$

Considering the first r.h.s term, by applying Equation (4.15), we can show the triangle inequality for this quantity and thus also by symmetry, the triangle inequality for NID:

$$\frac{H(X | Y)}{H(X, Y)} = \frac{H(X | Y)}{H(Y) + H(X | Y)} \quad (4.17)$$

$$\leq \frac{H(X | Z) + H(Z | Y)}{H(Y) + H(X | Z) + H(Z | Y)} \quad (4.18)$$

$$= \frac{H(X | Z) + H(Z | Y)}{H(X | Z) + H(Y, Z)} \quad (4.19)$$

$$\leq \frac{H(X | Z)}{H(X | Z) + H(Z)} + \frac{H(Z | Y)}{H(Y, Z)} \quad (4.20)$$

$$= \frac{H(X | Z)}{H(X, Z)} + \frac{H(Z | Y)}{H(Z, Y)} \quad (4.21)$$

$$\therefore \frac{H(X | Y)}{H(X, Y)} \leq \frac{H(X | Z)}{H(X, Z)} + \frac{H(Z | Y)}{H(Z, Y)} \quad (4.22)$$

$$\Rightarrow \frac{H(X | Z)}{H(X, Z)} \leq \frac{H(X | Y)}{H(X, Y)} + \frac{H(Y | Z)}{H(Y, Z)} \quad (4.23)$$

2 Property:

The NID is bounded in both directions: $1 \geq NID(X, Y) \geq 0$.

PROOF Following from VI Property 2: $H(X, Y) \geq VI(X, Y) \geq 0$, it is clear that $1 \geq NID(X, Y) \geq 0$.

4.3.3 Importance of Normalisation

The normalisation by the joint information in the NID rectifies the equivalence problem identified for the VI in Section 4.2.2. To understand this, consider again Figure 4.6. Although $VI(X_1, Y_1) = VI(X_2, Y_2)$, the joint information is different in each case: $H(X_1, Y_1) > H(X_2, Y_2)$. Hence after normalisation $NID(X_1, Y_1) <$

$\text{NID}(X_2, Y_2)$ as desired.

4.3.4 Relationship to Normalised Mutual Information

The Normalised Mutual Information (NMI) defined in Equation (4.24)

$$\text{NMI}(X, Y) \equiv \frac{H(X) + H(Y)}{H(X, Y)} \quad (4.24)$$

was first introduced in [107] where it was maximised to align medical images in different modalities, but has since been used in other work including in [122] which uses a problem formulation similar to our own. As discussed in Section 4.2.2, NMI was introduced to combat the same bias by the total information content as the NID. In fact, NMI is a simple linear function of NID:

$$\begin{aligned} \text{NID}(X, Y) &= \frac{H(X | Y) + H(Y | X)}{H(X, Y)} \\ &= \frac{H(X, Y) - I(X; Y)}{H(X, Y)} \\ &= \frac{2 \cdot H(X, Y) - H(X) - H(Y)}{H(X, Y)} \\ &= \frac{2 \cdot \cancel{H(X, Y)}}{\cancel{H(X, Y)}} - \frac{H(X) + H(Y)}{H(X, Y)} \\ &= 2 - \text{NMI}(X, Y) \end{aligned} \quad (4.25)$$

although, it should be noted that NMI is not a metric as $1 \leq \text{NMI}(X, Y) \leq 2$, with $\text{NMI}(X, X) = 2$ and $\text{NMI}(X, Y) = 1 \iff X \perp\!\!\!\perp Y$. Hence it does not satisfy equivalence and it is not a distance as it should be maximised not minimised and so does not satisfy the triangle inequality.

4.4 Application of the NID to LAPS

Consider again the optimisation formulation from Section 3.1.2 given in Equation (3.4):

$$\begin{aligned} & \arg \min_{G_{R,W}} \sum_c^{\text{Cameras}} f_{\text{distance}} \left(\mathcal{I}_c(\bar{\mathcal{S}}_c), \mathcal{I}_S(\bar{\mathcal{S}}_c) \right) \\ & = \arg \min_{G_{R,W}} \sum_c^{\text{Cameras}} f_{\text{distance}} \left(\mathcal{I}_c(\mathbf{x}), \mathcal{I}_S(\mathbf{q}) \mid \forall \mathbf{q} \in \bar{\mathcal{S}}_c \right) \end{aligned} \quad (3.4)$$

By modelling the appearance of each point in $\bar{\mathcal{S}}$ from the prior, and the live image from each camera under $G_{R,W}$, as simultaneous samples from two discrete random variables we can build a *joint appearance distribution*: $p_{\mathcal{I}_c \mathcal{S}}$ of $\bar{\mathcal{S}}_c$ for each camera. In which the axes are the (quantised) appearance of the scene in the live data and the prior respectively as shown in Figure 4.7, and from which the NID can be computed. It should be noted that the details of exactly how each $p_{\mathcal{I}_c \mathcal{S}}$ is constructed to enable analytical derivatives to be computed is the subject of Section 5.3.

Whilst it may appear that the 3D information is not utilised in $p_{\mathcal{I}_c \mathcal{S}}$, this is not the case. The 3D location of a point $\mathbf{q}$ and the pose $G_{R,W}$ in addition to the camera model, determine the reprojection of the point $\mathbf{x}$ in the live image and hence its appearance. Thus $p_{\mathcal{I}_c \mathcal{S}}$ is inherently a function of the 3D scene information, as it defines the samples from which $p_{\mathcal{I}_c \mathcal{S}}$ is constructed.

4.4.1 Choice of Colourspace Dimensionality

Although the general minimisation in Equation (3.4) assumed a generic m -channel appearance space, it is not practical to build the joint appearance distribution for $m > 1$. This is because for larger values of m even with coarse quantisation the number of cells in the joint distribution becomes very large, which in practical

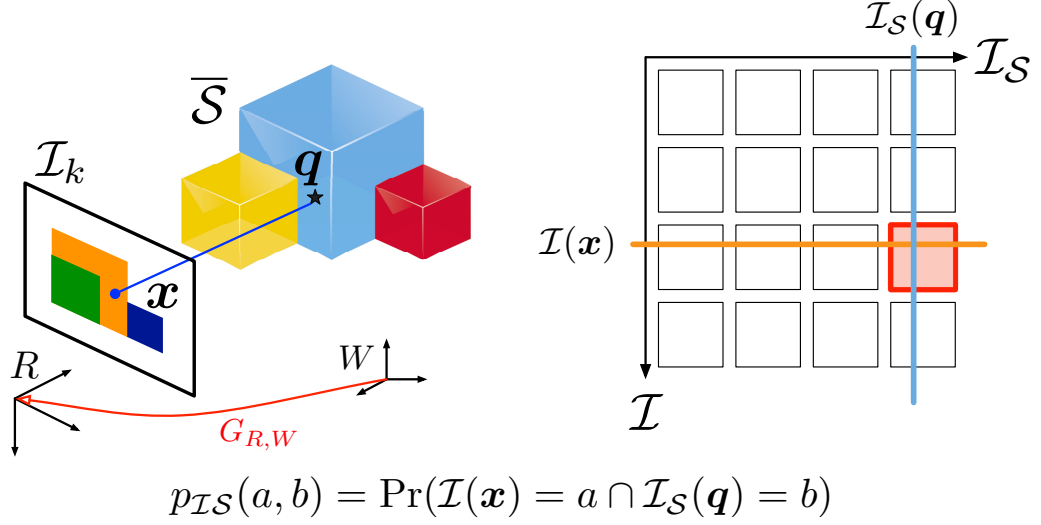


Figure 4.7: Construction of Joint appearance distribution. For every point $\mathbf{q} \in \bar{\mathcal{S}}$, we have two observations of its appearance, from the prior: $\mathcal{I}_S(\mathbf{q})$ and from the live image under $G_{R,W}$: $\mathcal{I}(\mathbf{x})$. Treating these two appearance values as simultaneous observations from different random variables, we can build the discrete joint appearance distribution $p_{\mathcal{IS}}$ using a normalised histogram. Whilst this figure is purely illustrative, in Section 4.4.3 we present some typical $p_{\mathcal{IS}}$ built from real data.

terms results in it becoming very sparse which detrimentally affects the sensitivity and utility of entropy measures such as NID. As NID is a metric, it is conceptually meaningful to sum the NID computed for each channel separately. Hence we can substitute $\text{NID}(X, Y)$ as the objective function in Equation (3.4) to form a candidate minimisation given in Equation (4.26):

$$\begin{aligned} & \arg \min_{G_{R,W}} \sum_c^{\text{Cameras}} \sum_m^{\text{Channels}} \text{NID}(\mathcal{I}(\bar{\mathcal{S}}), \mathcal{I}_S(\bar{\mathcal{S}})) \\ &= \arg \min_{G_{R,W}} \sum_c^{\text{Cameras}} \sum_m^{\text{Channels}} \text{NID}(\mathcal{I}(\mathbf{x}), \mathcal{I}_S(\mathbf{q}) \mid \forall \mathbf{q} \in \bar{\mathcal{S}}) : \mathbb{R}^{2 \times (|\bar{\mathcal{S}}| \times m)} \mapsto \mathbb{R}^1 \quad (4.26) \end{aligned}$$

However, although Equation (4.26) is for a general m -channel colourspace (e.g. $m = 3$ for RGB), in this work we will concern ourselves only with the single channel $m = 1$ case, albeit with different (single channel) colourspaces. This is because we

have observed minimal benefit when using Equation (4.27) for RGB vs Grayscale colourspaces and the additional computational cost is significant. As such, our proposed minimisation assuming single channel images simplifies to Equation (4.27):

$$\begin{aligned} \arg \min_{G_{R,W}} \sum_c^{\text{Cameras}} \text{NID}(\mathcal{I}_c(\bar{\mathcal{S}}_c), \mathcal{I}_{\mathcal{S}}(\bar{\mathcal{S}}_c)) \\ = \arg \min_{G_{R,W}} \sum_c^{\text{Cameras}} \text{NID}(\mathcal{I}_c(\mathbf{x}), \mathcal{I}_{\mathcal{S}}(\mathbf{q}) \mid \forall \mathbf{q} \in \bar{\mathcal{S}}_c) \end{aligned} \quad (4.27)$$

the minimum of which is the pose $G_{R,W}$ which in a strict and intuitive sense maximises the similarity of the information about the appearance of the scene from the prior and live imagery, as desired. We also inherit the beneficial properties of NID, namely its robustness and independence of the quantised colourspace values, which is important as we are not assuming that the prior and live imagery were captured with the same cameras, or under similar conditions.

4.4.2 Combining NID from Multiple Cameras

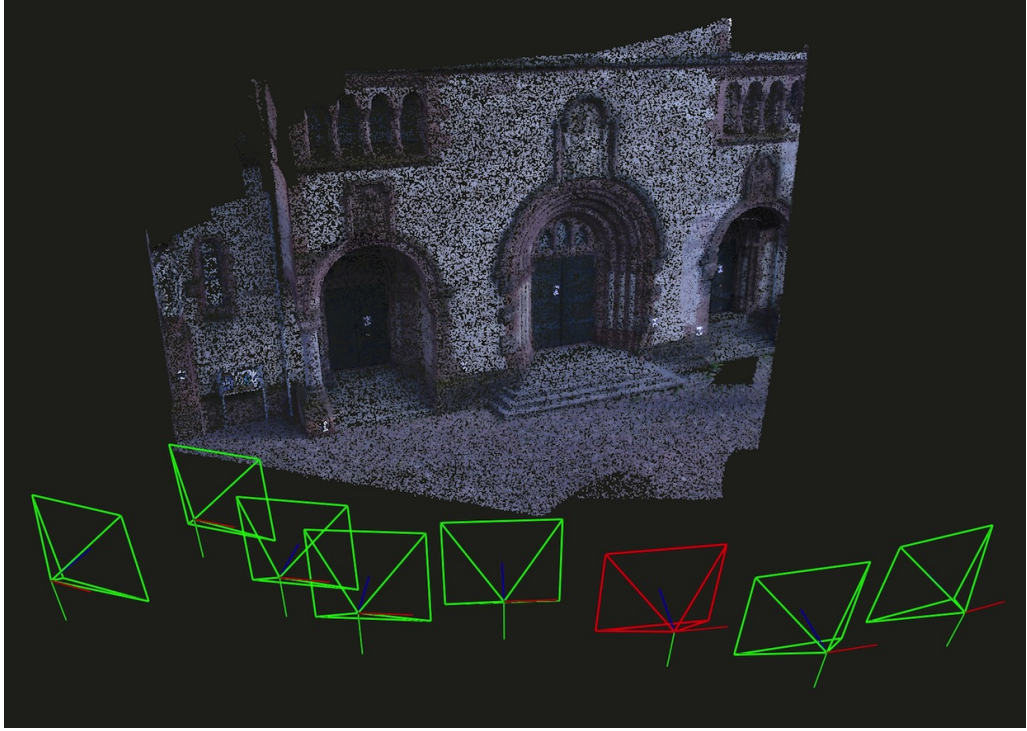
The summation of the NID over multiple cameras in Equation (4.27), assumed to have complementary fields of view and thus computed with different subsets of $\mathcal{S} : \bar{\mathcal{S}}_c$, is justified provided that the space in which $p_{\mathcal{I}_c}$ is defined is the same for all cameras. If this is true, then the $p_{\mathcal{I}_c \bar{\mathcal{S}}_c}$ for each camera can be thought of as local estimates of a complete $p_{\mathcal{I}_{\mathcal{S}}}$ for the scene. An alternative would be to combine all appearance observations from all cameras into a single $p_{\mathcal{I}_{\mathcal{S}}}$ and then use it to calculate a single value for the NID. However, this has the disadvantage of biasing the contribution of each camera by $|\bar{\mathcal{S}}_c|$, the number of points visible within it. It also complicates the artificial weighting of the contributions of different cameras, an extension discussed later in Section 5.6.

4.4.3 Results

Using the problem formulation given in the preceding section, we now present some preliminary results demonstrating the shape and robustness of the formulation on a real dataset. The dataset in question is the EPFL Herz-Jesu-P8 Multi-view Stereo dataset from [106]. It contains a dense pointcloud from a survey lidar and eight registered images of the entrance to a building as shown in Figure 4.8. The appearance prior for the pointcloud was constructed from the mean grayscale intensity (obtained from the standard RGB to Gray transform) of each point across all images in which it was visible. Throughout the results in this section we use a single image, highlighted in red in Figure 4.8, as the ‘live’ image $\mathcal{I}$ in the formulation presented previously.

Figures 4.9 and 4.10 show the iso-surface volumes of the NID evaluated for a regular mesh of samples in translation (XYZ) and orientation (roll, pitch & yaw) respectively about the ground-truth pose. On the right in each figure are plots of the NID and its analytical gradient for displacements along each axis independently, shown as black lines in the main iso-surface plot. Note that how the analytical gradient is computed is the subject of Chapter 5. The key point to note from Figures 4.9 and 4.10 is that the NID appears very well behaved with a clear minima, suggesting that an optimisation using the problem formulation presented earlier would correctly estimate the pose. Although we shall discuss the analytical gradients of NID in more detail later in Chapter 5, it should also be noted from Figures 4.9 and 4.10 that these are also well behaved and so are suitable for use in an optimisation.

As the images in the EPFL Herz-Jesu-P8 dataset were all taken at approximately the same time under overcast conditions, the appearance of the visible points in the live image using the ground-truth pose are very similar to those from the appearance prior. To demonstrate that the inherent robustness of the NID transfers to our pose estimation formulation, we modify the original image $\mathcal{I}$, by changing its illumina-



(a) Scene Prior, coloured with average appearance across all of the images



(b) Images used to colour the scene prior, the ‘live’ image used in the later plots is highlighted in red

Figure 4.8: EPFL Herz-Jesu-P8 Multi-view Stereo Dataset [106]. Ground-truth lidar subsampled to $2 \cdot 10^5$ points (from $18 \cdot 10^6$), coloured using average appearance across all images in which they were visible. The images are represented at their captured poses by green frustrums. The image whose ground-truth pose is represented by a red frustrum, is used as the ‘live’ image $\mathcal{I}$ for all subsequent plots in Section 4.4.3, i.e. is the image whose pose relative to the prior $\mathcal{S}$ is to be estimated in our formulation. For the results presented in Section 4.4.3, no optimisation is performed. Instead, the NID is evaluated for $\mathbb{SE}(3)$ offsets about the ground-truth pose shown to demonstrate the robustness of the NID and the form of its convergence basin. This reflects how our approach will function in practice via the presence of a seed pose, from either the previous iteration, or an external source.

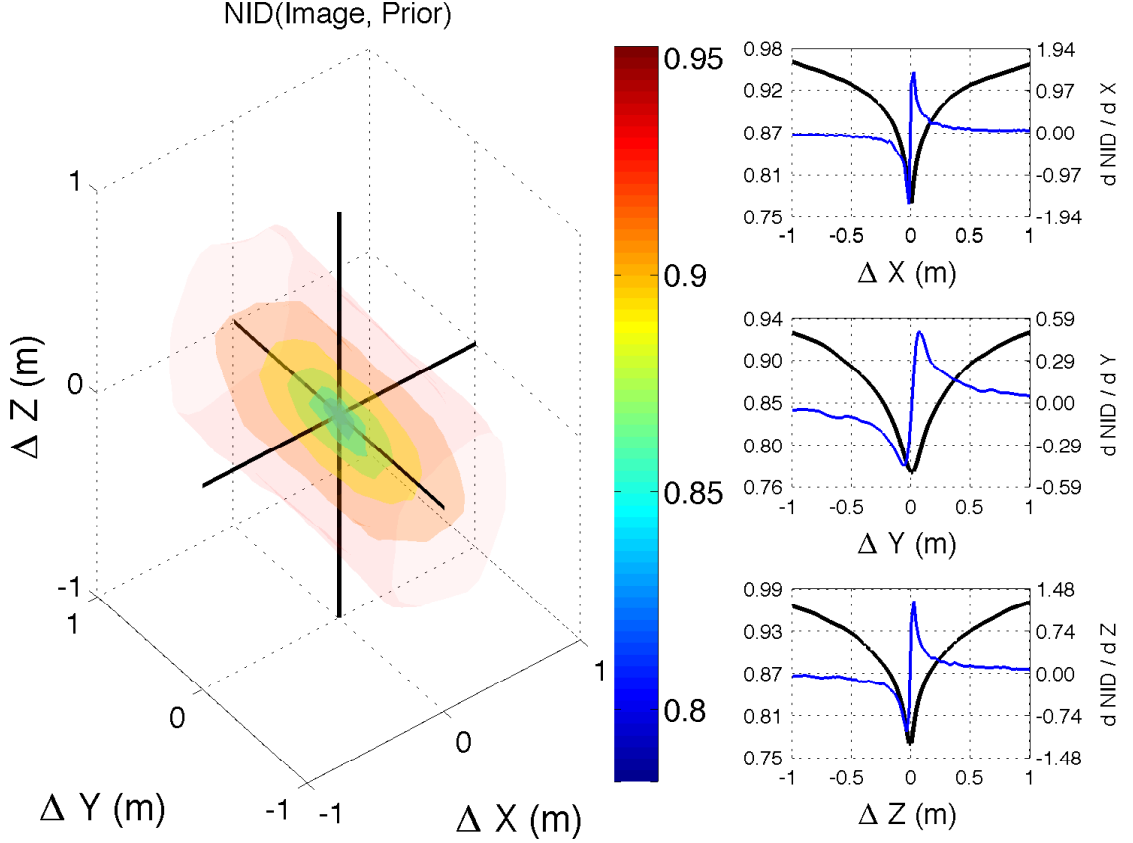


Figure 4.9: Iso-surface cost volume of NID for offsets in XYZ about ground-truth camera pose for EPFL Dataset shown in Figure 4.8. The NID evaluated at: $\Delta G \cdot G_{R,W}^*$ where $G_{R,W}^*$ is the ground truth camera pose for the frustrum shown in red in Figure 4.8 and ΔG is an offset transform. The angular orientation components of ΔG are all kept at zero and the translation components are sampled evenly across ± 1 metre to create the volume shown on the left. The iso-surfaces have been inverse alpha-blended (higher costs are more transparent) for ease of illustration. The subplots on the right show the evolution of the NID along the corresponding black sample lines aligned with each axis in the iso-surface cost volume. Also shown in blue on the subplots are the analytical gradients of NID evaluated along each axis, the evaluation of which is the subject of Chapter 5.

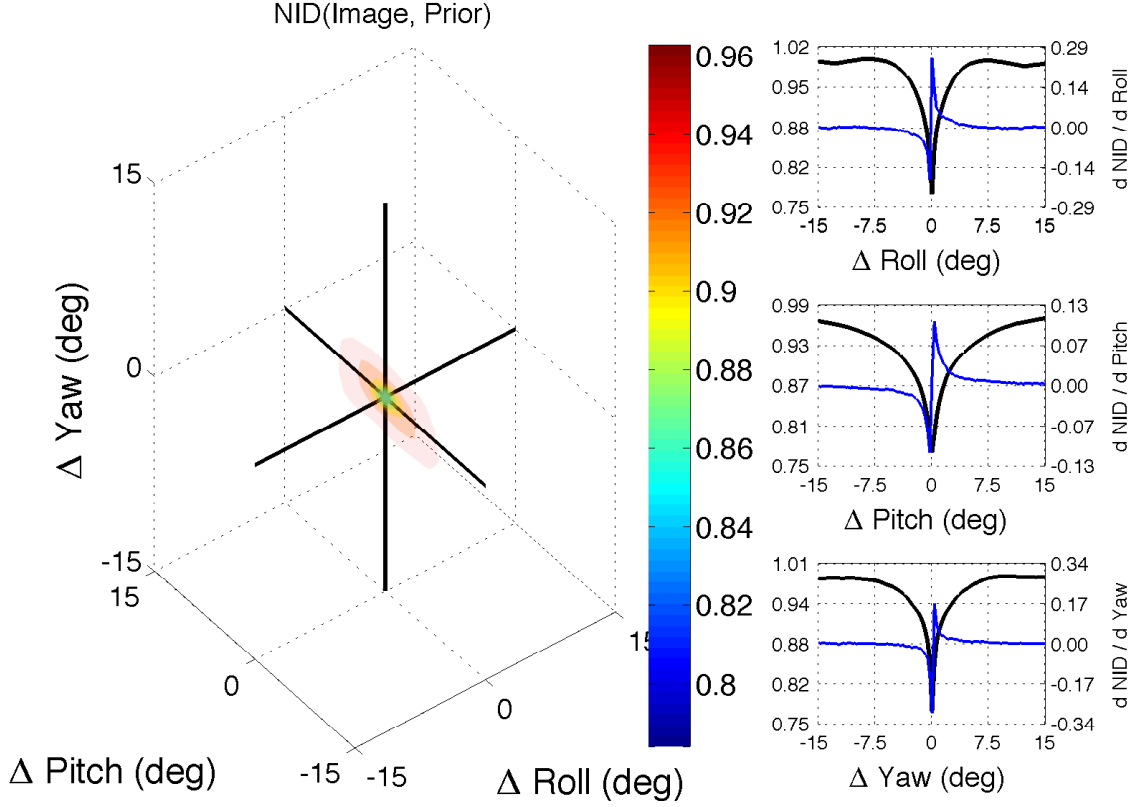


Figure 4.10: Iso-surface cost volume of NID for offsets in Roll, Pitch & Yaw about ground-truth camera pose for EPFL Dataset shown in Figure 4.8. The NID evaluated at: $\Delta G \cdot G_{R,W}^*$ where $G_{R,W}^*$ is the ground truth camera pose for the frustum shown in red in Figure 4.8 and ΔG is an offset transform. The XYZ translation components of ΔG are all kept at zero and the angular orientation components are sampled evenly across ± 15 degrees to create the volume shown on the left. The iso-surfaces have been inverse alpha-blended (higher costs are more transparent) for ease of illustration. The subplots on the right show the evolution of the NID along the corresponding black sample lines aligned with each axis in the iso-surface cost volume. Also shown in blue on the subplots are the analytical gradients of NID evaluated along each axis, the evaluation of which is the subject of Chapter 5.

tion (Figure 4.12), adding occlusions (Figure 4.13) and modifying its colourspace (Figure 4.14); all whilst keeping the appearance prior constant. For each case, we present what we term the *axes-sampled lines*, which are simply the NID evaluated for offsets along each Euler parameterised axis independently about the ground truth pose, as shown in the subplots on the right-hand sides of Figures 4.9 and 4.10.

Figure 4.11 shows the axes-sampled lines for the original (unmodified) image, with the translation terms on the left and the orientation terms on the right. Thus these are in fact exactly equal to the subplots on the right-hand sides of Figures 4.9 and 4.10, but simply presented in a more compact form. These should be used as the baseline for comparison for the mutated image cases shown in Figures 4.12 to 4.14. Note that the ground-truth pose, which is where the minima should occur, corresponds to the center (horizontally) of the plots, ie at an offset of zero.

In Figure 4.12, the right side of the image has been significantly brightened, in Figure 4.13 artificial textures have been added to the image that are not present in the prior and in Figure 4.14 the colourspace has been nonlinearly transformed into an illumination invariant space. In all cases, although the value of the NID at the minima is higher than in the unmodified case in Figure 4.11, the shape is broadly unchanged in all cases, particularly in Figures 4.12 and 4.13. The minima is still clear in all cases and occurs at ground-truth (zero offset) as desired, thus demonstrating that the robustness of the NID is indeed transferred to our pose estimation formulation, an important benefit of the proposed approach.

On the left hand side of Figures 4.11 to 4.14, under the modified images, are the joint appearance distributions p_{IS} , for the following cases: 20° (max) yaw offset (left), with the offsets for all other axes equal to zero; and ground-truth (right), i.e. zero offset for all axes. The colormap for both has been chosen such that zero corresponds to white, to highlight the spread of the distributions. The key points to note from these figures are that, as expected, p_{IS} is notably ‘peakier’, i.e. less

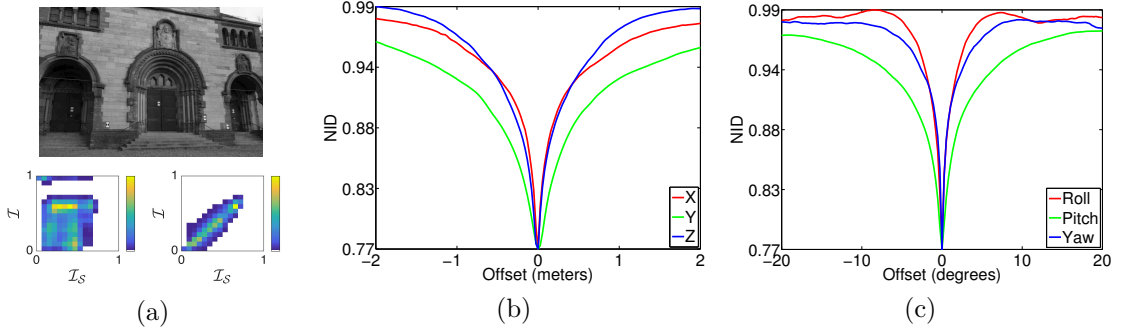


Figure 4.11: Axes-sampled lines for original (unmodified) image. NID evaluated for offsets in each axis independently about ground truth using the original image. On the left under the image, are the joint appearance distributions: p_{IS} for the 20° yaw offset (left), with the offsets for all other axes equal to zero; and ground-truth (right), i.e. zero offset for all axes.

uniform, in the ground-truth case; and that at ground-truth p_{IS} is *not* required to be a diagonal distribution.

Finally, we present a concrete example of the superiority of the NID over the Variation of Information in our formulation due to the equivalence problem discussed in Section 4.2.2. Figure 4.15 shows a camera looking down a straight road and the associated scene prior. In Figure 4.15c the VI and NID for pitch offsets about the ground-truth pose are shown in red and blue respectively. Note that although in the vicinity of the minima at ground-truth their shapes are similar, as the pitch offset increases the VI can reduce below its value at ground-truth, implying a better match. By comparison, the NID continues to increase to an approximately steady state value that is significantly higher than that at ground-truth. This occurs because when the camera is pitched, more points project either into the sky or onto the road. Either way, the resulting marginal appearance distribution from the image (p_I) becomes more homogeneous, thus reducing its entropy. Hence even though the mutual information has also decreased, the VI can still decrease below its value at ground-truth; this is exactly the failure case shown in Section 4.2.2.

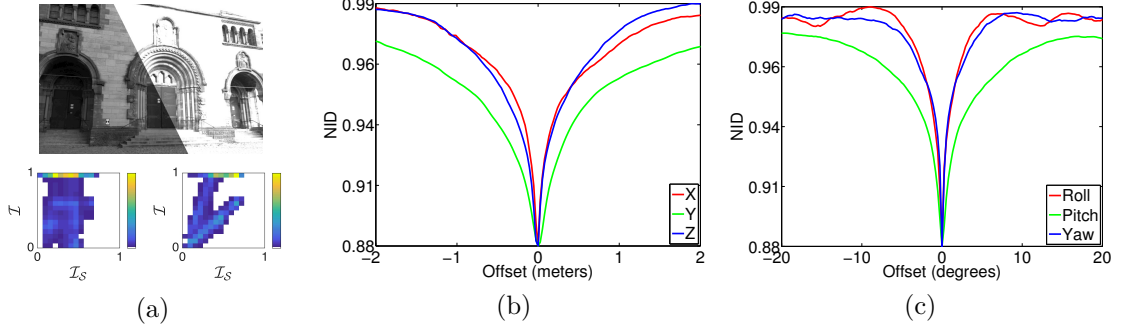


Figure 4.12: Axes-sampled lines for image where the right half has been artificially brightened. NID evaluated for offsets in each axis independently about ground truth using an image in which the right half has been made artificially brighter by a simple scaling. On the left under the modified image, are the joint appearance distributions: p_{IS} for the 20° yaw offset (left), with the offsets for all other axes equal to zero; and ground-truth (right), i.e. zero offset for all axes.

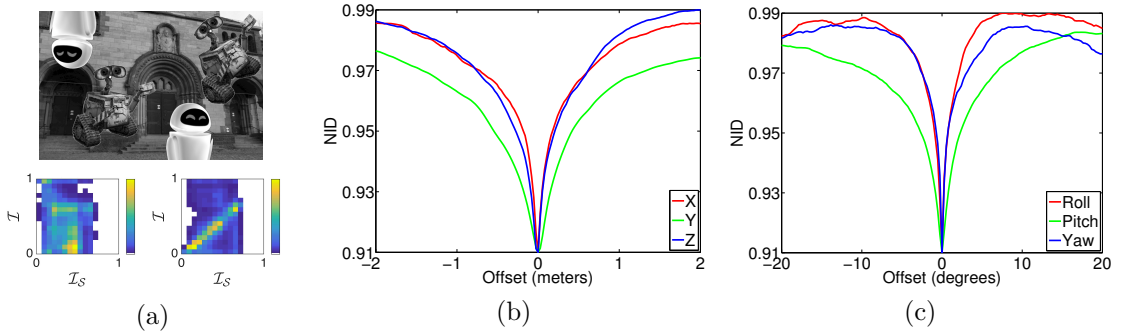


Figure 4.13: Axes-sampled lines for image with artificially added occlusions not present in the prior. NID evaluated for offsets in each axis independently about ground truth using an image to which artificial occlusions have been added which are not present in the prior appearance model. On the left under the modified image, are the joint appearance distributions: p_{IS} for the 20° yaw offset (left), with the offsets for all other axes equal to zero; and ground-truth (right), i.e. zero offset for all axes.

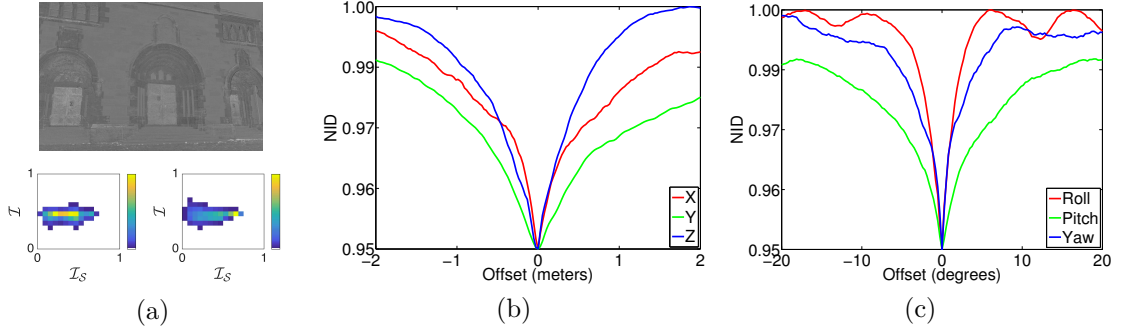


Figure 4.14: Axes-sampled lines for image after application of the nonlinear illumination invariant image transform. NID evaluated for offsets in each axis independently about ground truth using an image which has had its colour space mapped to an illumination invariant space via a nonlinear transform [57, 55]. The prior model colour space remains unchanged as the gray values computed via the standard linear RGB to Gray transform [6]. On the left under the modified image, are the joint appearance distributions: $p_{\mathcal{I}_S}$ for the 20° yaw offset (left), with the offsets for all other axes equal to zero; and ground-truth (right), i.e. zero offset for all axes.

4.5 Conclusions

In this chapter, we first reviewed the fundamental concepts of entropy and related quantities from the field of information theory. Then we discussed the concept of metric spaces for entropy measures, what they signify and why they are useful. We reviewed in detail two important information metrics, the Variation of Information and the Normalised Information Distance showing their relation, but also why the NID is superior despite both being metrics. Finally, we showed how the NID can be used as the objective function for the problem formulation introduced in Chapter 3 and presented some sample results demonstrating the desirable properties of the resulting formulation on a real dataset.

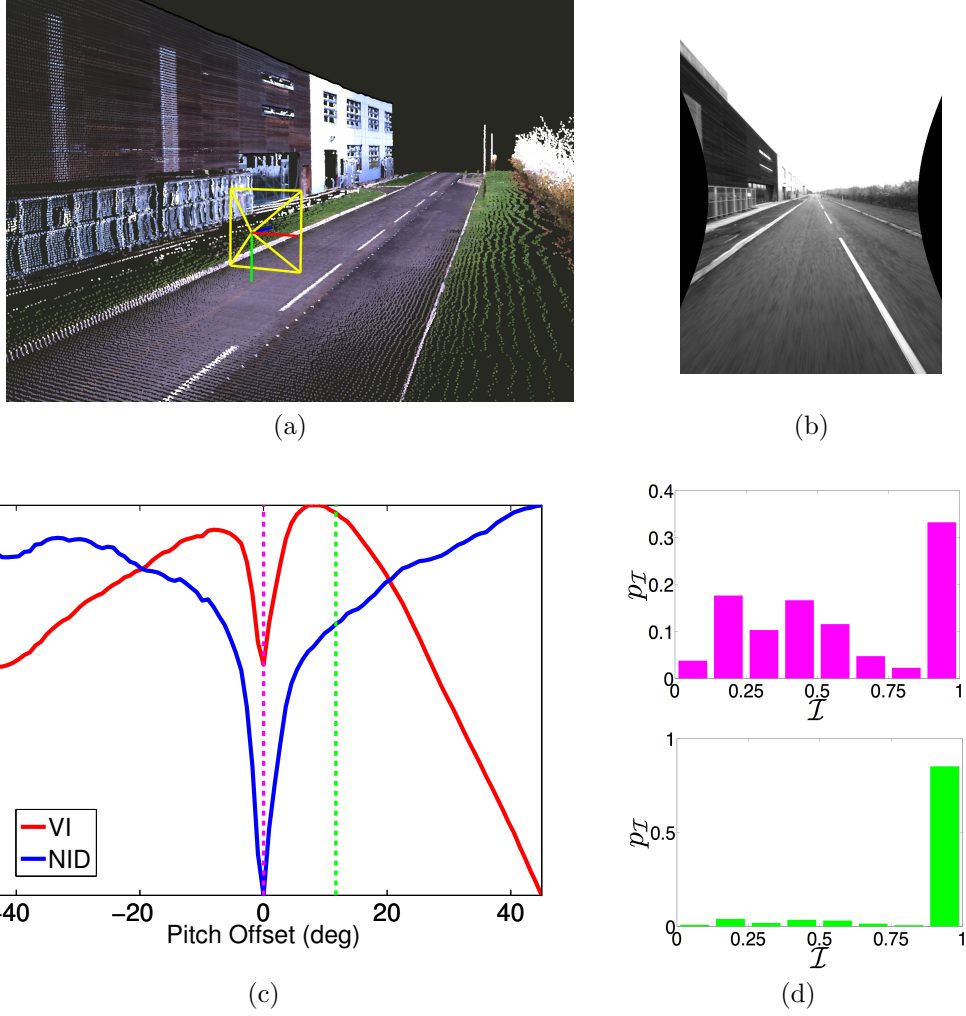


Figure 4.15: Effect of information content bias in the Variation of Information (VI) vs NID. As noted in Section 4.2.2, the VI is *biased* by the total information content and thus suffers from the equivalence problem highlighted in Figure 4.6. For our problem formulation this has significant practical consequences, as the sky is typically overexposed, reflecting our focus on the scene, not the sky. In the problem shown, if the true camera pose is offset by a downward pitch: $\Delta G_{\tilde{R},R} \cdot G_{R,W}$ where $\Delta G_{\tilde{R},R}$ parameterises an offset pitch, which by virtue of our transform definition (Section 2.1) and the direction of the transform, is positive. Then a large portion of the scene (Figure 4.15a) is projected into the sky in the live image (Figure 4.15b). Thus the intensity distribution p_I becomes dominated by the bin containing the intensity of the (white) sky reducing its overall entropy, as shown in Figure 4.15d. Where the magenta PDF corresponds to the correct camera pose: $G_{R,W}$ and green corresponds to the offset camera pose: $\Delta G_{\tilde{R},R} \cdot G_{R,W}$, this is exactly a manifestation of the case shown in Figure 4.6. Figure 4.15c shows the shape of the VI and the NID for different pitch offsets. As is evident, the VI is not a good metric as large pitch errors result in values less than at ground truth. By contrast, the NID is much more suitably behaved.

Chapter 5

Formulation of Optimisation

In section Section 4.4, we introduced the minimisation given in Equation (4.27)

$$\begin{aligned} & \arg \min_{G_{R,W}} \sum_c^{\text{Cameras}} \text{NID}(\mathcal{I}_c(\bar{\mathcal{S}}_c), \mathcal{I}_S(\bar{\mathcal{S}}_c)) \\ &= \arg \min_{G_{R,W}} \sum_c^{\text{Cameras}} \text{NID}(\mathcal{I}_c(\mathbf{x}), \mathcal{I}_S(\mathbf{q}) \mid \forall \mathbf{q} \in \bar{\mathcal{S}}_c) \end{aligned} \quad (4.27)$$

that we seek to solve for the $\mathbb{SE}(3)$ pose of the robot: $G_{R,W}$, from a set of synchronised images captured from cameras mounted on the robot. We now consider how this optimisation can be performed in practice. As a standard summation is used to combine the NID for each camera in Equation (4.27), we shall concern ourselves in this section solely with the computation of the NID for a single camera. This is because the same equations will be used separately for all cameras and their resulting combination, including that of their derivatives, is trivial.

An important thing to consider when choosing an optimisation approach is coupling in the parameter space of the optimisation. The optimisation we seek to solve is parameterised by the pose of the robot: $G_{R,W}$, which is an $\mathbb{SE}(3)$ transform that can be minimally represented by 3 translation and 3 rotation parameters, although

typically a quaternion with 4 rotation parameters is used to eliminate the singularity.

Although these components will, when considered independently have different effects on the pose of the robot, their combined effect on the NID will be coupled as shown in Figure 5.1. This is because for a single camera, the effect of a small error in one parameter of $G_{R,W}$ can be almost completely offset by error in a different parameter or parameters. In Figure 5.1, where the camera looks almost perpendicular to the direction of travel of the robot, the effect on the positions of the reprojected points in the camera for a small positive error in X in the robot frame will be mostly nullified by a small positive error in Yaw, as the resulting viewpoint of the camera will be approximately unchanged. This is represented in Figure 5.1 by the elongated diagonal (relative to the axes) shape of the contours of the minima.

The use of multiple cameras with complementary viewpoints will mitigate this effect, as the coupling would not be the same for each camera. However, as the couplings are a function of the camera parameters, its placement on the robot and the scene geometry & appearance, it should be assumed that some couplings will remain.

The presence of these couplings implies a second order optimisation method should be used to capture them and so achieve good performance. A steepest gradient descent algorithm would not be expected to work well. Similarly, algorithms which do not use gradient information, such as the Nelder-Mead Simplex method would typically be slower to converge than alternative methods that make use of at least the function gradient.

Second order methods are all variants on Newton’s method and perform a parameter step update of the form shown in Equation (5.1),

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \cdot H_k^{-1} \nabla f_k \quad (5.1)$$

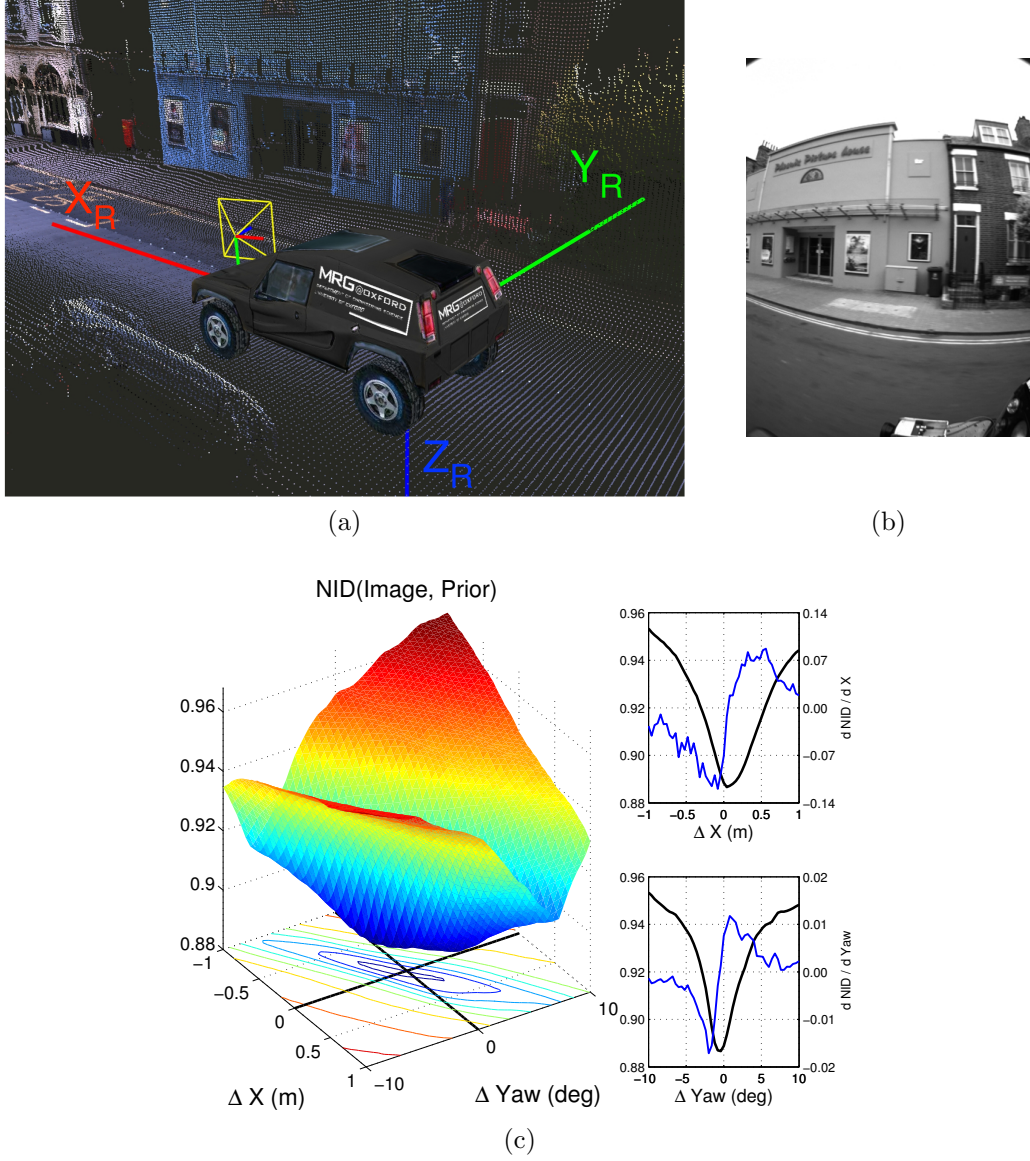


Figure 5.1: Effect of coupling between the robot pose parameters on NID. Shown in Figure 5.1a is the Wildcat robot for a typical scene with its base coordinate frame (X -forward, Y -right, Z -down), and a single camera attached to the front of the robot for which the frustum and camera coordinate frame (Z -forward, X -right, Y -down) are also shown. Figure 5.1c shows the NID surface for the single camera shown in Figure 5.1a and the captured image in Figure 5.1b for deltas in X & Yaw in the robot frame applied about the ground truth pose. The subplots in Figure 5.1c show the NID (black lines) and its gradient (blue lines) evaluated for deltas applied to the X & Yaw axes independently, i.e. the evaluation of the surface moving along the respective axes.

where $\mathbf{x}$ denotes the parameter vector, ∇f_k is the gradient matrix of the objective evaluated at $\mathbf{x}_k$ and H_k is the Hessian approximation. For Newton's method itself, H_k is the true Hessian, this method is not often used as the expense of computing the Hessian is typically very significant and thus computing it at each iteration is impractical.

The Levenberg-Marquardt (LM) algorithm is a variant on Newton's method for solving nonlinear least squares objectives, although as any minimisation of a non-negative objective f can be formulated as a nonlinear least squares minimisation of f^2 , this does limit its application. In LM, the Hessian is approximated using only the first order terms: $H_k = (J_k^T \Omega_k J_k + \lambda_k I)$, where J_k and Ω_k denote the Jacobian matrix and the inverse covariance matrix of the residuals respectively, and $\lambda_k > 0$ is a scalar. For large values of λ_k , the diagonal of H_k dominates and the method approximates gradient descent, conversely for small values of λ_k the method approximates Gauss-Newton. This allows the method to converge even when starting far away from the optimum where $J_k^T \Omega_k J_k$ results in a poor Hessian approximation, or may be rank deficient and thus not invertible, a prerequisite for Gauss-Newton.

Finally, in Quasi-Newton methods, the most prominent of which is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) [79] method discussed in Section 2.3, H_k represents an iteratively refined approximation of the Hessian updated using only the gradient at each iteration. These methods are explicit line search methods in that at each iteration once a descent direction is selected using the current value of H_k , a 1D line search is performed in that direction to select an α_k that minimises f sufficiently.

All of these methods require at least the gradient of the objective to be readily computable at each iteration and in the case of Newton's method, the Hessian too. They have also all been used in the related area of mutual information based

image alignment [60, 24, 17]. Before we discuss the relative merits of each approach for our problem in Section 5.4.5, we must first discuss how the derivatives of NID can be calculated given that it depends upon the joint appearance distribution $p_{\mathcal{IS}}$ constructed at each iteration. We will now show how the derivatives can be readily computed provided that the contribution of each point to $p_{\mathcal{IS}}$ is differentiable and thus $p_{\mathcal{IS}}$ is itself differentiable. We will also discuss and contrast the two common approaches to achieving this goal.

5.1 Derivatives of NID

5.1.1 Gradient of NID

Starting from the definition of NID given by Equation (4.12), the partial derivative with respect to a parameter ϕ , such as the roll from $G_{R,W}$, can be obtained by the quotient rule, as shown in Equation (5.2). Throughout this chapter, we will use following nomenclature substitutions for brevity: $H(X, Y) = H_{XY}$, $I(X; Y) = I_{X;Y}$ and $p_{XY}(x, y) = p_{XY}$ and equivalently for the marginal distributions.

$$\begin{aligned} NID(X, Y) &= \frac{H(X | Y) + H(Y | X)}{H(X, Y)} \\ &= \frac{H(X, Y) - I(X; Y)}{H(X, Y)} \end{aligned} \quad (4.12)$$

$$\frac{\partial NID(X, Y)}{\partial \phi} = \frac{\left(\frac{\partial H_{XY}}{\partial \phi} - \frac{\partial I_{X;Y}}{\partial \phi} \right) H_{XY} - (H_{XY} - I_{X;Y}) \frac{\partial H_{XY}}{\partial \phi}}{H_{XY}^2} \quad (5.2)$$

Our goal is now to obtain expressions for the partial derivative terms in Equation (5.2) which involve only the joint and marginal distributions and the derivative

of the joint distribution. Following the approach of [115, 24], considering the definition of the joint entropy H_{XY} in Equation (4.3)

$$H(X, Y) \equiv \sum_{x, y} -p_{XY} \log(p_{XY}) \quad (4.3)$$

and now assuming that the joint distribution p_{XY} is a function of the parameter vector ϕ , the partial derivative of H_{XY} is given by Equation (5.3):

$$\begin{aligned} \frac{\partial H_{XY}}{\partial \phi} &= - \sum_{x, y} \frac{\partial (p_{XY} \log(p_{XY}))}{\partial \phi} \\ &= - \sum_{x, y} \left(\frac{\partial p_{XY}}{\partial \phi} \log(p_{XY}) + p_{XY} \frac{\partial \log(p_{XY})}{\partial \phi} \right) \\ &= - \sum_{x, y} \left(\frac{\partial p_{XY}}{\partial \phi} \log(p_{XY}) + p_{XY} \frac{1}{p_{XY}} \frac{\partial p_{XY}}{\partial \phi} \right) \\ \frac{\partial H_{XY}}{\partial \phi} &= - \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi} (1 + \log(p_{XY})) \end{aligned} \quad (5.3)$$

Note that for a scalar ϕ , $\frac{\partial p_{XY}}{\partial \phi}$ is matrix of the same dimensions as p_{XY} , where each element is the partial derivative with respect to ϕ of the corresponding element in p_{XY} . Correspondingly, if ϕ is a vector, then $\frac{\partial p_{XY}}{\partial \phi}$ is a third order tensor. Intuitively, this is simply a vertical stack of: $|\phi|$ standard 2D matrices where the i -th matrix in the stack is the partial derivative of p_{XY} with respect to the i -th element of ϕ as in the previous scalar case.

The derivation for the partial derivative of $I_{X:Y}$ follows the same approach as for H_{XY} . First we rearrange the definition of $I_{X:Y}$ given in Equation (4.6) to obtain an expression in terms of the distributions given in Equation (5.4),

$$I_{X;Y} \equiv H_X + H_Y - H_{XY} \quad (4.6)$$

$$\begin{aligned} &= \sum_x p_X \log \left(\frac{1}{p_X} \right) + \sum_y p_Y \log \left(\frac{1}{p_Y} \right) + \sum_{x, y} p_{XY} \log(p_{XY}) \\ &= \sum_{x, y} \left(p_{XY} \log \left(\frac{1}{p_X} \right) + p_{XY} \log \left(\frac{1}{p_Y} \right) + p_{XY} \log(p_{XY}) \right) \\ &= \sum_{x, y} p_{XY} \log \left(\frac{p_{XY}}{p_X \cdot p_Y} \right) \end{aligned} \quad (5.4)$$

where we have used the identities: $-\log(x) = \log(\frac{1}{x})$, and $\sum_x p_{XY} = p_Y$ and equivalently for p_X .

At this point we will make a simplifying assumption about the form of X & Y , namely that $X = X(\phi)$, but that $Y \perp \phi$. In the context of our problem, this arises because the appearance of the prior is independent of the pose of the live camera. It is important to emphasise that this is not a fundamental requirement, in that the partial derivative of $I_{X;Y}$ can be computed using the same approach even if both X & Y were functions of ϕ and the form is similar, but with some extra terms.

Taking the derivative of the term inside the summation from Equation (5.4) with respect to ϕ yields Equation (5.5), where we have explicitly utilised the assumption that $Y \perp \phi$ to pull p_Y outside of the derivative.

$$\begin{aligned}
 & \frac{\partial}{\partial \phi} p_{XY} \log \left(\frac{p_{XY}}{p_X \cdot p_Y} \right) \\
 &= \frac{\partial p_{XY}}{\partial \phi} \log \left(\frac{p_{XY}}{p_X \cdot p_Y} \right) + p_{XY} \frac{\partial}{\partial \phi} \log \left(\frac{p_{XY}}{p_X \cdot p_Y} \right) \\
 &= \dots + p_{XY} \frac{p_X \cdot p_Y}{p_{XY}} \cdot \frac{\partial}{\partial \phi} \frac{p_{XY}}{p_X \cdot p_Y} \\
 &= \dots + p_{XY} \frac{p_X \cdot p_Y}{p_{XY}} \frac{1}{p_Y} \cdot \frac{\partial}{\partial \phi} \frac{p_{XY}}{p_X} \\
 &= \frac{\partial p_{XY}}{\partial \phi} \log \left(\frac{p_{XY}}{p_X \cdot p_Y} \right) + p_X \cdot \frac{\partial}{\partial \phi} \frac{p_{XY}}{p_X} \tag{5.5}
 \end{aligned}$$

$$\begin{aligned}
 &= \dots + p_X \cdot \frac{\frac{\partial p_{XY}}{\partial \phi} p_X - p_{XY} \frac{\partial p_X}{\partial \phi}}{p_X^2} \\
 &= \dots + p_X \frac{1}{p_X} \left(\frac{\partial p_{XY}}{\partial \phi} - \frac{p_{XY}}{p_X} \frac{\partial p_X}{\partial \phi} \right) \\
 &= \frac{\partial p_{XY}}{\partial \phi} \left(1 + \log \left(\frac{p_{XY}}{p_X \cdot p_Y} \right) \right) - \frac{p_{XY}}{p_X} \frac{\partial p_X}{\partial \phi} \tag{5.6}
 \end{aligned}$$

The right hand term of Equation (5.5) can now be simplified using the quotient rule to Equation (5.6). Using Equation (5.6), we can now write the full expression for the partial derivative of $I_{X;Y}$ as Equation (5.7).

$$\frac{\partial I_{X;Y}}{\partial \phi} = \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi} \left(1 + \log \left(\frac{p_{XY}}{p_X \cdot p_Y} \right) \right) - \sum_{x, y} \frac{p_{XY}}{p_X} \frac{\partial p_X}{\partial \phi} \tag{5.7}$$

Considering now the second summation term, we can show that this is in fact identically zero, as shown in Equation (5.8).

$$\begin{aligned}
 \sum_{x, y} \frac{p_{XY}}{p_X} \frac{\partial p_X}{\partial \phi} &= \sum_x \frac{1}{p_X} \cdot \frac{\partial p_X}{\partial \phi} \sum_y p_{XY} \\
 \text{But: } \sum_y p_{XY} &= p_X \\
 \Rightarrow \sum_{x, y} \frac{p_{XY}}{p_X} \frac{\partial p_X}{\partial \phi} &= \sum_x \frac{1}{p_X} \cdot \frac{\partial p_X}{\partial \phi} \cdot p_X \\
 &= \frac{\partial}{\partial \phi} \sum_x p_X = \frac{\partial}{\partial \phi} 1 = 0
 \end{aligned} \tag{5.8}$$

Thus we can further simplify Equation (5.7) to Equation (5.9). Finally, we show that the second summation in Equation (5.9) is in fact also identically zero as shown in Equation (5.10).

$$\begin{aligned}
 \frac{\partial I_{X;Y}}{\partial \phi} &= \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi} \left(1 + \log \left(\frac{p_{XY}}{p_X \cdot p_Y} \right) \right) \\
 &= \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi} \left(1 + \log \left(\frac{p_{XY}}{p_X} \right) - \log(p_Y) \right) \\
 &= \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi} \left(1 + \log \left(\frac{p_{XY}}{p_X} \right) \right) - \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi} \log(p_Y)
 \end{aligned} \tag{5.9}$$

$$\begin{aligned}
 \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi} \log(p_Y) &= \sum_y \log(p_Y) \sum_x \frac{\partial p_{XY}}{\partial \phi} \\
 &= \sum_y \log(p_Y) \frac{\partial}{\partial \phi} \sum_x p_{XY} \\
 \text{But: } \sum_x p_{XY} &= p_Y, \quad \frac{\partial p_Y}{\partial \phi} \equiv 0 \\
 \Rightarrow \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi} \log(p_Y) &= \sum_y \log(p_Y) \cdot 0 = 0
 \end{aligned} \tag{5.10}$$

Thus we can finally simplify Equation (5.9) to arrive at the final expression for the partial derivative of $I_{X;Y}$ given in Equation (5.11).

$$\frac{\partial I_{X;Y}}{\partial \phi} = \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi} \left(1 + \log \left(\frac{p_{XY}}{p_X} \right) \right) \quad (5.11)$$

Now we have expressions for the partial derivative terms in Equation (5.2) solely in terms of the marginal & joint distributions and the partial derivative of the joint distribution. How the partial derivative of the joint distribution can be evaluated is something we shall discuss in detail in Sections 5.2 and 5.3, after we first discuss a first order approximation to the NID Hessian.

5.1.2 Hessian of NID

Considering the equation for the gradient of the NID given in Equation (5.2), differentiating again yields the Hessian for NID given in Equation (5.13).

$$\frac{\partial NID(X, Y)}{\partial \phi_i} = \frac{\overbrace{\left(\frac{\partial H_{XY}}{\partial \phi_i} - \frac{\partial I_{X;Y}}{\partial \phi_i} \right)}^A H_{XY} - \overbrace{(H_{XY} - I_{X;Y})}^B \frac{\partial H_{XY}}{\partial \phi_i}}{H_{XY}^2} \quad (5.2)$$

$$\frac{\partial^2 NID(X, Y)}{\partial \phi_i \partial \phi_j} = \frac{\left(\frac{\partial A}{\partial \phi_j} - \frac{\partial B}{\partial \phi_j} \right) H_{XY}^{\cancel{2}} - (A - B) \cdot 2 \cancel{H_{XY}} \frac{\partial H_{XY}}{\partial \phi_j}}{H_{XY}^{\cancel{3}}} \quad (5.12)$$

$$= \frac{\left(\frac{\partial A}{\partial \phi_j} - \frac{\partial B}{\partial \phi_j} \right) H_{XY} - 2(A - B) \frac{\partial H_{XY}}{\partial \phi_j}}{H_{XY}^3} \quad (5.13)$$

The partial derivatives of the placeholders: A & B in Equation (5.13) are given in Equations (5.14) and (5.15) respectively.

$$\frac{\partial A}{\partial \phi_j} = \left(\frac{\partial^2 H_{XY}}{\partial \phi_i \partial \phi_j} - \frac{\partial^2 I_{X;Y}}{\partial \phi_i \partial \phi_j} \right) H_{XY} + \left(\frac{\partial H_{XY}}{\partial \phi_i} - \frac{\partial I_{X;Y}}{\partial \phi_i} \right) \frac{\partial H_{XY}}{\partial \phi_j} \quad (5.14)$$

$$\frac{\partial B}{\partial \phi_j} = \left(\frac{\partial H_{XY}}{\partial \phi_j} - \frac{\partial I_{X;Y}}{\partial \phi_j} \right) \frac{\partial H_{XY}}{\partial \phi_i} + (H_{XY} - I_{X;Y}) \frac{\partial^2 H_{XY}}{\partial \phi_i \partial \phi_j} \quad (5.15)$$

Analogously to the derivation of its gradient, the Hessian of H_{XY} used in Equations (5.14) and (5.15) can be derived from Equation (5.3) to obtain Equation (5.16) using the product rule.

$$\begin{aligned} \frac{\partial^2 H_{XY}}{\partial \phi_i \partial \phi_j} &= -\frac{\partial}{\partial \phi_j} \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi_i} (1 + \log(p_{XY})) \\ &= -\sum_{x, y} \frac{\partial^2 p_{XY}}{\partial \phi_i \partial \phi_j} (1 + \log(p_{XY})) + \frac{\partial p_{XY}}{\partial \phi_i} \frac{\partial p_{XY}}{\partial \phi_j} \frac{1}{p_{XY}} \end{aligned} \quad (5.16)$$

Likewise, following [108] the Hessian of $I_{X;Y}$ can be computed from Equation (5.11) resulting in Equation (5.17).

$$\begin{aligned}
 \frac{\partial I_{X;Y}}{\partial \phi_i} &= \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi_i} \left(1 + \log \left(\frac{p_{XY}}{p_X} \right) \right) \\
 \frac{\partial^2 I_{X;Y}}{\partial \phi_i \partial \phi_j} &= \sum_{x, y} \frac{\partial^2 p_{XY}}{\partial \phi_i \partial \phi_j} \left(1 + \log \left(\frac{p_{XY}}{p_X} \right) \right) + \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi_i} \frac{p_X}{p_{XY}} \frac{\partial}{\partial \phi_j} \left(\frac{p_{XY}}{p_X} \right) \\
 &= \dots + \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi_i} \left(\frac{\partial p_{XY}}{\partial \phi_j} \frac{1}{p_{XY}} - \frac{1}{p_X} \frac{\partial p_X}{\partial \phi_j} \right) \\
 &= \dots + \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi_i} \frac{\partial p_{XY}}{\partial \phi_j} \frac{1}{p_{XY}} - \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi_i} \frac{\partial p_X}{\partial \phi_j} \frac{1}{p_X} \\
 &= \dots + \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi_i} \frac{\partial p_{XY}}{\partial \phi_j} \frac{1}{p_{XY}} - \sum_x \frac{\partial p_X}{\partial \phi_j} \frac{1}{p_X} \sum_y \frac{\partial p_{XY}}{\partial \phi_i} \\
 &= \dots + \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi_i} \frac{\partial p_{XY}}{\partial \phi_j} \frac{1}{p_{XY}} - \sum_x \frac{\partial p_X}{\partial \phi_j} \frac{1}{p_X} \frac{\partial p_X}{\partial \phi_i} \\
 &= \sum_{x, y} \frac{\partial^2 p_{XY}}{\partial \phi_i \partial \phi_j} \left(1 + \log \left(\frac{p_{XY}}{p_X} \right) \right) + \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi_i} \frac{\partial p_{XY}}{\partial \phi_j} \frac{1}{p_{XY}} - \sum_x \frac{\partial p_X}{\partial \phi_i} \frac{\partial p_X}{\partial \phi_j} \frac{1}{p_X}
 \end{aligned} \tag{5.17}$$

Clearly Equations (5.16) and (5.17) contain the Hessian of the joint distribution. Irrespective of how the joint distribution is estimated, this is in general going to be very expensive to compute. In the context of our problem, computing this term would require the Hessian of the projection of a point in 3D into the image with respect to the pose of the camera. This can be computed [45], but is going to be prohibitively expensive to compute and accumulate for large number of points, and so computing the full Hessian at each iteration is not practical.

A widely used approximation to Equation (5.17) used in mutual information based medical image alignment is to ignore the second order term requiring the Hessian of the joint distribution entirely [108, 24] resulting in Equation (5.18).

$$\frac{\partial^2 I_{X;Y}}{\partial \phi_i \partial \phi_j} \approx \frac{\partial^2 \tilde{I}_{X;Y}}{\partial \phi_i \partial \phi_j} = \sum_{x, y} \frac{\partial p_{XY}}{\partial \phi_i} \frac{\partial p_{XY}}{\partial \phi_j} \frac{1}{p_{XY}} - \sum_x \frac{\partial p_X}{\partial \phi_i} \frac{\partial p_X}{\partial \phi_j} \frac{1}{p_X} \tag{5.18}$$

However, as noted in [17], this is in fact a bad approximation as when maximising the mutual information. The Hessian should be negative at the maxima at convergence by definition, but using Equation (5.18) results in a positive Hessian at convergence. We shall present results in Section 5.3.4 which demonstrate that this problem also occurs for the NID if the second order terms are neglected in Equations (5.16) and (5.17).

5.2 Differentiable form of Discrete PDFs

Now that we have expressions for the partial derivative terms in Equations (5.2) and (5.13) solely in terms of the marginal & joint distributions and the partial derivatives of the joint distribution, the only thing preventing us from evaluating it is an expression for the partial derivatives of the joint distribution, which we shall now discuss. Note that as we can trivially obtain the derivative of the marginal distributions from joint through summation, we need only discuss the computation of the derivative of the joint.

Typically, given a set of samples from a discrete random variable, its distribution can be approximated by a normalised histogram. For a ‘vanilla’ histogram, each observation increments the count of the bin into which it falls, the histogram is then normalised by the total number of observations. This form of histogram is *not* differentiable.

In order for a histogram to be differentiable ($C^{\geq 1}$), the contribution of each observation must be differentiable. Assuming that the total contribution to the histogram of each observation is the same ($= \frac{1}{N}$ where N is the number of observations), then a practical consequence of this is that each observation must (at least potentially) update multiple bins in the histogram. If this were not the case, when an observation transitioned from one bin to another there would inevitably be a

discontinuity.

Formally, for a joint distribution p_{XY} , we can write the value of each bin in a normalised histogram as a summation over all observations as shown in Equation (5.19).

$$p_{XY}(x, y) = \frac{1}{N} \sum_{i=0}^N \beta_X(x, q_i) \cdot \beta_Y(y, q_i) \quad (5.19)$$

In the most general case, each of the N observations potentially updates all of the bins in the joint, with a weight determined by the product of the normalised weight functions $\beta_X()$ & $\beta_Y()$ in X & Y respectively. The weight functions $\beta()$ are normalised in that $\sum_{x=0}^{|X|} \beta_X(x, q) = 1 \ \forall q$ (and respectively for β_Y) and thus the product of weight functions is also normalised. If this property were not true, then the normalisation constant in Equation (5.19) would not be N , it would be the sum of all weight contributions for all points. For a ‘vanilla’ histogram, where only a single bin is incremented per observation $\beta() = \delta()$, the Kronecker delta function.

Differentiating Equation (5.19) with respect to a parameter ϕ yields another histogram, where the contribution from each observation to a given bin is the derivative of the product of the normalised weight functions as shown in Equation (5.20).

$$\frac{\partial p_{XY}(x, y)}{\partial \phi} = \frac{1}{N} \sum_{i=0}^N \frac{\partial}{\partial \phi} (\beta_X(x, q_i) \cdot \beta_Y(y, q_i)) \quad (5.20)$$

It is important to note that β_X & β_Y do not have to be the same function, as we shall discuss in Section 5.3.1. Under the assumption used in Section 5.1.1 that $X = X(\phi)$ and $Y \perp \phi$, Equation (5.20) simplifies to Equation (5.21).

$$\frac{\partial p_{XY}(x, y)}{\partial \phi} = \frac{1}{N} \sum_{i=0}^N \frac{\partial \beta_X(x, q_i)}{\partial \phi} \cdot \beta_Y(y, q_i) \quad (5.21)$$

Now that we have a general form for a discrete joint distribution which is differentiable, in the next section we will specialise it to our problem. Principally this concerns the choice of the normalised weight functions $\beta()$ and the space in which they operate. In addition to the full form of their derivative for our problem using the chain rule.

5.3 Joint Appearance Distribution

Recalling again our desired minimisation given in Equation (4.27),

$$\begin{aligned} & \arg \min_{G_{R,W}} \sum_c^{\text{Cameras}} \text{NID}(\mathcal{I}_c(\bar{\mathcal{S}}_c), \mathcal{I}_S(\bar{\mathcal{S}}_c)) \\ &= \arg \min_{G_{R,W}} \sum_c^{\text{Cameras}} \text{NID}(\mathcal{I}_c(\mathbf{x}), \mathcal{I}_S(\mathbf{q}) \mid \forall \mathbf{q} \in \bar{\mathcal{S}}_c) \end{aligned} \quad (4.27)$$

the joint appearance distribution $p_{\mathcal{IS}}$ that we need to construct for each camera, as shown graphically in Figure 4.7, takes as input: the appearance of the points from the prior $\mathcal{I}_S(\mathbf{q})$; the live image $\mathcal{I}_c$, and the reprojected coordinates for the visible points in the live image $\mathbf{x}$; from which their live appearance $\mathcal{I}_c(\mathbf{q})$ can be obtained via interpolation. As we require $p_{\mathcal{IS}}$ to be differentiable, it needs to be constructed as a weighted histogram as described in Section 5.2, where each point $\mathbf{q}$ is used to update, at least potentially, multiple bins with different weights.

There are two distinct choices of the space in which the normalised weight functions $\beta()$ introduced in Section 5.2 can be applied to build $p_{\mathcal{IS}}$: intensity and position. Both approaches have been used in the context of mutual information based image alignment, a widely used technique in medical imaging [59] where images from different modalities such as X-rays and fMRI need to be aligned where no clear functional mapping between the two input spaces exists. The former (intensity) is

often described as Parzen windowing [115, 24, 17, 86], whilst the latter (position) was introduced as partial volume interpolation in [58], but is now also known as partial volume estimation [24]. We use the terms intensity weighting and spatial weighting in this work, as although they are not widely used, they allow a clearer exposition as they highlight the key difference between the methods.

In the results presented in this work, we use spatial weighting to build the joint appearance distribution, the justifications for which are given in Section 5.3.5. However, as these two approaches are often confused as being exactly equivalent in the literature, we will first review in detail both approaches and present results comparing them for a 1D problem in Section 5.3.4.

Before reviewing these two approaches, it is important to stress that the fundamental distinction between them is the space in which $\beta()$ is applied, not its functional form which can be, and indeed often is, the same in both. Various choices for the functional form of $\beta()$ have been used, including linear interpolation [58], Gaussians [115] and most prominently B-Splines [59, 24, 17].

The principal requirements for $\beta()$ are that it have finite support as the extent of the p_{IS} is bounded, be differentiable and that its value and analytical derivative be fast to compute. As such, B-Splines [88] discussed in Section 2.2, are particularly well suited as they are fast to compute and differentiable (degree -1) times, e.g a cubic B-Spline is twice differentiable. In this work we shall only consider the case when $\beta()$ is chosen to be the cubic B-Spline basis functions due to their aforementioned benefits and because they have finite support, as at most 4 control points have non-zero weights.

5.3.1 Spatial Weighting

The use of what we term spatial weighting to build the joint appearance distribution was originally introduced as partial volume interpolation by [58], but is now also

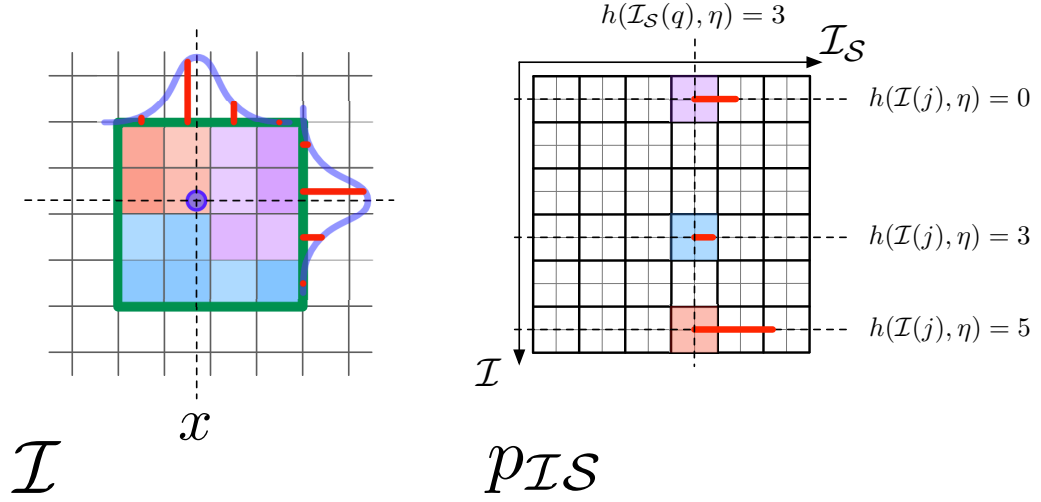


Figure 5.2: Spatial weighting joint appearance distribution update. The 3D point $\mathbf{q}$ reprojects to the coordinates $\mathbf{x}$ in the input image $\mathcal{I}$ shown on the left. The 4×4 patch of pixels bounding $\mathbf{x}$ that would have non-zero weight when calculating $\mathcal{I}(\mathbf{x})$ using cubic B-Spline interpolation (Equation (2.15)) is shown with a green border. For each pixel within the patch, the weight it would be assigned when computing $\mathcal{I}(\mathbf{x})$ using Equation (2.15) is the product of two univariate cubic B-Splines in reprojected image coordinates centered at $\mathbf{x}$ shown in blue on the left. This weight is then added to the bin containing the pixel's intensity in the column identified by the prior appearance $h(\mathcal{I}_S(\mathbf{q}, \eta))$ in the histogram $p_{\mathcal{I}S}$ shown on the right,. Where $h(x, \eta)$ is the histogramming function: $h(x, \eta) = \lfloor x \cdot \eta \rfloor$, η is the number of bins per axis in the histogram and we assume that all intensities have been normalised. Note that this means that the bins updated in the histogram for each point are *not* (necessarily) adjacent in the histogram and thus intensity. The number of bins updated per point is thus variable between 1 and $\min(16, \eta) = \eta = 6$ in the example shown.

known as partial volume estimation [24]. In this approach, the normalised weighting function $\beta() = \beta(\mathbf{x})$ is applied in image coordinates for the live image only, and is independent of the intensity of the reprojected point or surrounding pixels, as shown graphically in Figure 5.2.

To understand the intuition for this approach, consider the task of cubic B-Spline image interpolation discussed in Section 2.2.3. Assuming that the B-Spline interpolating image for $\mathcal{I}$: $\mathcal{I}^\beta$ has been computed, for example using the fast Z-transform technique described in [92], recall from Section 2.2.3 that the C^2 interpolated value

of the image at $\mathbf{x}$ is given by Equation (2.15),

$$\mathcal{I}(\mathbf{x}) = \sum_r \sum_c \mathcal{I}^\beta(r, c) \cdot \beta(x - c) \cdot \beta(y - r) \quad (2.15)$$

where only pixels in the 4×4 pixel patch in the image bounding $\mathbf{x}$ have non-zero weights. Also note that $\beta()$ in Equation (2.15) is independent of intensity, it depends solely upon position.

Spatial weighting is analogous to the image interpolation in Equation (2.15), for each point $\mathbf{q}_i$, the 4×4 pixel patch centered on $\mathbf{x}_i$ containing the pixels which have non-zero weight in Equation (2.15) is identified. Then, for each pixel within the 4×4 patch the bin in the column of the histogram identified by the reference point appearance $\mathcal{I}_S(\mathbf{q}_i)$ containing the *pixel's* intensity in $\mathcal{I}$ is increased by the weight the same pixel in $\mathcal{I}^\beta$ (i.e. at the same row-column position) would be assigned in Equation (2.15), i.e: $\beta(x - c) \cdot \beta(y - r)$.

The reason that only a single column in the histogram is updated by any point is that in spatial weighting, the weight function for the prior is the Kronecker delta function: δ_s . This is intuitive, as the prior represents an immutable reference which is independent of the live camera pose, and as the intensity value is all that remains (i.e. the image used to generate the intensity of the point in the prior is not available), no purely spatial weighting can be performed for $\mathcal{I}_S(\mathbf{q})$. This is an example of the case we highlighted in Section 5.2 where the weighting functions for X & Y , respectively the live and prior data, are not the same.

The resulting specialisation of Equation (5.19) for spatial weighting is given in Equation (5.22),

$$p_{\mathcal{IS}}(i, s) = \frac{1}{N} \sum_{j=0}^N \sum_{r=0}^{\text{Rows} \in \mathcal{I}} \sum_{c=0}^{\text{Cols} \in \mathcal{I}} \left(\beta_u(u_j - c) \cdot \beta_v(v_j - r) \cdot \delta_i(\lfloor \mathcal{I}(r, c) \cdot \eta \rfloor - i) \right) \cdot \delta_s(\lfloor \mathcal{IS}(\mathbf{q}_j) \cdot \eta \rfloor - s) \quad (5.22)$$

assuming as before that all intensities are normalised. It is important to note that although there are two weight functions, one for each of the two axes defining the image plane identified as u & v , where $\mathbf{x} \equiv [u, v]$ as stated earlier, both of them weight the contribution from the live image, not the prior. The weighting function for the prior is the Kronecker delta: δ_s .

The reason for the second Kronecker delta: δ_i , is that because $\beta()$ is now solely a function of position, there must be a selector to identify the bin to be updated in the column of the histogram identified by δ_s . The additional large brackets in Equation (5.22) group all of the terms constituting collectively the weight function for the live image to allow easier comparison with Equation (5.19). Finally, analogously to the case of cubic B-Spline image interpolation, although the inner summations in Equation (5.22) are over all rows and columns in $\mathcal{I}$, the use of cubic B-Spline weight functions means that their product is non-zero for at most a 4×4 pixel patch for each observation.

Note that in Equation (5.22) $\mathcal{I}^\beta$, the B-Spline interpolated image for $\mathcal{I}$, is not used. This is because no interpolation is actually being performed, i.e. no resulting interpolated value $\mathcal{I}(\mathbf{x})$ is being calculated. Instead, the weights that would be used for interpolation which are independent of intensity are being accumulated. Using $\mathcal{I}^\beta$ in place of $\mathcal{I}$ would have the disadvantage that the intensity information being accumulated would not be that of the captured image, and computing $\mathcal{I}^\beta$ imposes additional computational cost. Although, due to the use of NID as a comparison metric, and because the per-pixel difference between $\mathcal{I}^\beta$ and $\mathcal{I}$ is typically not large,

the effect of using $\mathcal{I}^\beta$ instead of $\mathcal{I}$ is generally imperceptible.

An important consequence of spatial weighting shown in Figure 5.2, is that the bins updated by a single observation are *not* (necessarily) adjacent in the histogram. Also the number of bins updated per point is not constant (excluding the boundaries), if the intensities of all bounding pixels fell into the same bin, then only a single bin would be updated. Conversely, if they all fell into different bins, a maximum of 16 bins would be updated. As such, this approach does not add any information that is not present in the image.

An additional major benefit of spatial weighting is that it can faithfully operate on images with edges and does not require smoothing of the image beforehand to ensure continuity; as shown graphically in Figure 5.3, which should be contrasted with Figure 5.5 for the intensity weighting case. This is possible because by applying $\beta()$ in position, not intensity space the resulting chain rule expression for the derivative of Equation (5.24) does not involve the image gradient, as shown in Equation (5.23),

$$\frac{\partial \beta(\mathbf{x}_i)}{\partial G_{R,W}} = \frac{\partial \beta}{\partial \mathbf{x}_i} \frac{\partial \mathbf{x}_i}{\partial G_{R,W}} \quad (5.23)$$

and thus the image gradient is not required when using spatial weighting.

A similar performance optimisation when computing the derivatives of $p_{\mathcal{I}\mathcal{S}}$ to that of [20] for the intensity weighting case discussed in Section 5.3.2 exists for the spatial weighting case. Any reprojected point for which the intensities of all of the pixels in the bounding 4×4 pixel patch fall into the same bin in the histogram should logically contribute zero to the derivatives and thus need not be considered when computing them.

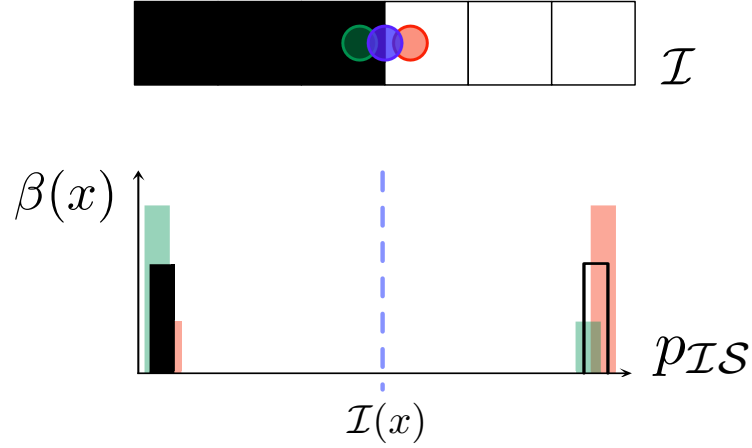


Figure 5.3: Spatial weighting update for a 1D image with an edge. When the sample point, shown in blue, is exactly half way along the image: $\mathcal{I}(x) = 0.5$ (assuming normalised intensities). However, the bins updated are those which contain the intensities of the bounding pixels, which are 0 & 1 respectively to the left and right. Thus the bins at opposite ends of the histogram are updated with equal weight. This accurately reflects that the image only contains black and white pixels. When the sample point is moved slightly to the left or right shown in green or red respectively, which bins are updated is unchanged (the bins at opposing ends) and their weights vary smoothly and inversely as shown, which accurately represents what is happening.

5.3.2 Intensity Weighting

The alternative to spatial weighting for building the joint appearance distribution histogram is what we term intensity weighting, where the normalised weighting function $\beta()$ is applied in intensity space in the histogram, as shown graphically in Figure 5.4.

For each point $\mathbf{q}$, first the sub-pixel interpolated intensity of the point $\mathcal{I}(\mathbf{x})$ is computed in the live image $\mathcal{I}$ using the reprojected point position $\mathbf{x}$, which is typically performed using bilinear interpolation. The interpolated intensity in the live image and $\mathcal{I}_S(\mathbf{q})$, the intensity of the point in the prior, are then used as coordinates in the histogram about which $\beta()$ is evaluated in each axis independently. Each bin in the histogram is then updated with the product of $\beta()$ evaluated in each axis at the intensity of the bin center.

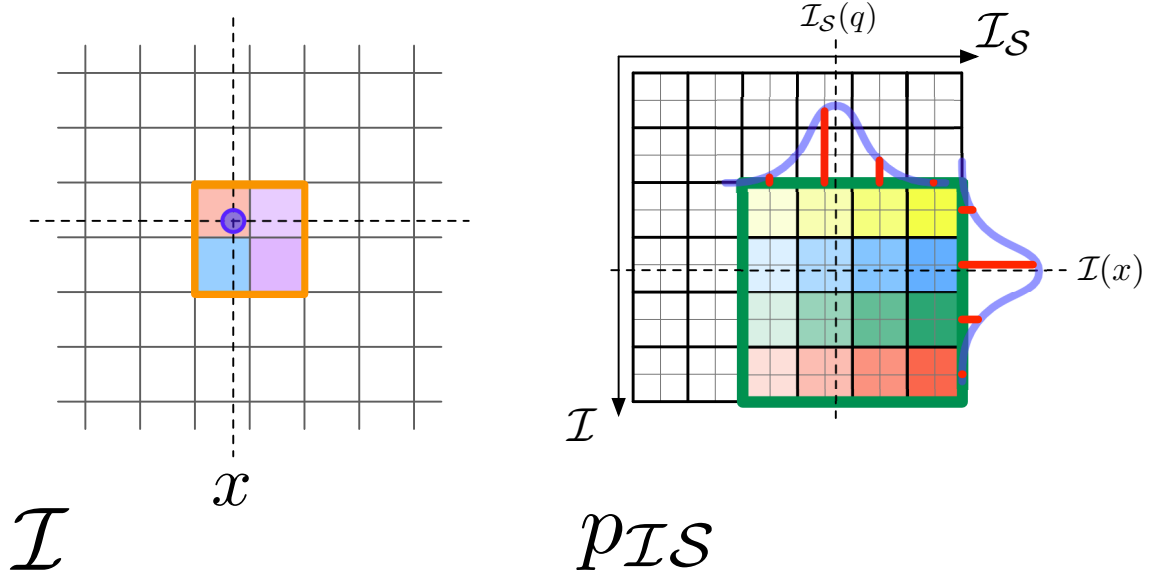


Figure 5.4: Intensity weighting joint appearance distribution update. The 3D point $\mathbf{q}$ reprojects to the coordinates $\mathbf{x}$ in the image $\mathcal{I}$ shown on the left. Using bilinear interpolation with the four bounding pixels highlighted by the orange border, the intensity of the image at $\mathbf{x}$ (the live appearance of the point) $\mathcal{I}(\mathbf{x})$ is estimated. Next, separate 1D cubic B-Spline weighting functions are centered at $\mathcal{I}(\mathbf{x})$ and $\mathcal{I}_S(\mathbf{q})$ in the *histogram* aligned with each axis, shown in purple on the right. As $\beta(\cdot)$ is the cubic B-Spline weighting function, only 4 adjacent bins in each axis have non-zero weights. The 16 adjacent bins updated by the observation are contained within the 4×4 bin patch centered on $\{\mathcal{I}(\mathbf{x}), \mathcal{I}_S(\mathbf{q})\}$ in the histogram highlighted by the green border. Each bin is updated with the product of the 1D B-Spline weighting function for each axis, evaluated at the bin centres (defined in intensity space). For each axis, the 1D B-Spline weighting function returns a weight that is proportional to the distance in intensity space between the appearance of the point in that axis ($\mathcal{I}(\mathbf{x})$ or $\mathcal{I}_S(\mathbf{q})$ respectively) and the centre of the bin in that axis, shown as red lines in the figure. It is important to note that the bins updated using this scheme do *not* (necessarily) correspond to pixel intensities that appear in the image.

The specialisation of the general differentiable histogram equation Equation (5.19) for intensity weighting is given in Equation (5.24),

$$p_{\mathcal{IS}}(i, s) = \frac{1}{N} \sum_{j=0}^N \beta(\mathcal{I}(\mathbf{x}_j) - i) \cdot \beta(\mathcal{I}_{\mathcal{S}}(\mathbf{q}_j) - s) \quad (5.24)$$

where we assume that all intensities are normalised and that η is the number of bins per side in the histogram. In this work we consider only square histograms where $\eta_X = \eta_Y = \eta$. Recall that although the summation is over all observations for each bin, when $\beta()$ is the cubic B-Spline basis function there are by definition a maximum of 4×4 bins with non-zero weights per observation, with fewer bins at the edges.

A consequence of this approach is shown in Figure 5.4, namely that in contrast to spatial weighting, it adds information that is not present in the original image. The bin updated with the highest weight (blue) contains the interpolated point appearance $\mathcal{I}(x_j)$. However, the image may not in fact have any pixels which would fall into the adjacent bins in intensity space which are updated, shown by their different colours. Conversely, the image may have pixels which bound sample points for which the bins containing their intensities are never updated.

This problem is particularly apparent when considering an edge, as shown in Figure 5.5 for a 1D image, which should be contrasted with Figure 5.3 for the spatial weighting case. The image contains only black and white pixels, but the intensity weighting scheme updates the bins about the central gray value, despite there being no pixels in the image with these values. Furthermore, if the sample point is moved slightly to the left or the right, the mass in the histogram shifts significantly to the left or right and in fact crosses the entire histogram for movements ≥ 0.5 pixels.

The choice of weighting in the intensity space means that the resulting chain rule expression for the derivative of Equation (5.24) must include the image gradient $\frac{\partial \mathcal{I}}{\partial \mathbf{x}}$

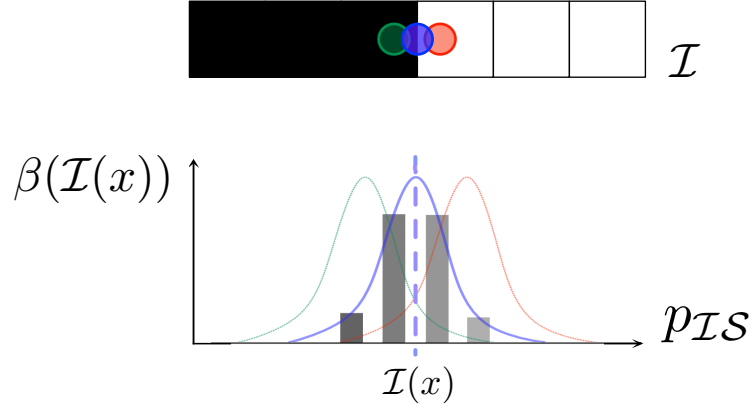


Figure 5.5: Intensity weighting update for a 1D image with an edge, contrast with Figure 5.3 for the spatial weighting case. When the sample point, shown in blue, is exactly half way along the image $\mathcal{I}(x) = 0.5$ (assuming normalised intensities), intensity weighting updates the four bins around the centre of the histogram as shown. This is despite there being *no* pixels in the image which would fall into the updated bins, in contrast to the spatial weighting case shown in Figure 5.3. Furthermore, if the reprojected point is moved slightly to the left or right shown in green or red respectively, the mass added to the histogram will move across the full width of the histogram.

as shown in Equation (5.25).

$$\frac{\partial \beta(\mathcal{I}(\mathbf{x}_i) - k)}{\partial G_{R,W}} = \frac{\partial \beta}{\partial \mathcal{I}} \frac{\partial \mathcal{I}}{\partial \mathbf{x}_i} \frac{\partial \mathbf{x}_i}{\partial G_{R,W}} \quad (5.25)$$

Although the cost of computing the image gradient is relatively small when using a Sobel filter, for typical images which have a large number of edges the resulting gradients are typically noisy. As the image gradient will be used in the chain rule expression to generate the derivatives, it is important that it be smooth to ensure a smooth overall result for the derivative of NID. Hence, when intensity weighting is used, the input images should be blurred, e.g. with a Gaussian filter [24, 17] before being used. Blurring the image smooths the image gradient estimate and reduces the likelihood of the situation shown in Figure 5.5. However, it does this by modifying the information in the image to match the output of the weighting technique, rather than modifying the technique to fit the data.

The use of bilinear interpolation to compute $\mathcal{I}(\mathbf{x})$ is common [24, 17] because it is fast, but the result is only C_0 . In practice, the smoothing of the image commonly performed for these methods mitigates the practical effect of this on the continuity of the resulting derivatives. In [86], quadratic B-Spline image interpolation was used instead, which would by virtue of being C_1 avoids this problem irrespective of whether the image was smoothed.

The presence of the image gradient in the chain rule is not however purely disadvantageous, as it enables an approximation which can improve the performance when computing the derivatives of p_{IS} as noted in [20]. Specifically, any points reprojecting into areas where the image gradient is small, i.e. areas with homogeneous intensity, at least over the window size used to compute $\mathcal{I}(\mathbf{x})$, will intuitively, contribute little to the derivative. In [20], this is used as the justification for using only those pixels in the image near strong edges when computing the derivatives of p_{IS} . As this is often a small subset of the total number of pixels in the image, the resulting derivative evaluations are faster.

It is also the case that intensity weighting in general has a greater dependence upon the choice of η , the number of bins per axis in the histogram, as it always updates the 4×4 patch of adjacent bins about the bin containing $\mathcal{I}(\mathbf{x})$ for each observation (assuming cubic B-Spline weighting). Hence as η changes, the range in intensity spanned by the updated bins changes inversely. For smaller values of η , this smooths the resulting NID signal but compresses its range as when every observation updates a larger portion of the histogram, it becomes more uniform. Thus the overall entropy, and hence the NID, increases at ground truth and so the relative change as a function of the error, i.e. the height of the minima, decreases. Results demonstrating this for some sample problems will be presented later in Section 5.3.4.

5.3.3 Point Projection Jacobians

Irrespective of whether intensity or spatial weighting is used, the chain rule expression for the gradient of the NID must contain the point projection Jacobian: $\frac{\partial \mathbf{x}_i}{\partial G_{R,W}}$, which describes the effect of changes in $G_{R,W}$ on the reprojected coordinates in the image plane $\mathbf{x}_i$, of a 3D point $\mathbf{q}_i$ defined in the camera frame. This can be separated using the chain rule into the product of the camera projection Jacobian and the point transformation Jacobian as shown in Equation (5.26).

$$\frac{\partial \mathbf{x}_i}{\partial G_{R,W}} = \frac{\partial \mathbf{x}_i}{\partial \mathbf{q}_i} \frac{\partial \mathbf{q}_i}{\partial G_{R,W}} \quad (5.26)$$

For the standard pinhole camera model [33], the projection of a 3D point $\mathbf{q} = [X, Y, Z]$ in the camera frame into its image in the image plane $\mathbf{x}$ is given by Equation (5.27),

$$\mathbf{x} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{f_x \cdot X}{Z} + c_x \\ \frac{f_y \cdot Y}{Z} + c_y \end{bmatrix} \quad (5.27)$$

where f_x & f_y and c_x & c_y represent respectively the focal lengths and principal point of the camera in X & Y. A skew parameter can also be added to handle cameras where the axes are not perpendicular, but for most cameras this is zero, and thus we neglect it here.

Although any real camera will have radial and tangential distortion, this can be estimated through a calibration procedure typically involving a checkerboard [123, 93]¹. Once the parameters defining the distortion have been estimated, a function to transform the distorted input image into an equivalent undistorted image with an associated distortion-free pinhole camera model can be generated. Typically this takes the form of a lookup table mapping integer pixel coordinates in the undistorted

¹<https://sites.google.com/site/scarabotix/ocamcalib-toolbox>

output image to subpixel coordinates in the input image. Throughout this work we will always assume that this procedure has been performed with the appropriate transformation of the input image, and so we can use the pinhole camera model.

Differentiating Equation (5.27) with respect to $\mathbf{q}_i$ yields the camera projection Jacobian given in Equation (5.28).

$$\frac{\partial \mathbf{x}_i}{\partial \mathbf{q}_i} = \frac{\partial}{\partial \mathbf{q}_i} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{f_x \cdot X}{Z^2} \\ 0 & \frac{f_y}{Z} & -\frac{f_y \cdot Y}{Z^2} \end{bmatrix} \quad (5.28)$$

When considering the point transformation Jacobian, it is important to remember that it is the pose of the *robot* to which the camera is attached that is being optimised, not the pose of the camera itself. Additionally, Equation (5.27) assumes that $\mathbf{q}$ is in the camera frame, but the points are natively stored in the world frame referenced by $G_{R,W}$. Hence the point transformation Jacobian can be represented as Equation (5.29), where $G_{C,R}$ is the extrinsic calibration of the camera with respect to the robot and ΔG is a small $\mathbb{SE}(3)$ delta transform representing a motion applied in the robot frame parameterised by $\Delta g \in \mathfrak{se}_3$ indexed by j . We have also explicitly added subscripts denoting the frame of reference to $\mathbf{q}$ for clarity. As discussed in Section 2.1, the derivative of an $\mathbb{SE}(3)$ transform with respect to the j -th component of its $\mathfrak{se}_3$ representation evaluated at zero is given by the j -th generator matrix: gen_j . Hence Equation (5.29) can be simplified to produce our final expression for the point transformation Jacobian given in Equation (5.30).

$$\frac{\partial \mathbf{q}_C}{\partial \mathbf{g}_{R,W}^j} = \frac{\partial (G_{C,R} \cdot \Delta G \cdot G_{R,W} \cdot \mathbf{q}_W)}{\partial \Delta \mathbf{g}^j} \quad (5.29)$$

$$\frac{\partial \mathbf{q}_C}{\partial \mathbf{g}_{R,W}^j} = G_{C,R} \cdot \underset{\mathbb{SE}(3)}{\text{gen}_j} \cdot G_{R,W} \cdot \mathbf{q}_W \quad (5.30)$$

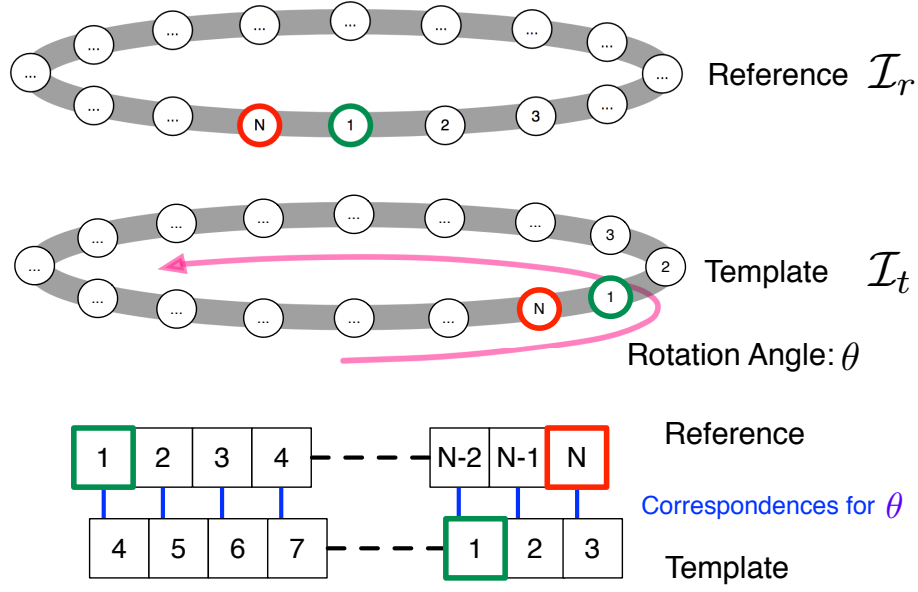


Figure 5.6: Problem setup for comparison of intensity and spatial histogram weighting methods. A 1D reference image $\mathcal{I}_r$ is represented as a ring, with N values equally spaced in azimuth. The template image $\mathcal{I}_t$, which has the same values as the reference (with the same spacing) is then rotated by an angle θ relative to the reference. As the images are modelled as rings, $\theta \in [-\pi, +\pi]$. For any given value of θ , the N correspondences used to build the histogram and compute the NID are the N reference values and the interpolated values in the reference, with the manner of this interpolation dependent upon the update method.

5.3.4 Results

Having discussed both weighting approaches in the previous sections, in this section we present some results that compare their performance. To simplify the problem we consider the case of aligning two 1D image rings by rotating one (the template) whilst keeping the other (the reference) fixed as shown in Figure 5.6. Each ring is marked with the same N values that are regularly spaced in azimuth and for any given offset rotation θ , N correspondences are used to update the histogram with interpolation so that θ is not constrained to be an integer multiple of the separation of the values in azimuth. As the rings are marked with the same values, $\theta = 0$ should correspond to perfect alignment (self with self).

For each example, the NID and its gradient as a function of θ calculated using

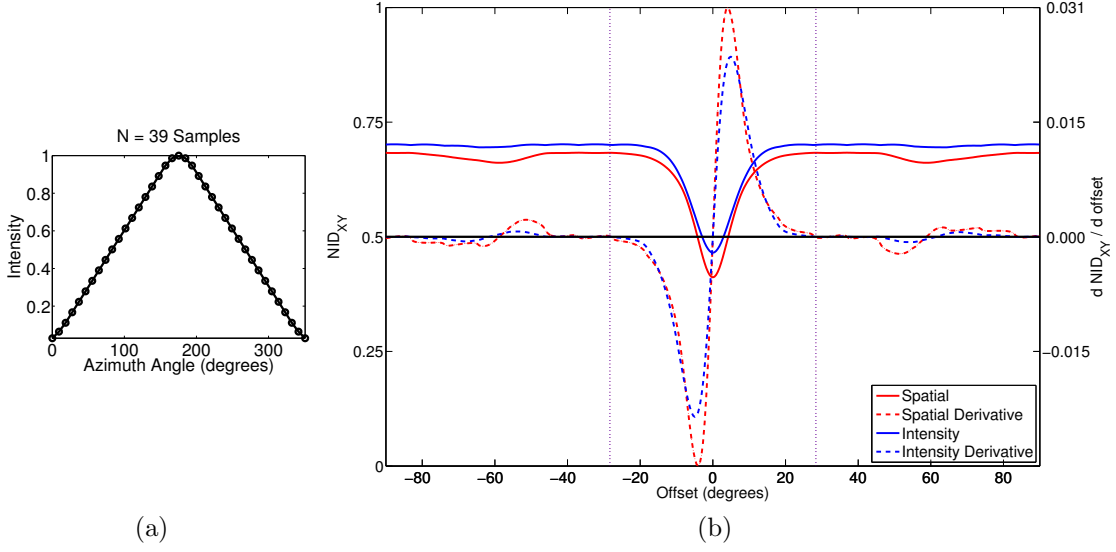


Figure 5.7: Comparison of spatial and intensity weighting histogram update methods for a simple triangular signal image shown on the left.

both intensity weighting and spatial weighting are shown, in addition to the 1D image wrapped around the ring, which was filtered with a 5 pixel wide Gaussian filter with $\sigma = 1$ as required by the intensity weighting method. The image gradients required for the intensity weighting method were computed using the central difference method. The vertical dashed lines on each plot denote the first zero crossings of the NID derivative measured outwards from the origin ($\theta = 0$, no offset).

Figure 5.7 shows the simple case of a triangular pattern using $\eta = 16$. For this case, the two methods result in nearly equivalent results, except that as noted in Section 5.3.2 the intensity method produces a slightly shallower peak.

The NID is still relatively high at $\theta = 0$ (ground-truth): ≈ 0.4 for spatial and ≈ 0.46 for intensity, for related reasons. In the spatial case, because there are relatively few samples, a moderate number of bins combined with the choice of pattern mean that the bounding pixels in the reference for each observation are in different bins. As such, every point is updating multiple bins the histogram, even at ground-truth, thus increasing the entropy and thus the NID. In the intensity case,

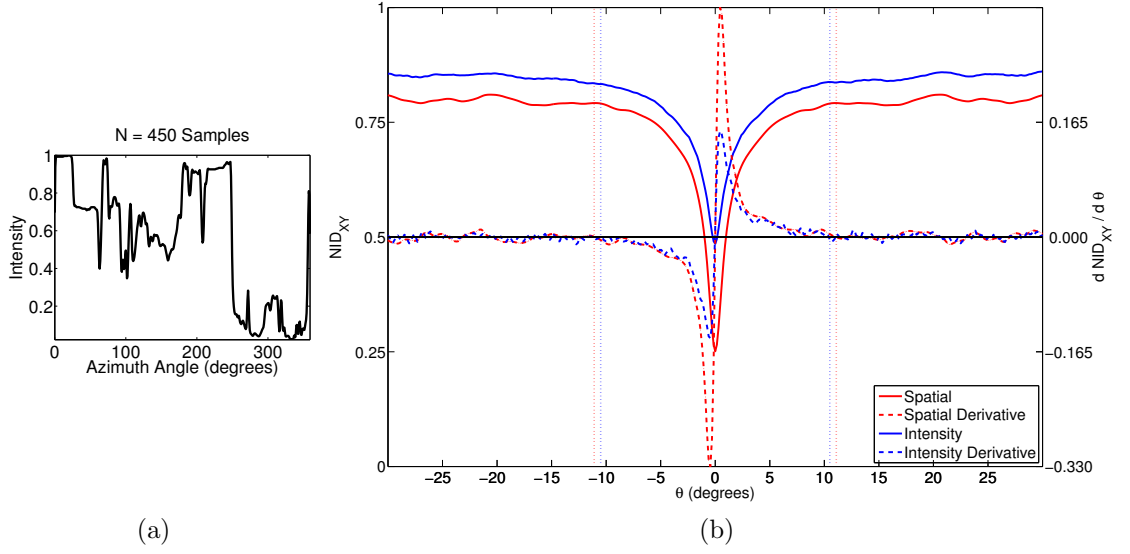


Figure 5.8: Comparison of spatial and intensity weighting histogram update methods for an arbitrary row from a real image, shown on the left.

η relative to the number of bins updated per observation (4) is the distinguishing factor. As the number of bins decreases, the minimum attainable NID increases, as the number of bins updated per observation is fixed as shown in Figure 5.9.

The triangle example from Figure 5.7 is decidedly unlike real image data, which will typically not have smooth periodicity and will have a large number of edges. Figure 5.8 uses instead an arbitrarily selected row from a real image. As can be seen from the figure, both methods again produce very similar results, although for a more realistic signal, the spatial case reports a significantly lower NID at ground truth when using the same $\eta = 16$.

Figure 5.9 shows the effect of changing η , the number of bins per side in the histogram, using the same image row from Figure 5.8. Although the location and shape of the minima is essentially the same for both methods, the spatial method is much more invariant to changes in η than the intensity method. As η becomes large, high frequency noise becomes evident in the intensity case, particularly on the gradient estimate, whilst the spatial method remains smooth.

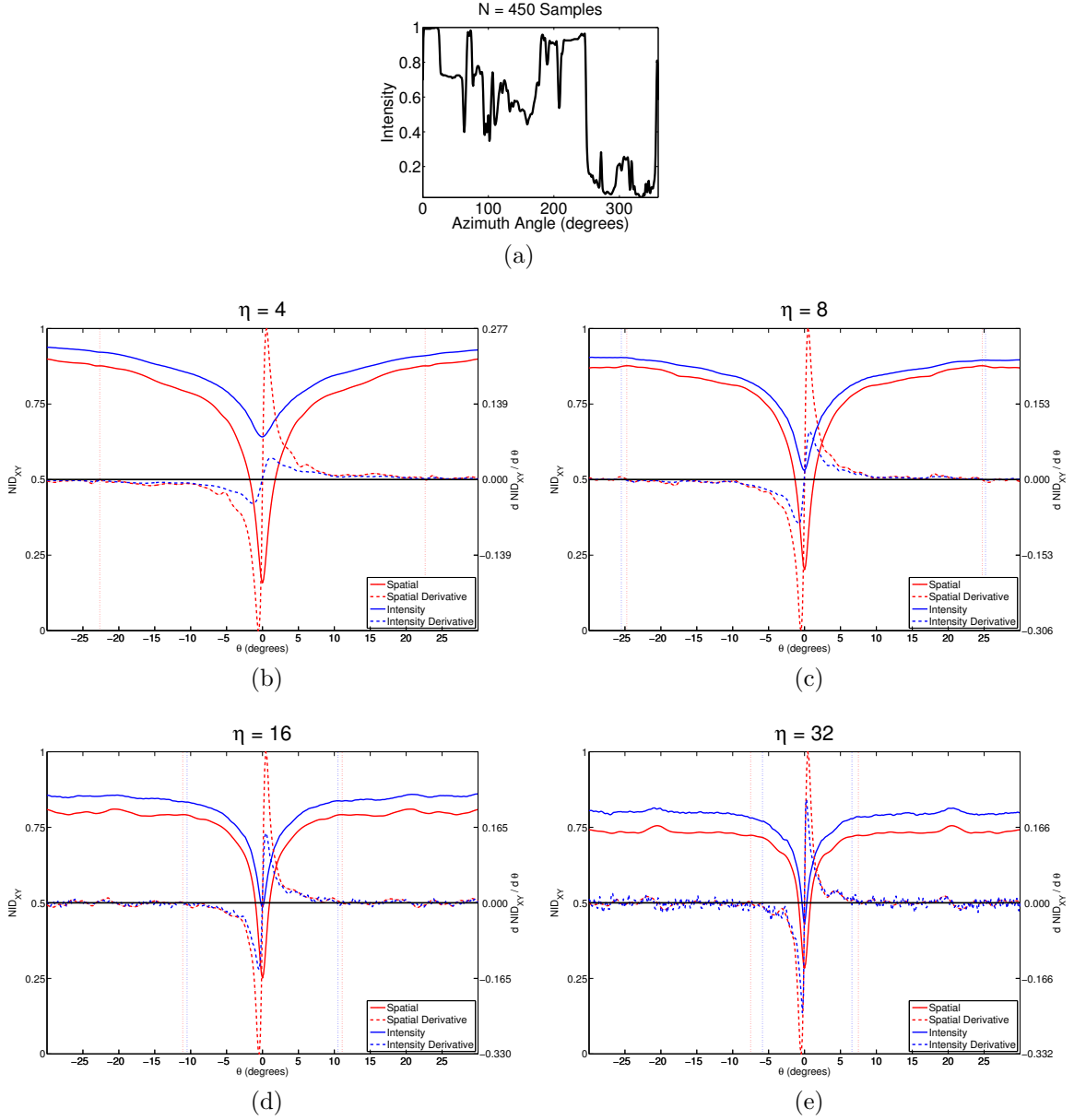


Figure 5.9: Effect of η , the number of bins per side in the histogram, on the NID and its gradient for both the spatial and intensity weighting update methods for an arbitrary row from a real image show at the top. Although the location and shape of the minima is broadly the same for both methods, the spatial method is much more invariant to changes in η than the intensity method. As η becomes large, high frequency noise becomes evident in the intensity case, particularly on the gradient estimate, whilst the spatial method remains smooth.

The primary considerations for choosing η are that large histograms are more expensive to compute and $\eta = 2^m : m \in \mathbb{Z}_{>0}$ and a uniform bin width are desirable for fast implementations. If a non-uniform bin width was desired, this could be equivalently achieved using uniform bin width by pre-warping the data by the inverse of the desired non-uniform bin width profile. In the context of mutual information based medical image alignment where images in different modalities (fMRI, CT, PET etc) are being aligned, typically with relatively small initial errors, large values: $\eta = 100 - 256$ are common [59]. However for mutual information based visual servoing with typical grayscale images, much smaller values: $\eta = 8$ have been used successfully [17, 18] and are significantly more attractive for real-time operation.

As discussed in Section 5.1.2, a common approximation for the Hessian used in mutual information based image alignment is to neglect the second order terms which results in a Hessian approximation that has the wrong sign at convergence. If the 1st order approximated Hessians for H_{XY} & $I_{X;Y}$ are used in Equation (5.13) to compute the NID Hessian, the same problem occurs irrespective of the weighting method used, as shown in Figure 5.10. As the NID has a minima at convergence, the Hessian should be positive by definition, but as can be seen in Figure 5.10c, the first order approximate Hessian is in fact negative (but small) at convergence. It should also be noted that the range of angles in θ for which the numerical Hessian is positive about $\theta = 0$ is very small: ≈ 1.2 degrees. This would be a severe constraint for a pure Newton's method solver, as it would only converge in this region when using the numerical Hessian, and would never converge using the approximate analytical Hessian.

5.3.5 Choice of Weighting Method

Based on the results presented in the previous section, we elected to investigate spatial weighting in this work. The method is theoretically attractive, as it does

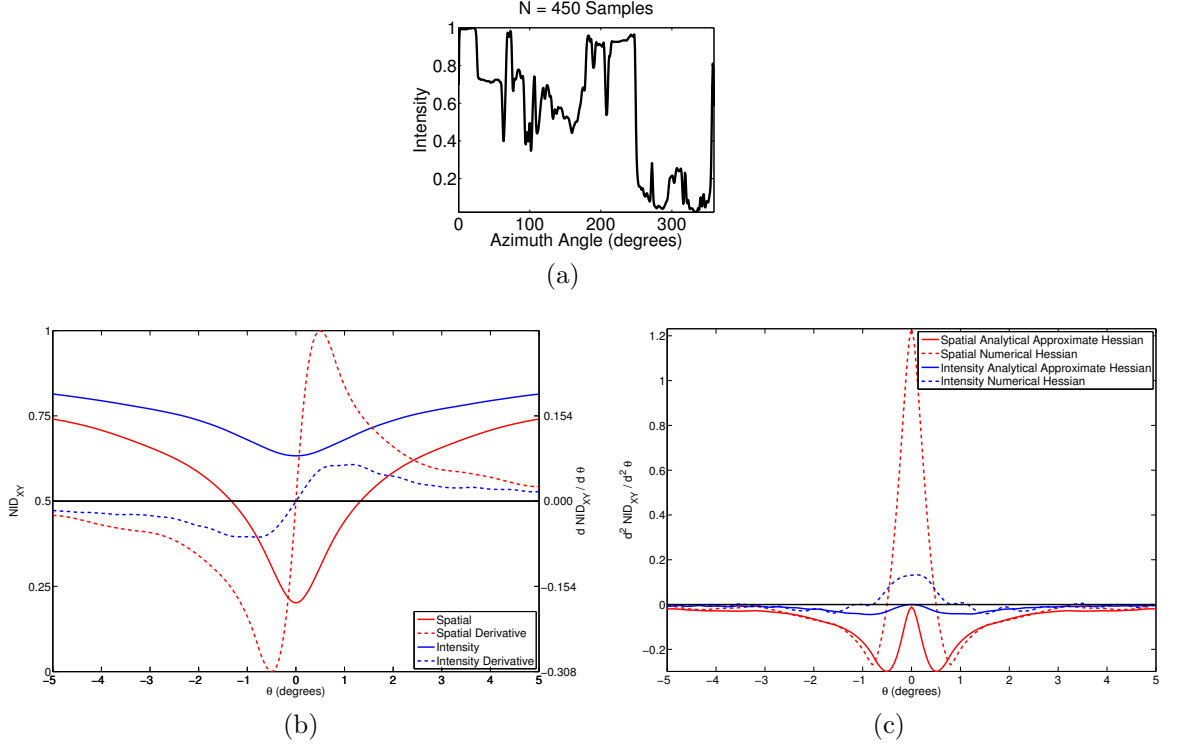


Figure 5.10: Inaccuracy of 1st order approximation to the NID Hessian. As discussed in Section 5.1.2, a common approximation for the Hessian used in mutual information based image alignment is to neglect the second order terms, as noted in [17], this yields a Hessian approximation that has the wrong sign at convergence. If the 1st order approximated Hessians for H_{XY} & $I_{X;Y}$ are used in Equation (5.13) to compute the NID Hessian, the same problem occurs irrespective of the weighting method used. Figure 5.10c shows the approximate NID Hessian versus the numerical Hessian obtained by numerically differentiating the gradient from Figure 5.10b for the arbitrary row from a real image shown in Figure 5.10a. As NID has a minima at convergence, the Hessian should be positive by definition, as the numerical estimates are, however the 1st order analytical approximate Hessian is negative (but small) at convergence.

not add any information which is not already present in the image and can more faithfully represent edges, which are common in real images. Furthermore, it has a greater dynamic range and is more robust to changes in η . In addition to not requiring the use of the image gradient, which necessitates smoothing of the image, the degree of which is another parameter to be tuned and which also mutates the information in the image.

An ancillary benefit of spatial weighting is that it updates a variable number of bins per observation and so can be more performant when implemented in parallel, particularly on GPUs, where the principal bottleneck is the necessarily atomic bin updates. As such, even though the maximum number of bins updated per observation is the same as the number updated in the intensity weighting case (16 using cubic B-Spline weighting functions), in practice for our problem formulation using typical images and values of η (number of bins per dimension) of 8-16, a large number of points update fewer than 4 bins in the spatial case, and few update more. By contrast, the intensity weighting approach always updates 16 bins per observation, except for samples near the boundaries of the histogram and so typically would have to perform more atomic updates per observation resulting in slower overall performance. An additional consideration is the cost of computing and then querying for each point the gradient image.

As mentioned earlier, in this work we will only consider square histograms: $\eta_x = \eta_y = \eta$ with uniform bin widths. This is both desirable for a high performance implementation and also ensures that the range of the information content of the prior and live image are equal, rather than assuming that one intrinsically contains more information than the other. Finally as stated earlier, if non-uniform bin widths were desired, a similar effect can be achieved using uniform bin widths by pre-warping the input data by the inverse of the desired non-uniform bin width profile.

5.4 Optimising NID

Now that we have discussed the methods by which the gradient and the Hessian of NID can be computed, we will return to the question of what optimisation algorithm we will use to solve our objective for the pose of the robot. First, we shall review various optimisation choices from the related field of mutual information based image alignment and visual servoing, then we shall present our choice of optimiser.

Mutual information based image alignment is a widely used technique in medical imaging [89, 59] and a wide variety of optimisation approaches have been tried in the literature for this problem and the related problem of mutual information based visual servoing. Broadly, these approaches can be divided into four groups: Levenberg-Marquardt (LM), Newton's Method, Quasi-Newton and Sampling methods, this is not a perfect categorisation, as Powell's method and Conjugate Gradient methods in particular are not natural fits for one of these groups, however they are also less widely used in the literature.

As mutual information based alignment is a maximisation, not a minimisation problem like our objective using the NID and indeed most pose estimation algorithms, we will assume throughout this section that the mutual information objective is: $f_{I_{X;Y}} = \max(I_{X;Y}) - I_{X;Y}$ such that we can talk purely in terms of a minimisation problem. As the mutual information is bounded and this bound can be easily precomputed and is independent of distributions of X & Y as it depends solely on η (Chapter 4), this does not affect the result and removes a potential source of confusion.

5.4.1 Levenberg-Marquardt

As discussed earlier, the standard Levenberg-Marquardt method for nonlinear least squares problems has an update of the form given in Equation (5.31),

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \cdot (J_k^T \Omega_k J_k + \lambda_k I)^{-1} \nabla f_k \quad (5.31)$$

where the Hessian is approximated as: $(J_k^T \Omega_k J_k + \lambda_k I)$ using the Jacobian J_k and inverse covariance matrix of the residuals Ω_k .

As $f_{I_{X,Y}}$ and NID are non-negative functions, they could theoretically be solved by minimising their square using standard LM. However, in general LM is used to solve problems where: $|f| \geq |\mathbf{x}|$, i.e. there are more residuals than parameters, this does not on its own imply that the problem is overdetermined, but it is a necessary prerequisite and this is certainly the ideal situation. As $|f_{I_{X,Y}}| = |NID| = 1$, but we are interested in estimating an SE(3) pose and in medical imaging their problem is often parameterised by an equivalent Affine warp, we have: $|f| < |\mathbf{x}|$. In this situation with single residual, it can be shown that LM is equivalent to a scaled gradient descent, as shown in Appendix A, and so will generally not perform well. We note this explicitly, as in some of the literature [60] this approach is used without a realisation that it equates to scaled gradient descent in this case.

The technique first presented in [108] to address this shortcoming, is to replace Equation (5.31) with Equation (5.32),

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\frac{\partial^2 \tilde{I}_{X,Y}}{\partial \mathbf{x}^2} + \lambda_k I \right)^{-1} \nabla f_k \quad (5.32)$$

where $J_k^T \Omega_k J_k$ is replaced with the approximate Hessian obtained using only the first order terms given in Equation (5.18). This mimics the desirable behaviour of LM, in that it alternates between Gradient Descent and an approximation to Newton's method based on the choice of λ_k without requiring any additional terms beyond

those required for the gradient to be computed.

This method has since become widely used in medical imaging [24] and has also been used in the context of mutual information based object tracking [86]. However, as discussed in Section 5.1.2, the first order approximation of the Hessian has the wrong sign at (at least) convergence, implying that this approach would be problematic. That this approach works in light of this problem is due to the presence of the $\lambda_k I$ term, which ensures that the full Hessian approximation used by LM is always positive definite.

5.4.2 Newton’s Method

In the context of visual tracking, a variation on Newton’s method was first presented in [16] and has subsequently also been used for visual servoing [17, 19]. The intuition for this approach, is that in both cases the target image is known, and at the desired convergence condition, the template and the reference should be perfectly aligned. Thus, the Hessian at convergence $H^* = \frac{\partial^2 I_{X;Y}^*}{\partial \mathbf{x}^2}$ can be approximated by computing the full Hessian from Equation (5.17) including the second order terms, by using the reference image as both the reference and the template. This Hessian is then held constant throughout the optimisation, which uses a standard Newton’s method step, resulting in Equation (5.33).

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\frac{\partial^2 I_{X;Y}^*}{\partial \mathbf{x}^2} \right)^{-1} \nabla f_k \quad (5.33)$$

By evaluating the full Hessian keeping all second order terms, the problem identified earlier whereby the approximate Hessian has the wrong sign at convergence is avoided. As the Hessian is evaluated at convergence, the speed of convergence when $\mathbf{x}$ is near $\mathbf{x}^*$ should be high. Keeping the Hessian constant throughout the optimisation ensures that the convergence basin for the optimiser should match that

of gradient descent, rather than the narrower region in which the Hessian is positive. As only the reference image is required to compute H^* , it can be precomputed in advance offline, thus allowing real-time performance online, as computing the full Hessian is an expensive operation that could not practically be performed live. However if it were, the benefit of an increase in the size of the convergence basin would be lost.

This method yields good performance for visual servoing, but requires the ability to compute the true Hessian at convergence, which is not possible for our problem formulation, as it would require knowing a priori either the true robot pose, which is what we are trying to estimate, or the true range of every point in the live image which is effectively equivalent. This approach has been used in recent work for pose estimation [9], where it is assumed that the consecutive poses are close and thus the initial pose error is small, thus the range values estimated using the initial pose are assumed to be those at the true pose. This does however necessitate computing the Hessian live once per image as it cannot be precomputed which is expensive, with their current implementation reportedly taking approximately 4 seconds per image.

5.4.3 Quasi-Newton

Quasi-Newton methods [79] have been widely used in the literature [60, 41]. The most prominent is BFGS, which is widely regarded as the best general Quasi-Newton method [5] and is reviewed in Section 2.3. They are attractive as they require only the gradient at each iteration to be computed, whilst their iteratively refined estimate of the Hessian allows correlations between the parameters to be captured resulting in improved convergence speeds. The BFGS update is given in Equation (5.34),

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \cdot B_k^{-1} \nabla f_k \quad (5.34)$$

where B_k is the iteratively refined approximation of the Hessian. At each iteration a rank-2 update is applied to B_k derived from the change in $\mathbf{x}$ and ∇f between the current and previous iterations. A robust line search satisfying the Wolfe conditions is performed at each iteration to find the value of α_k which decreases the objective sufficiently, as discussed in Section 2.3.1.

As our parameter space is small: $|\mathbf{se}_3| = 6$, B_k would converge quickly to a good estimate of the true Hessian if the objective were a quadratic [5]. In practice, fast convergence will still be achieved when the problem can be reasonably modelled as a quadratic, which occurs around the minima of $f_{I_{X;Y}}$, and the common shape of the NID and the line search provides robustness outside of this region. Whilst the dimensionality of the parameters (6) being close to that of the iterative rank update to B_k (2), means that the optimiser can quickly incorporate information about local correlations across the parameter space.

5.4.4 Sampling Methods

In one of the earliest works on mutual information based model alignment [115], a stochastic gradient descent algorithm was used, in which randomly drawn samples from the full set of observations of X & Y are used to estimate the mutual information and its derivative at each iteration. A step is then taken in this direction, with the step size, referred to as the learning rate, controlled heuristically. More recent work [41] has also experimented with alternative stochastic sampling methods, which for some cases can result in faster convergence compared to Quasi-Newton methods.

Other early work in this area [58] used Powell’s method, which does not require the gradient to be computed as it works solely on the value of the function, but is typically not as performant as methods which make use of the gradient. The same authors also experimented with Conjugate Gradient (CG) methods [60]. In

general these methods will not be as performant as Quasi-Newton methods such as BFGS, although for large scale problems they can be more attractive as CG methods require less memory than BFGS, but the difference is less pronounced when using the limited memory variant of BFGS: L-BFGS [79].

Some very recent work [122] uses normalised mutual information for $\mathbb{SE}(2)$ robot pose estimation against a prior lidar intensity road surface map, a more constrained version of our problem formulation using normalised mutual information in place of the NID. This work uses a novel sampling strategy using purely the function value, the number of samples taken across each of the 3 dimensions of $\mathbb{SE}(2)$ is constant, but the scaling of the sampling in each axis is proportional to the covariance of the current pose estimate in that axis obtained from an EKF, updated with the output from the image alignment and an IMU. The attraction of a sampling strategy is that it permits convergence even when starting outside the convergence basin where gradient based methods would typically fail. The principal downside is the cost of sampling grows rapidly with the dimensionality of the parameter space. However, this should be weighed against the additional cost of computing the function gradient, if the value of the function is much faster to evaluate than the gradient then sampling may still result in comparable run time performance to gradient methods.

5.4.5 Choice of Optimisation Method

In this work we elected to use the Quasi-Newton BFGS method, as its model of the Hessian allows the efficient capture of the known coupling present in the $\mathbb{SE}(3)$ parameter space, whilst the line search component of the method aids robustness. Furthermore, the BFGS estimate of the Hessian is, by construction, necessarily positive definite whereas as shown previously, the approximate analytical Hessian used in the modified LM approach, obtained using only the first order terms is not (Section 5.1.2 and Figure 5.10). Finally, a sampling strategy in $\mathbb{SE}(3)$ would

necessitate a prohibitive number of samples to achieve similar performance to the Quasi-Newton approach and even accounting for the increased cost of computing the derivatives, this number of evaluations would be significantly slower to compute than a single Quasi-Newton solve.

5.5 Point Subset Selection

In our initial presentation of the minimisation in Equation (4.27), the subset of all points in the scene which are used to compute the cost at each iteration for a given camera is specified as: $\mathbf{q} \in \overline{\mathcal{S}}$. Clearly this subset can contain only those points which reproject into the camera, but this alone does not take into account whether those points should be visible. For example, whether they are from the far side of a wall on a building rather than the near side. As our representation of the scene prior is a sampled pointcloud, in practice such cases can occur.

However the visibility problem particularly for outdoor scenes swept with a 3D lidar(s) mounted on a vehicle can often be mitigated by considering only the subset of the scene captured from the surrounding area defined along the survey route, as shown later in Chapter 6. This typically excludes for even moderate distances instances such as driving on parallel roads, and thus seeing the opposite faces of buildings. Additionally, as our priors are all maintained in separate relative reference frames, this approach also reflects how they are stored and used, which will be discussed in Chapter 6.

Even for scenes where the visibility problem is not directly an issue, it is still not desirable to use all points that reproject into the camera when computing the NID, as this can lead to uneven sampling in the image due to supersampling at vanishing points. This is particularly prevalent when the camera is facing forwards or backwards looking along a straight road an example of which is shown in Figure 5.11.

Although this supersampling faithfully encodes the geometry of the scene, it can reduce the entropy of the joint distribution as the supersampled points will typically all contribute to the same bin(s) essentially irrespective of $G_{R,W}$, provided the vanishing point remains visible. As the number of points at the vanishing point increases as a percentage of N , the total number of points used to build the histogram, the contribution of the rest of the scene is downweighted and the NID becomes approximately invariant to $G_{R,W}$, and thus not suited to localisation. One additional implementation consideration for GPUs, is that for large N , the accuracy of less populated bins can be reduced due to the limitations of single precision floating point arithmetic.

In light of this, we choose $\overline{\mathcal{S}}$ and thus the contribution of the information from the image, to be approximately uniform in the image plane by subsampling to obtain (approximately) one point per pixel in the live image. This is analogous to Z-Buffering for meshes of triangles in OpenGL, a standard method for solving the visibility problem. The approximation in this context comes from when we choose to evaluate the subset of points to process: at each iteration, i.e. for every candidate $G_{R,W}$; or once per optimisation (equivalently image) using the initial estimate of G_{R,W_0} . There are two considerations affecting this choice, firstly if $\overline{\mathcal{S}}$ is changed at every iteration then the resulting NID may not be smooth; and secondly we wish to minimise any unnecessary computational load.

Although in practice for any change in $G_{R,W}$ points around the perimeter of the image will enter / leave the image, as a percentage of the total number of points they are not typically significant and so the resulting NID remains smooth. But if $\overline{\mathcal{S}}$ is recalculated at every iteration to achieve one point per pixel then a large percentage of the points across the image can change at each iteration and the resulting NID is not typically smooth.

Our approach is to subsample once per optimisation such that $\overline{\mathcal{S}}$ contains one

point per pixel using the initial seed pose G_{R,W_0} and then at each iteration use only the subset of $\bar{\mathcal{S}}$ which reproject into the camera using the current candidate pose $G_{R,W}$. Thus for any iteration but the first there is approximately one point per pixel and thus no supersampling of areas within the image, and the resulting NID is smooth whilst also reducing the computational cost.

Figure 5.11 shows a typical straight road scene, with a single camera looking down the road. Approximately 200k points reproject into the image in total, and their density is shown on a log scale in Figure 5.11c, from which it is clear that the density of points at the vanishing point vastly exceeds the rest of the image. In the bottom row, the NID and its gradient for offsets in pitch around the ground truth camera pose are shown for three different subsampling methods, respectively: all points (no subsampling); one point per pixel recalculated at every iteration and one point per pixel calculated only once using the initial seed pose. Note that the NID is obviously not smooth for the one point per pixel recalculated at each iteration case, but is smooth for the case when the subset is only calculated once at the start. However, both look significantly better than the case in which no subsampling is performed.

5.6 Point Importance Weighting

In Section 5.5 we demonstrated the problems caused by non-uniform sampling of the image by the reprojected points, how this undesirably biases the resulting PDF and how it can be solved by using a one point per pixel subset selection algorithm to ensure uniform sampling. However, this motivates the question: are all areas of the live image equally useful for localisation? Intuitively the answer should be no, and we would like to be able to incorporate semantic information extracted from the live images online to reflect the relative importance of different areas of the image

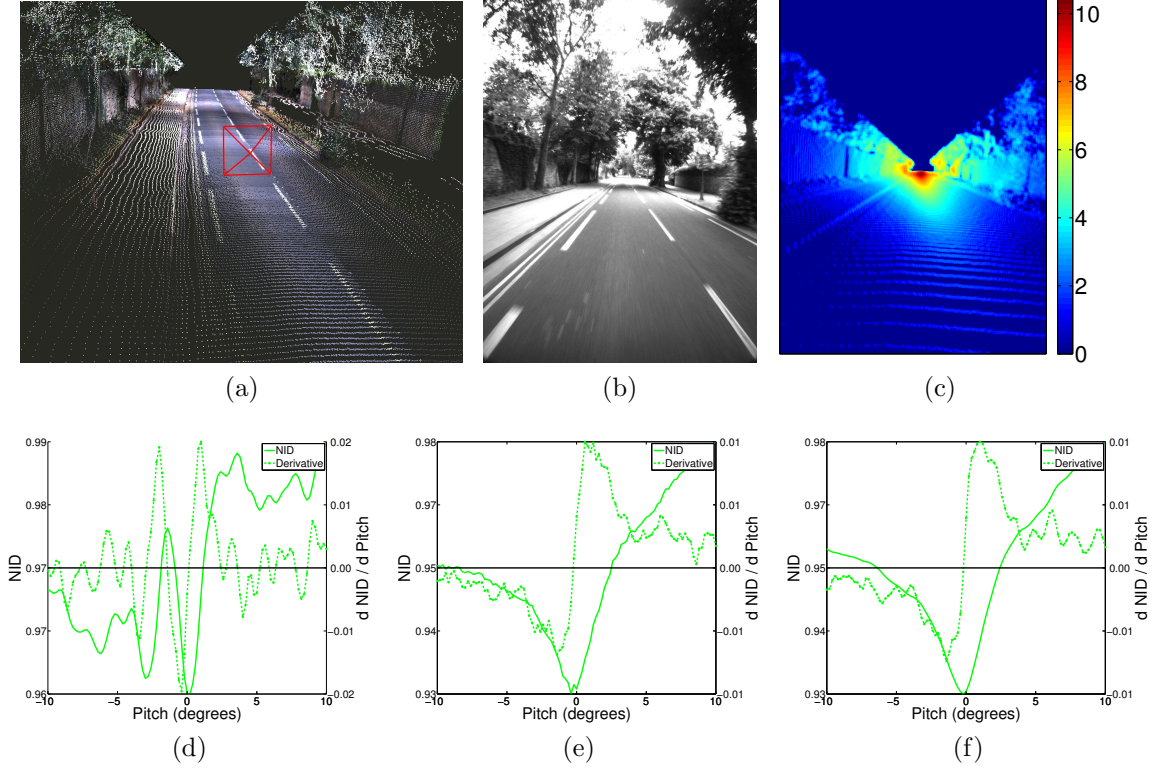


Figure 5.11: Importance of point subset selection. The top row shows left to right: the 3D coloured pointcloud prior $\mathcal{S}$ for a straight section of road with the camera frustum shown in red; the captured camera image and the reprojected point density image, coloured on a log scale. The bottom row shows the NID and its analytically computed gradient versus a pitch offset applied to the ground-truth camera pose for three different point subset selection strategies to build $\bar{\mathcal{S}}$. From left to right: Figure 5.11d, all points that reproject into the camera; Figure 5.11e, a one-point-per-pixel point subset, recomputed for each pitch angle and finally Figure 5.11f, a one-point-per-pixel point subset computed only once using the initial (ground truth) pose and then subsequently reused for all pitch angles.



Figure 5.12: Example distraction suppression mask generated online using stereo and a 3D pointcloud Scene Prior. Taken from [64], courtesy of Colin McManus. On the left, an image from dense traffic in central London, on the right a distraction suppression mask generated online using dense stereo compared against a 3D pointcloud Scene Prior captured earlier. The image is coloured according to the confidence of the prior being visible, with white denoting high confidence and black denoting low confidence. Thus areas coloured black are believed to contain distractions, rather than the prior and should not be used for localisation, as they contain dynamic objects.

for localisation. For example, to downweight the importance of areas of the image containing dynamic objects that are not present in the static scene prior. Although the NID provides good robustness to such occlusions, they are outliers that do not aid localisation against the prior. This is particularly evident in busy traffic, where dynamic outliers may take up the majority of the image, but the prior may still be partly visible and in this case we would like to use only those areas for localisation. This technique was used to great effect in [64] through the use of a distraction suppression mask, an example of which is shown in Figure 5.12. Alternatively, in a manner inspired by [80, 100], we could assign different weights to areas of the image based on the quality of the exposure, to downweight regions where important appearance cues may have been lost due to overexposure.

Formally, we wish to heterogeneously weight the contribution of the visible points in the prior to the overall PDF based on their reprojected position, with the weight determined from semantic information extracted online from the live image. We denote the weight image obtained online from $\mathcal{I}$ by $\mathcal{W}$, assumed to be the same size

as $\mathcal{I}$. The value of each pixel in $\mathcal{W}$ denotes its importance for localisation, with a higher value denoting greater importance.

Starting from Equation (5.22), in the interests of clarity we define $\bar{\beta}_{uv}(x, y)$ given in Equation (5.35),

$$\bar{\beta}_{uv}(\mathbf{x}, z) = \beta_u(u - c) \cdot \beta_v(v - r) \quad (5.35)$$

which simply combines the product of two univariate B-Splines, one in each axis in the image, into a single term for brevity. Where we revert to using the vector: $\mathbf{x} = [u; v]$ to denote the reprojected position and have switched to using pixel linear indexing, denoted by z , in place of separately indexing over the rows and columns.

We also correspondingly replace the double summation over the rows and columns in Equation (5.22),

$$p_{\mathcal{IS}}(i, s) = \frac{1}{N} \sum_{j=0}^N \sum_{r=0}^{\text{Rows} \in \mathcal{I}} \sum_{c=0}^{\text{Cols} \in \mathcal{I}} \left(\beta_u(u_j - c) \cdot \beta_v(v_j - r) \cdot \delta_i(\lfloor \mathcal{I}(r, c) \cdot \eta \rfloor - i) \right) \cdot \delta_s(\lfloor \mathcal{IS}(\mathbf{q}_j) \cdot \eta \rfloor - s) \quad (5.22)$$

with a single summation over the linear pixel indices and also abbreviate the Kronecker delta functions, resulting in Equation (5.36).

$$p_{\mathcal{IS}}(i, s) = \frac{1}{N} \sum_{j=0}^N \sum_{z=0}^{\text{Rows} \times \text{Cols}} \bar{\beta}_{uv}(\mathbf{x}_j, z) \cdot \delta_i \cdot \delta_s \quad (5.36)$$

We stress that these changes are purely cosmetic for brevity before we introduce the modifications for importance weighting: they do not change the result.

Now, we update Equation (5.36) to include the importance weight for the con-

tributions of the j -th point: $\mathcal{W}(\mathbf{x}_j)$, which is exactly the interpolated value of the weight image $\mathcal{W}$ evaluated at $\mathbf{x}_j$, to obtain Equation (5.37).

$$\begin{aligned}
 p_{IS}(i, s) &= \overbrace{\frac{1}{\sum_{j=0}^N \mathcal{W}(\mathbf{x}_j)}}^{\Psi} \overbrace{\sum_{j=0}^N \sum_{z=0}^{\text{Rows} \times \text{Cols}} \mathcal{W}(\mathbf{x}_j) \cdot \bar{\beta}_{uv}(\mathbf{x}_j, z) \cdot \delta_i \cdot \delta_s}^{\Omega} \\
 &= \Psi \cdot \Omega
 \end{aligned} \tag{5.37}$$

For the interpolation, we use cubic B-Spline image interpolation, via an implementation of the GPU-based prefilter from [92], to allow the fast and accurate computation of both the value and the gradient of $\mathcal{W}(\mathbf{x})$.

The derivative of Equation (5.37) can be computed in a similar manner to that of Equation (5.22), using the product rule, resulting in Equation (5.38),

$$\frac{\partial p_{IS}(i, s)}{\partial G_{R,W}} = \frac{\partial \Psi}{\partial G_{R,W}} \Omega + \Psi \frac{\partial \Omega}{\partial G_{R,W}} \tag{5.38}$$

where the partial derivatives terms are given in Equations (5.39) and (5.40).

$$\begin{aligned}
 \frac{\partial \Psi}{\partial G_{R,W}} &= \frac{\partial}{\partial G_{R,W}} \left(\frac{1}{\sum_{j=0}^N \mathcal{W}(\mathbf{x}_j)} \right) \\
 &= \frac{-1}{\left(\sum_{j=0}^N \mathcal{W}(\mathbf{x}_j) \right)^2} \cdot \sum_{j=0}^N \frac{\partial \mathcal{W}(\mathbf{x}_j)}{\partial G_{R,W}}
 \end{aligned} \tag{5.39}$$

$$\frac{\partial \Omega}{\partial G_{R,W}} = \sum_{j=0}^N \sum_{z=0}^{\text{Rows} \times \text{Cols}} \left(\frac{\partial \mathcal{W}(\mathbf{x}_j)}{\partial G_{R,W}} \bar{\beta}_{uv}(\mathbf{x}_j, z) + \mathcal{W}(\mathbf{x}_j) \frac{\partial \bar{\beta}_{uv}(\mathbf{x}_j, z)}{\partial G_{R,W}} \right) \cdot \delta_i \cdot \delta_s \tag{5.40}$$

The partial derivative of $\mathcal{W}(\mathbf{x})$ is formed as described in Section 5.3.3 as: $\frac{\partial \mathcal{W}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial G}$, with $\frac{\partial \mathcal{W}}{\partial \mathbf{x}}$ obtained from cubic B-Spline image interpolation Section 2.2.3.

Finally, we note that the first term from Equation (5.38) simplifies to a scalar weighting of p_{IS} as shown in Equation (5.41).

$$\begin{aligned} \frac{\partial \Psi}{\partial G_{R,W}} \Omega &= \frac{-1}{\left(\sum_j \mathcal{W}(\mathbf{x}_j)\right)^2} \cdot \sum_j \frac{\partial \mathcal{W}(\mathbf{x}_j)}{\partial G_{R,W}} \cdot \sum_j \sum_z \mathcal{W}(\mathbf{x}_j) \cdot \bar{\beta}_{uv}(\mathbf{x}_j, z) \cdot \delta_i \cdot \delta_s \\ &= \frac{-1}{\sum_j \mathcal{W}(\mathbf{x}_j)} \cdot \sum_j \frac{\partial \mathcal{W}(\mathbf{x}_j)}{\partial G_{R,W}} \cdot \left(\frac{1}{\sum_j \mathcal{W}(\mathbf{x}_j)} \sum_j \sum_z \mathcal{W}(\mathbf{x}_j) \cdot \bar{\beta}_{uv}(\mathbf{x}_j, z) \cdot \delta_i \cdot \delta_s \right) \\ &= \frac{-1}{\left(\sum_j \mathcal{W}(\mathbf{x}_j)\right)} \cdot \sum_j \frac{\partial \mathcal{W}(\mathbf{x}_j)}{\partial G_{R,W}} \cdot p_{IS} \end{aligned} \quad (5.41)$$

It is also possible to weight the contribution of the points based on a fixed weight, assigned a priori, to each point in the prior. If the prior was semantically labelled offline, then this would allow the weight of each point to be based on its label and / or its location within the scene prior. This has the attraction that the processing of the pointcloud can happen offline and so does not have to be real-time. As the weight of each point in this case: $\mathcal{W}_S(\mathbf{q})$ is constant, in that it is not a function of $G_{R,W}$, the form is the same as that of Equation (5.37). However, as the per-point weight is constant, the derivatives are unchanged except for the inclusion of $\mathcal{W}_S(\mathbf{q})$.

Although our real-time implementation detailed in Chapter 7 contains the weighting methods described in this section, they are not used in the results presented in this thesis. However, this importance weighting strategy is included here for completeness, as we consider the application of semantic information, perhaps in the form of distraction masks, to be a clear and imminent next step, a matter we return to later in the discussion of future work in Chapter 9.

5.7 Conclusions

In this chapter we first derived the form of the gradient and Hessian of the NID in terms of the derivatives of the joint and marginal PDFs, and discussed the problems associated with approximating the Hessian. We next reviewed the two methods by which the joint appearance distribution required for our formulation can be constructed in order that it be differentiable. The key distinction between them is the space in which the weighted updates for each point are constructed. We presented results for both approaches for a sample problem for comparison and explained our choice of spatial weighting for this work. Next, we discussed the various optimisation methods which have been used for related optimisation problems using entropy measures, and detailed our choice of the BFGS optimiser for this work. In the penultimate section, we demonstrated the importance of selecting a subset of the points from the full scene prior that approximately uniformly samples the image to avoid oversampling at the vanishing points. Finally, we presented a new modification that allows for the points in the prior to be weighted according to their importance using both offline and online semantic information.

Chapter 6

Building the Scene Prior

A core prerequisite of this work is that the environment in which the robot is to be localised has been previously surveyed and that a 3D coloured pointcloud scene prior has been generated. In this chapter we describe the hardware and software used to generate the scene priors used throughout this work.

6.1 Scene Prior Pipeline

The structure of the pipeline used to build the scene priors is shown in Figure 6.1. In the following sections we will discuss each of the components in detail, including the experimental platform used to capture the raw input data and the sensors used and how they were calibrated. Finally, we shall discuss the mechanics by which the scene priors are generated, post-processed to remove dynamic obstacles and then stored to allow fast subsequent access and analysis.

6.2 Experimental Platform

The Mobile Robotics Group’s Wildcat autonomous vehicle shown in Figure 6.2 was used as the survey vehicle in this work, for which it was equipped with the following

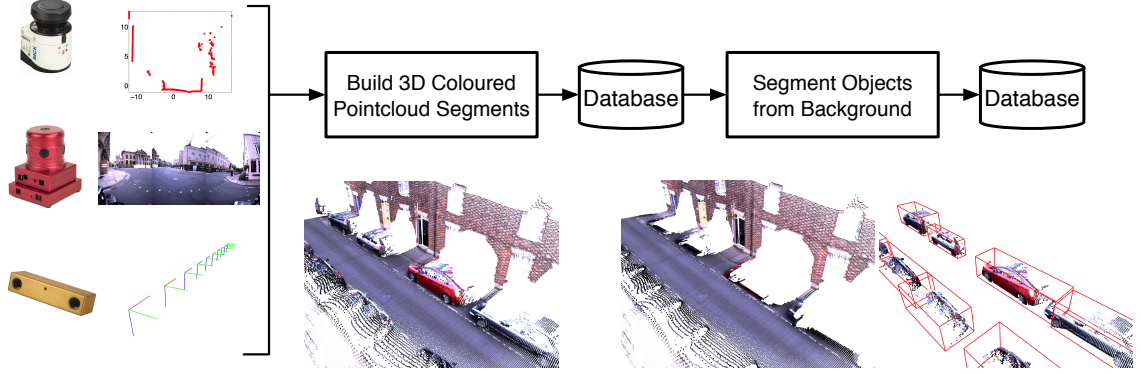


Figure 6.1: The pipeline used to build the Scene Priors used in this work. The inputs are, in order from top to bottom on the left: 2D laser scans from a SICK LMS-151 lidar, camera images from a PGR Ladybug2 omnidirectional camera and a purely relative pose chain obtained using MRG’s Visual Odometry (VO) system [12] processing data from a PGR Bumblebee2 stereo camera. All data was collected using MRG’s Wildcat platform described in Section 6.2. In the first stage, the VO pose chain is used to decompose the input data into segments of approximately equal length based on the traversed distance (typically 30m). For each of these segments, the VO poses are used to construct a locally metric 3D pointcloud from the lidar scans, in which the points are coloured using the closest in time Ladybug2 image in which they were visible. The pointclouds are then saved to disk, references to which, are written to a graph structure maintained in a database for fast subsequent access / analysis. Finally, each pointcloud segment is post-processed using [116] to segment dynamic objects (cars, pedestrians and cyclists) from the static background, the results of which are written back to the database.

sensor payload:



SICK LMS-151 2D Lidar

Captures range and intensity of 541 points in a planar scan evenly spaced in azimuth over 270° field of view at 50 Hz, used to generate 3D pointcloud.



Point Grey Research (PGR) Ladybug2 Omnidirectional Camera

Captures 10 Hz 360° imagery (6 cameras at 768×1024), used to colour 3D points captured by the lidar.



Point Grey Research (PGR) Bumblebee2 Stereo Camera

Captures 20 Hz synchronised stereo images at 1024×768 , used to generate relative pose chain using Visual Odometry (processed at 512×384).



Oxford Technical Solutions (OxTS) RT 3042 Dual-Antenna INS

Provides 100 Hz UTM referenced output, used purely for subsequent geo-referencing of data, not directly used to construct scene prior.

All sensors except for the INS are mounted on the front bumper of the Wildcat as shown in Figure 6.2a. The Bumblebee2 is angled down approximately 12.5° relative to the horizontal ground plane to minimise the percentage of the sky in the image and thus maximise the area of the image in which useful features for VO can be found. The LMS-151 lidar is mounted overhanging the front of the vehicle such that its scanning plane is vertical, perpendicular to the direction of travel and does not intersect the vehicle. The 90° in the scanning plane in which the lidar does not sense as it has a 270° field of view, faces upwards towards the sky ($\pm 45^\circ$ about vertical). The Ladybug2 is mounted above the Bumblebee2, with one camera looking directly in front of the vehicle and the other two sideways facing cameras

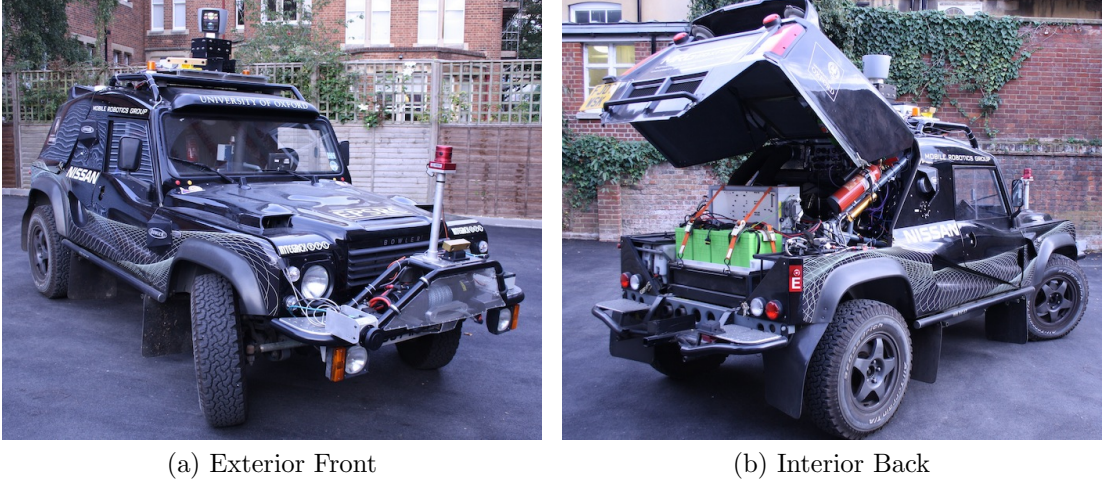


Figure 6.2: The Mobile Robotics Group’s Wildcat autonomous vehicle platform used for collecting scene priors. The base vehicle is a Bowler Wildcat 200 off-road rally car, which has been upgraded into a fully autonomous vehicle by BAE SYSTEMS. The vehicle is equipped with an OxTS RT 3042 dual-antenna GPS inertial navigation system (INS) for ground-truth and has a Dell R710 server for compute, both mounted in the rear. The sensors used for survey collection are mounted on the front bumper, they include: a Point Grey Research Bumblebee2 stereo camera, a SICK LMS-151 lidar with a 270-degree field-of-view mounted such that the scanning plane is vertical and intersects the road but not the vehicle and a Point Grey Research Ladybug2 omnidirectional camera. The Bumblebee2 stereo camera is used for visual odometry, the output of which is the purely relative pose source used to build the scene prior in a relative frame. The scene prior pointclouds are captured by the LMS-151 lidar and coloured by their appearance in the Ladybug2 camera. Note that although the Wildcat is equipped with a Velodyne HDL-64E lidar, it is not used in this work.

yawed $\pm 72^\circ$ relative to the vehicle centerline, such that they are very approximately perpendicular to the lidar’s scanning plane. This results in the left and right (relative to the vehicle) portions of the lidar scan reprojecting into approximately vertical lines down the center of the sideways facing camera sensors. The downward facing section of the lidar which intersects the road cannot be seen in the forward facing Ladybug2 camera when captured and must be coloured with an image taken 2 meters before the capture point.

The vertical SICK LMS-151 lidar configuration is attractive for collecting accurate scene priors relative to a Velodyne HDL-64E for a variety of reasons. Firstly, mounted vertically the LMS-151 provides a large vertical field of view and resolution which allows the facades of buildings to be well captured even when driving in the adjacent lane, as we are primarily interested in capturing a good model of the static background environment this is particularly desirable. The Velodyne HDL-64E by contrast provides only a $27^\circ(-25^\circ, +2^\circ)$ field of view and so does not capture well structure above the height of the sensor and so would not provide good coverage of building facades.

Although the number of points per second produced by the LMS-151: $\approx 2.7 \cdot 10^4$ is substantially less than the HDL-64E: $1.3 \cdot 10^6$, the dense sampling within its field of view provided by the HDL-64E will typically contain a lot of redundant information about the scene structure, which would increase the storage required for the map and computational cost of the subsequent localisation without significantly improving performance. Indeed, the reduced vertical coverage of the scene would reduce the information available for localisation and thus performance.

As noted in [15], the hardware auto-exposure controller in the PGR Ladybug2 does not typically give good performance outdoors, often resulting in well textured clouds and under-exposed buildings or roads, which are the very areas of primary interest to us. As such, a custom auto-exposure controller was implemented in

software for the Ladybug2, heavily inspired by the approach used in [15]. A drawback to this approach is that there is an inherently delay which damps the ability to respond to sudden changes in illumination and this can result in a striping pattern in the pointcloud appearance data when this occurs.

6.2.1 Sensor Calibration

6.2.1.1 Camera Intrinsic Calibration

For the intrinsic calibration of the Ladybug2 omnidirectional and Bumblebee2 stereo cameras, we use the calibration files provided by Point Grey Research. These take the form of a lookup table which maps every integer pixel coordinate in an undistorted output image as would be captured by a idealised pinhole camera, to a sub-pixel coordinate in the distorted input image as captured from the camera. These lookup tables are used to generate undistorted images via bilinear interpolation which are then used for all subsequent processing.

We have also successfully used the OCamCalib toolbox [93]¹ to calibrate both the Ladybug2 and other fisheye cameras, however we found that the resulting coloured pointclouds were qualitatively better with the PGR factory calibration, albeit the difference was not large.

6.2.1.2 Lidar - Camera Extrinsic Calibration

To calibrate the LMS-151 lidar to the Ladybug2 camera, we used an implementation of the method from [83], using components from the AMCC toolbox described in [117]². The method requires that a calibration checkerboard is covisible in both the lidar and the camera(s) to be calibrated. As the dimensions of the checkerboard and camera intrinsics are assumed to be known, the 3D position of the extracted

¹<https://sites.google.com/site/scarabotix/ocamcalib-toolbox>

²<https://code.google.com/p/amcctoolbox/>

checkerboard corners in the camera frame for each image can be estimated, and then used to estimate the equation of the plane in the camera frame on which the checkerboard lies. The lidar points in the corresponding scans that lie on the checkerboard are obtained via a human guided procedure (one click per scan). The resulting cost function is then the L_2 norm of the distances of the lidar points from the checkerboard plane, this method converges after approximately 45 iterations when initialised with the lidar and camera being coincident using 82 image-scan pairs. We have also used an implementation of the method from [72] using datasets from driving around Oxford and achieved similar results.

6.2.1.3 Camera - Camera Extrinsic Calibration

The Ladybug2 camera to Bumblebee2 camera calibration was also performed using the method of [83] described in Section 6.2.1.2, by treating the checkerboard corners extracted from one camera as 3D points sampled from a lidar. This method has the benefit that not all checkerboard corners have to be covisible in both cameras, allowing a more complete coverage across the field of view to be used.

6.3 Construction of Scene Priors

As noted previously, all of the scene priors are built in purely relative, locally metric reference frames using the Visual Odometry (VO) as the pose source, we do not use any georeferenced (GPS) data during their construction. We represent any given prior dataset as a graph stored in a database, where the nodes define the coordinate frame in which metric 3D pointcloud data stored in the directed edges which originate at the node are defined. The nodes correspond exactly to specific VO frames and are placed approximately equal distances along the traversed route as reported by VO, for the priors in this work, the separation is ≈ 30 metres. This

distance is only practically constrained by the distance over which the assumption that VO is locally metrically accurate is valid. In practice 30 metres is a conservative estimate of this distance, but one which we have found yields good results.

All of the lidar scans captured between the timestamps of the bounding nodes are used to construct the pointcloud for that segment, defined in the reference frame of the origin node (that with the smaller timestamp). The resulting pointcloud is then attached to a directed edge from the origin node to the next node when stored in the database.

The use of the VO to build the scene priors over the INS, and thus their existence in purely relative reference frames is not only desirable, but necessary. This is because our primary concern is that the maps are locally metrically accurate, we never require that the priors be registered in a single monolithic metric reference frame. Instead, it is enough for our purposes to have topological and metric links between disjoint areas in the graph, allowing the construction of a locally metric sub-window. For driving in dense urban environments, even an expensive INS system such as the OxTS in the Wildcat can fail to localise completely, or produce spurious data with invalid covariance estimates. Fundamentally, an INS system which strives to produce a globally accurate estimate of the pose inherently trades local metric accuracy for global consistency, for some applications this is desirable. However, when trying to build locally accurate 3D maps, this results in tears, or smears in the 3D structure where local accuracy was violated in favour of global consistency. Our VO system has also been shown to be very robust to changes in illumination and weather conditions [55, 11] and the resulting pointclouds demonstrate the local accuracy obtained, finally this approach also allows us to operate seamlessly indoors where GPS information is not available.

The coloured pointclouds are constructed by processing each Ladybug2 image captured in the temporal window defining the segment in sequence, the VO is used

to reproject all lidar scans captured between the current and next Ladybug2 image into the Ladybug2 image captured $X(\theta)$ metres back along the VO pose chain, where θ is the angle in the lidar scanning plane at which the point was captured, in which they are coloured using bilinear interpolation. For the points looking to the left and right of the vehicle, $X(\theta) = 0$ (i.e. the current image), as the lidar points are directly visible in the sideways facing cameras of the Ladybug2. However, for the points sampled on the road, the $X(\theta) = -2$ metres, because the vertical field of view of the forward facing camera is not great enough to view the points when they are captured, as such a previously captured image taken from further back along the pose chain must be used. Once the points have been coloured, they are then transformed into the origin node’s frame in which they are saved along the VO pose chain. Examples of typical scene priors from a variety of areas in Oxford are given in Figures 6.3 and 6.4.

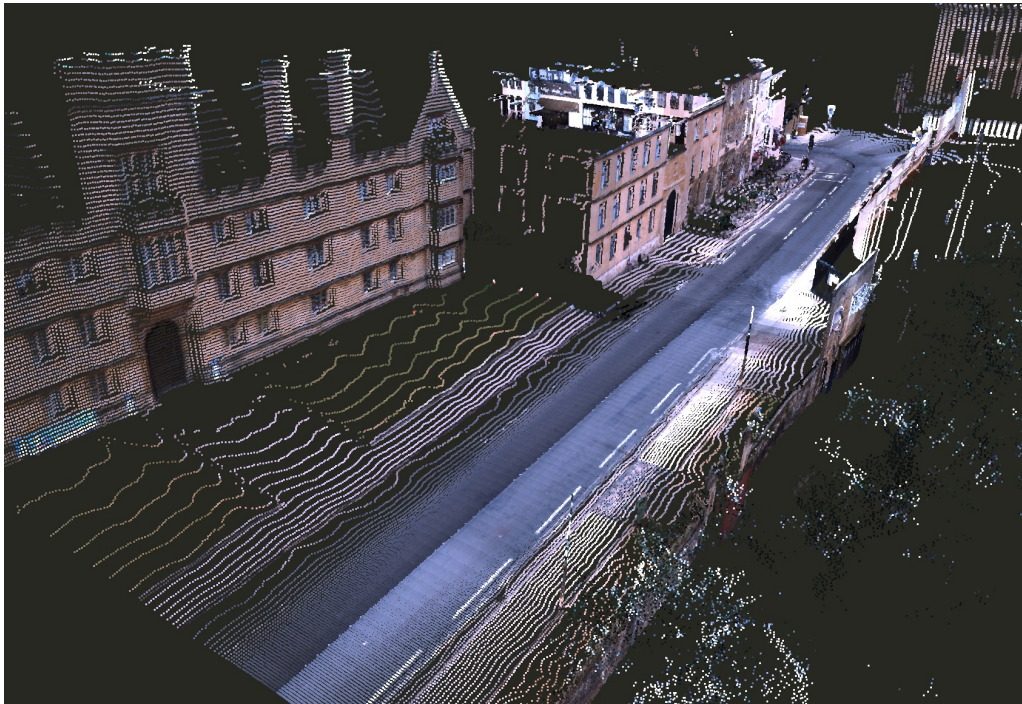
The resulting graph for each dataset starts out as a linear chain with no loop closures. Metric loop closures are then added as additional edges in the graph by performing a VO step between temporally distant images which have been identified by FABMAP [15] as having been taken from the same place. Georeferenced poses from the INS, where available, are also appended as meta data to the edges representing each segment.

All data is stored as Google Protocol Buffers³, which enable subsequent backwards compatible modification of the format, in addition to enabling access across multiple languages including C++, Python and Java, the latter of which enables fast processing within MATLAB. Inclusive of all meta data, a scene prior requires ~ 140 MB / km of storage without additional compression, for comparison the raw data used to build the scene prior requires ~ 9 GB / km. We briefly note that when we refer to the pointcloud data as being stored in a database, we mean that

³<https://code.google.com/p/protobuf/>



(a) Bodleian Library

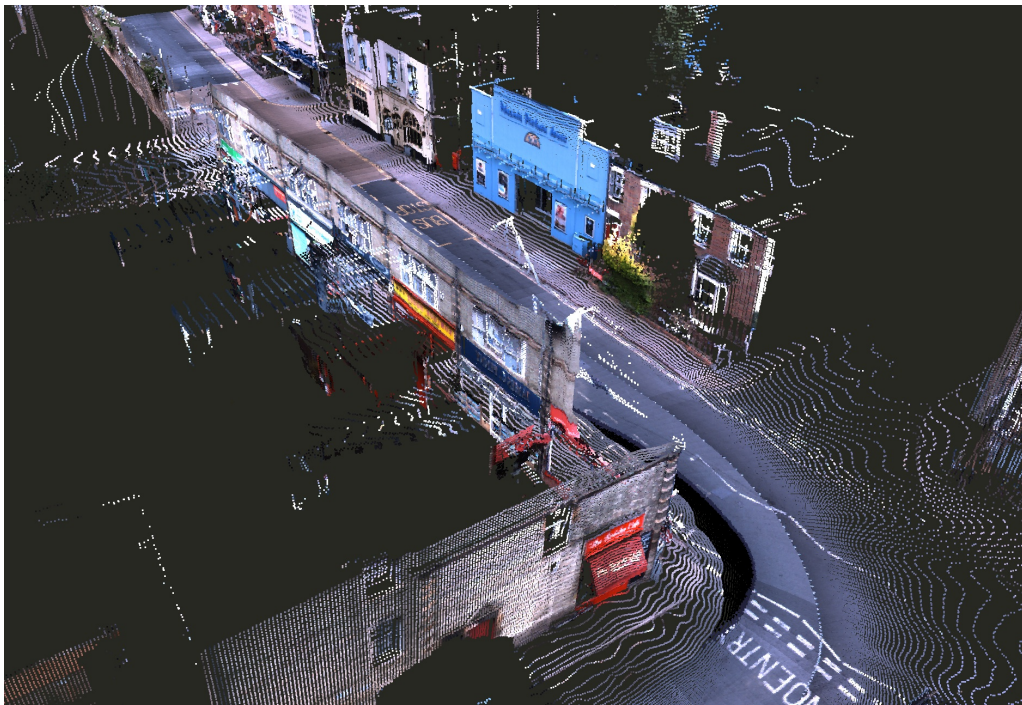


(b) Wadham College

Figure 6.3: Sample Scene Priors: Bodleian Library & Wadham College.



(a) Keble Engineering Triangle Junction



(b) Phoenix Cinema, Walton Street

Figure 6.4: Sample Scene Priors: Keble Engineering Triangle Junction & Phoenix Cinema, Walton Street.

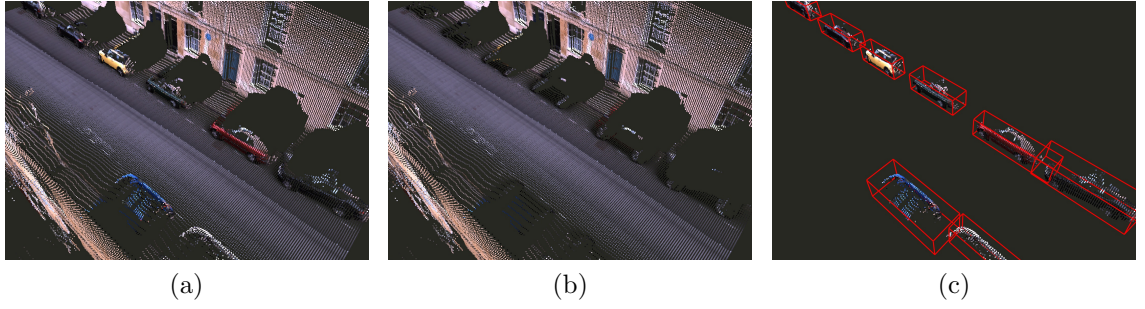


Figure 6.5: Scene Prior Dynamic Object Segmentation. Shown respectively are: all points captured by the Wildcat for a section of St John’s Street in Oxford, the Background class after all dynamic objects (here solely cars) as classified by [116] have been removed and lastly the segmented cars themselves.

references to the indexed binary files containing the actual data are stored in the database, not the data itself. However, the API for subsequently accessing the data hides this distinction, although it is necessary for performance and portability.

6.4 Segmentation of Dynamic Obstacles

Although the scene priors generated using the technique from Section 6.3 generally produce good coverage of the 3D structure and appearance of the static structure in the scene, they can also include potentially dynamic objects such as cars. As we are only interested in the static background component of the scene, we post process the pointclouds using the method of [116] as shown in Figure 6.1 to segment any potentially dynamic objects (cars, pedestrians and cyclists) from the background. Note that as we are using a swept 2D lidar to build a 3D model (not a direct 3D lidar) and as its scanning plane is vertical and perpendicular to the survey vehicle, moving cars are rarely captured in the data and when they are, their effect on the quality of the map is usually minimal. The purpose of the segmentation stage is primarily to remove potentially (but not currently) dynamic objects, in particular, stationary cars.

The results of the segmentation are written back to the database and the segmented background is then used as the scene prior for this work. Although the segmented output is not perfect and can fail to identify abnormally large vans or lorries, or occasionally misclassify a section of hedge as a car, in general we found this method combined with a lightweight sanity check validation correctly captures nearly all vehicles captured in the prior. An example of the typical output is shown in Figure 6.5.

6.5 Conclusions

In this chapter we presented the method by which the coloured 3D pointcloud scene priors used throughout this work are collected, constructed, post-processed and stored. We discussed the approach by which the scene priors are built in a purely relative reference frame, stored in a database for fast indexing and segmented offline to remove dynamic objects, and presented some sample results from central Oxford.

Chapter 7

Implementation

In this chapter, we describe in detail the development of a real-time capable implementation, using commodity GPUs, of the formulation for monocular localisation against a coloured 3D pointcloud prior presented in Chapters 3 and 5. We present results demonstrating the convergence performance of the method using data from central Oxford, which motivates our development of a solution classifier to independently identify failed localisations from the shape of the NID at termination. Finally, we discuss the Spline Fusion algorithm introduced in [53], and explain how it is used to fuse the LAPS localisations with an odometry source.

7.1 System Overview

In order to evaluate the approach proposed in this work at scale, a real-time capable implementation is required. As part of this work, a reference system was constructed leveraging GPUs to accelerate computation of the NID and its derivatives. The components of this system and how they are combined to produce a complete localisation system are shown in Figure 7.1. In this section, we will provide an overview of the components, their purpose and make-up. The remaining

sections in this chapter then describe each component in detail.

The online inputs to the system are: synchronised (in time) images captured from a collection of calibrated monocular cameras with complementary fields of view mounted on the robot to be localised; and an odometry source, which provides purely relative pose updates and which is fused with LAPS localisations to produce the output pose. The synchronisation of the cameras is required so that we can assume the images are all coincident in time and thus perform a single optimisation for the pose of the robot at that time, treating the information from all cameras equally. If this were not true, we would have to separately optimise for the pose of the robot at the times at which the images from each of the cameras were captured. Whilst possible, this is in practice more computationally expensive, but is also less desirable because the combined NID from cameras with complementary fields of view will be more tightly constrained with respect to $G_{R,W}$ and is thus better to optimise. Intuitively, a camera which looks straight down a road may constrain well the pitch of the robot, but not forward motion along the vehicle’s trackline, whilst cameras facing sideways may have the opposite behaviour, thus the combination of the two would well constrain the pose of the robot across all of the components of $G_{R,W}$.

For every set of images, a LAPS observation is built and subsequently processed to produce a candidate LAPS localisation: $\hat{G}_{R,W}$. Each observation consists of the undistorted images, optionally rescaled and converted into normalised single channel images, a seed pose G_{R,W_0} , and the section of the complete scene prior centered at G_{R,W_0} : $\mathcal{S}$.

Any function which maps a multi-channel input image into a normalised single channel image is permissible. In this work, we use the standard RGB to Gray linear colourspace transform [6]. However in other work we have also explored using a nonlinear, illumination invariant colourspace transform [55].

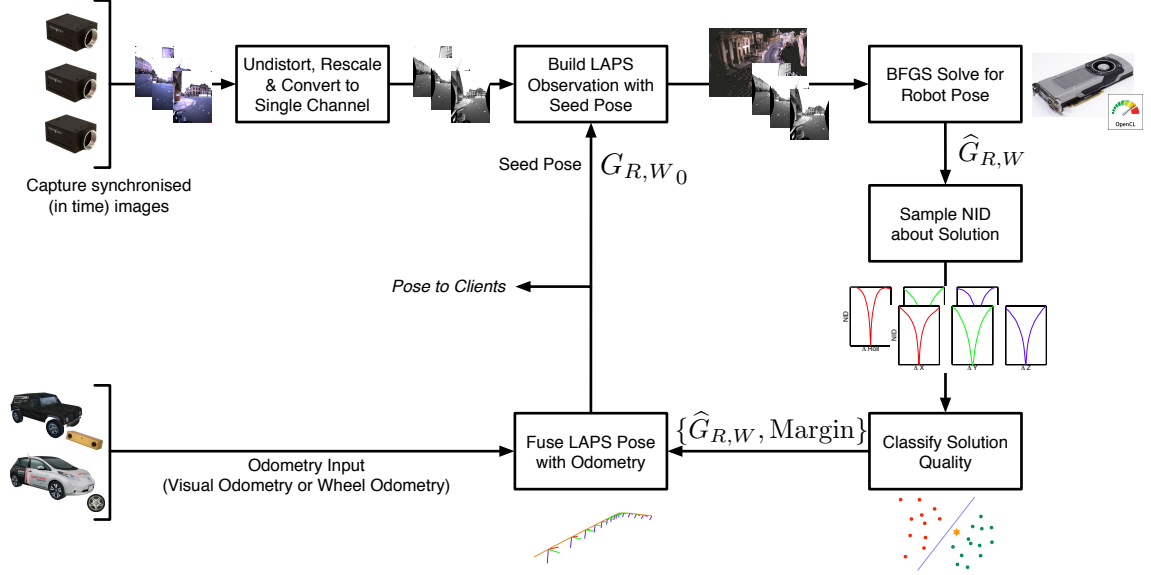


Figure 7.1: LAPS Localisation System. The complete LAPS localisation system pipeline showing the interconnections between the core components. Input images synchronised in time are read from the calibrated cameras, undistorted, rescaled and converted to normalised single channel images, where in this work we use the standard linear RGB to Gray transform. A seed pose, an estimate of the pose of the robot at the time the images were captured: G_{R,W_0} is obtained from fusing the (purely relative) odometry source with the previous LAPS localisations. The seed pose and a local subset of the complete $\mathcal{S}$ (identified using the seed pose) are associated with the images to form a LAPS observation. A BFGS solver optimising the L_2 norm of the vector of the NID for all cameras with respect to the robot pose $G_{R,W}$ is initialised using the data in the LAPS observation and the resulting $G_{R,W}$ is the candidate LAPS localisation: $\hat{G}_{R,W}$. The total NID across all cameras is sampled about $\hat{G}_{R,W}$, normalised and used in addition to the NID at convergence as features for an RBF SVM classifier to classify whether $\hat{G}_{R,W}$ is a valid or spurious solution. Finally, all valid LAPS localisations are fused with an odometry source (either VO or wheel odometry) to produce the pose estimate output to clients, and used as the seed pose for the next set of input images.

When the system is first initialised or after it has become lost, and thus has no idea of its location, the seed pose must come from an external source such as GPS or FABMAP [15]. However, during normal continuous operation, the seed pose is simply the pose output from the system itself, formed from fusing a purely relative odometry source with the previous LAPS localisations without requiring any external source.

For each LAPS observation, a BFGS solver minimising the L_2 norm of the vector of the NID values for all cameras with respect to the estimated robot pose: $\hat{G}_{R,W}$ is initialised at G_{R,W_0} . The use of the L_2 norm is purely a consequence of using the Ceres solver [1], which is at present specific to solving nonlinear least squares problems. However, we have found by comparison with a separate BFGS implementation in MATLAB minimising the L_1 norm of the NID vector that this makes a negligible difference to the overall convergence performance.

The joint appearance distribution and its derivatives from which NID and its Jacobian used in the solver are computed, are constructed in parallel for each camera on commodity GPUs using OpenCL¹, a cross-platform heterogeneous compute language. The resulting pose, returned by the solver, assuming convergence occurs, is then the candidate LAPS localisation.

As we will show in Sections 7.3.1 and 7.4, convergence of the solver and the NID at termination are not on their own necessarily good indicators of a correct localisation. In order to prevent such a failed localisation from propagating through the system and preventing further localisations, we sample the NID about the candidate pose and use a previously trained classifier to estimate the validity of the localisation. This is based on the observation that the NID sampled about the true minima derives from a very characteristic shape in all dimensions and that this signature is not shared by failed localisations and is thus a good feature for classification.

¹<https://www.khronos.org/opencl/>

All LAPS localisations accepted by the classifier as valid are fused with the odometry input using an implementation of the continuous time Spline Fusion algorithm [53] to produce the final output pose. This is the pose which is then fed to other clients, and also used as the seed pose for future LAPS observations. The pose reported is by definition, relative to the scene prior $\mathcal{S}$, which as noted in Chapter 6 is defined in purely relative, locally metric reference frames maintained in a graph. In practice, we output pose relative to the ‘closest’ node in the graph which will typically be the centre of $\mathcal{S}$ for the observation being processed. Using this pose, clients can index into the additional meta data stored in the graph, such as the INS trace collected with the prior (if available), to estimate the current georeferenced pose. Although, in general the systems developed by MRG are designed to work with poses defined in sequences of purely relative reference frames, generally referred to as ‘Experiences’ [11].

7.2 Calculation of NID & Derivatives on GPUs

In Chapter 5, we described in detail the formulation of the optimisation we seek to solve, in this section we consider how we can evaluate the NID and its Jacobian used in this formulation at the speeds required for a real-time capable system.

Evaluating NID and its Jacobian quickly, requires the fast evaluation of the joint appearance distribution: $p_{\mathcal{I}\mathcal{S}}$ and its derivatives, as this is by far the most computationally expensive component of evaluating the NID. As noted in Section 5.3, we use spatial weighting with cubic B-Splines to build $p_{\mathcal{I}\mathcal{S}}$, for which we need to reproject every point in $\mathcal{S}$ into the camera and for all interior points, query a 4×4 patch in the image $\mathcal{I}$ and then update up to 16 bins in a histogram. Including the derivatives of $p_{\mathcal{I}\mathcal{S}}$, there are 7 histograms which need to be constructed, one for $p_{\mathcal{I}\mathcal{S}}$ itself and one for each of the derivatives with respect to each of the 6 dimensions of

the minimal $\mathbf{sc}_3$ representation of $G_{R,W}$: $\frac{\partial p_{IS}}{\partial \mathbf{g}_{R,W}^i}$.

For realistic environments where $|\mathcal{S}| \approx 10^6$, because a significant number of these operations are independent for each point and require fast image queries, this problem is well suited to evaluation on a GPU. The construction of each histogram can also be parallelised to a degree by building separate histograms for consecutive chunks of the data and then reducing them to a single histogram.

There are several languages available for GPU compute, the most established of which are CUDA, arguably the most prevalent, which is a proprietary language for NVIDIA GPUs and OpenCL, which is a cross-platform heterogeneous compute language for GPUs, CPUs and other more exotic accelerators such as Intel’s Xeon Phi and IBM’s Cell processor. The OpenGL Shader Language (GLSL) is another cross platform language that can also be used for compute, but is primarily focussed on graphics. In this work, we selected OpenCL, due to its cross-platform nature and focus on compute, which imposes fewer design constraints than GLSL.

The pipeline used to evaluate the NID and its Jacobian is shown in Figure 7.2. The evaluation of all stages to generate p_{IS} and its derivatives are computed on the GPU, whilst the final stage which evaluates the NID and its Jacobian from p_{IS} and its derivatives is computed on the CPU. This reflects the fact that for typically sized histograms where η is 8 or 16, the final steps of iterating over the PDF to evaluate the entropy terms is significantly faster in series on a CPU.

As the primary client of this pipeline is the BFGS solver, we will explain it in terms of that use case, which is the one for which it was optimised. For every new LAPS observation, the optimiser will evaluate the NID and its Jacobian multiple times with different values of $G_{R,W}$ at each iteration using the same set of images and $\mathcal{S}$. The images from each camera are processed by separate parallel instances of the pipeline shown in Figure 7.2, which can be spread across multiple OpenCL devices. As such in the following text we will describe the pipeline in its true form,

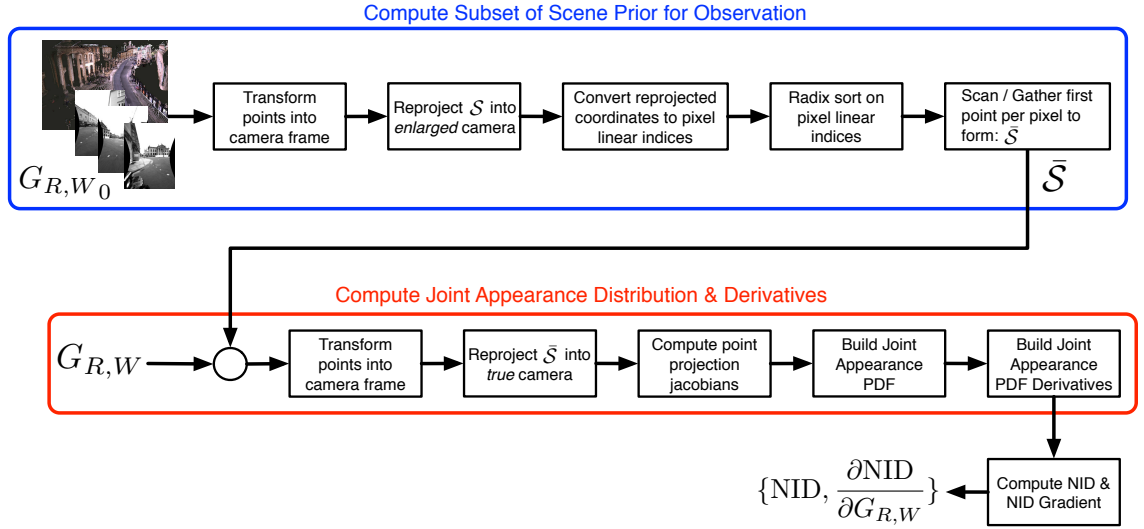


Figure 7.2: The OpenCL pipeline to evaluate NID and its Jacobian. Each camera being processed has its own instantiation of the pipeline executing in parallel. When a new LAPS observation is received, the seed pose: G_{R,W_0} is used to subsample $\mathcal{S}$ to form $\bar{\mathcal{S}}$ using a one point per pixel strategy assuming an enlarged camera model with a wider field of view and greater resolution, highlighted in blue. Subsequent repeated evaluations of NID and its Jacobian with different values of $G_{R,W}$ for the current LAPS observation, highlighted in red, fix $\bar{\mathcal{S}}$. This maps to the use case of the primary client of the pipeline, the BFGS optimiser. All stages except the final evaluation of the NID and its Jacobian from the p_{IS} and its derivatives takes place on a GPU.

as processing a single camera but it should be understood that in reality there is an instance of the pipeline for each camera running in parallel.

7.2.1 One Point per Pixel Subsampling of $\mathcal{S}$

As noted in Section 5.5, it is important to subsample $\mathcal{S}$ to prevent oversampling of vanishing points in the image, for which we use a one point per pixel strategy. To ensure continuity of the cost function, we wish to keep the subset $\bar{\mathcal{S}}$ fixed for all $G_{R,W}$ poses evaluated for a single LAPS observation, which also helps improve performance. Hence the first stage for each new LAPS observation is to evaluate the scene prior subset $\bar{\mathcal{S}}$, using the seed pose G_{R,W_0} , highlighted in blue in Figure 7.2.

We emphasise that $\bar{\mathcal{S}}$ constitutes the candidate points for all subsequent evaluations of $G_{R,W}$ for a given LAPS observation. The actual points used at each iteration are only those from $\bar{\mathcal{S}}$ which reproject into the (true) camera using the specified $G_{R,W}$.

One obvious potential drawback of using G_{R,W_0} to compute $\bar{\mathcal{S}}$ is that if it is not close to the true value of $G_{R,W}$, then there might be large gaps in the distribution of $\bar{\mathcal{S}}$ when reprojected into the image at the true pose. For example as the camera is rotated away from G_{R,W_0} , there will be no points on the side of the image towards which the camera is being turned. To alleviate this issue and thus increase robustness, we use an *enlarged* camera model with a shorter focal length (and thus a wider field of view) and a higher resolution than the true camera when subsampling $\mathcal{S}$. This results in an $\bar{\mathcal{S}}$ which contains points outside of the field of view of the true camera model positioned at G_{R,W_0} and a potentially greater number of points than there are pixels in the true camera. Hence during the optimisation, as $\hat{G}_{R,W}$ moves away from G_{R,W_0} there will still be points in $\bar{\mathcal{S}}$ that cover the image; thus maintaining a more constant image sampling which improves performance. Increasing the resolution of the enlarged camera allows more points to be used whilst still

retaining the benefits of subsampling, or for the number of points processed to be held constant if the input images are downsampled.

Ideally, the (single) point selected for each pixel would be the closest to the camera, this way some occlusion checking is implicitly performed, as the nearest points occlude more distant points. Thus if observing the front of a building with sampled points on both the front and back faces, then for pixels in which points from both the front and back faces reproject, the closest front face points will be selected. This is however imperfect, as points from the back face would still be selected if no points from the front face fell into the same pixel. Correctly handling occlusion would require knowledge of the surfaces from which the points are sampled, so that in the building example, the front surface occludes all of the points sampled from the back. When processing graphics, modern GPUs process triangular surface primitives rather than points for this reason and occlusion checking is performed in hardware via a process known as Z-buffering.

As our scene prior $\mathcal{S}$ consists of points, not a triangular mesh, exploiting these native hardware instructions directly is not possible. As such, we approximate the desired result by relaxing the requirement that the point selected for each pixel be the closest, instead we merely require that a single point be selected per pixel. This allows us to build $\bar{\mathcal{S}}$ by sorting: `sub2ind([x])` where `sub2ind()` converts a (row,column) subscript pixel index into a single linear pixel index assuming row-major ordering; then using Scan - Gather operations ([78]) to select the first point per pixel by selecting those points for which in the sorted indices: `index[k]  $\neq$  index[k-1]`. All of these operations can be performed quickly in parallel on GPU hardware, in this work we have used the optimised OpenCL implementations of Scan and Radix Sort in the open source CLOGS library².

We have found that this approximation works well in practice, because as dis-

²<http://sourceforge.net/p/clogs/wiki/Home>

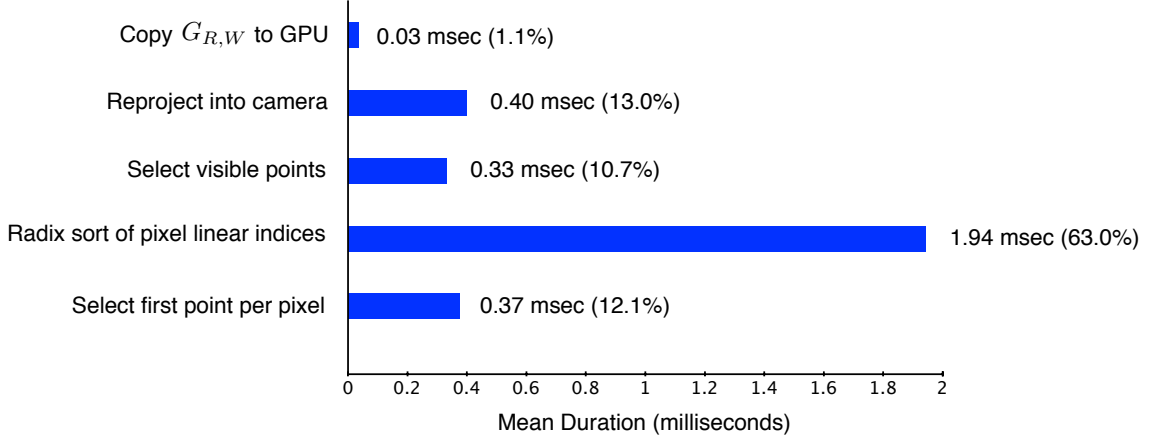


Figure 7.3: Performance of OpenCL one point per pixel subsampling. Mean duration of each stage within the one point per pixel subsampling OpenCL pipeline in milliseconds, the mean total time is 3.08 milliseconds. Evaluated over 600 evaluations with a 768×1024 image and $|\mathcal{S}| = 2^{20} \approx 1.04 \cdot 10^6$ points resulting in $|\bar{\mathcal{S}}| = 2^{16} \approx 65.5 \cdot 10^3$ subsampled points, all evaluations performed on an NVIDIA TITAN GPU.

cussed in Section 5.5, the manner in which $\mathcal{S}$ is stored and local subsets are selected means that it is unusual for multiple faces of a building to be in the same local window of $\mathcal{S}$ defined along the trackline. It is not however impossible and does occur for sharp bends, however we have found that even in these cases this algorithm yields acceptable results in practice for these cases. Thus our most common rationale for subsampling is to handle vanishing points, for which point selected is less critical, as all are far away from the camera and occlusion is not a concern.

Figure 7.3 shows the mean duration of each stage within the one point per pixel subsampling pipeline in milliseconds, evaluated over 600 evaluations with a 768×1024 image and $|\mathcal{S}| = 2^{20} \approx 1.04 \cdot 10^6$ points resulting in $|\bar{\mathcal{S}}| = 2^{16} \approx 65.5 \cdot 10^3$ subsampled points. All timings were generated using an NVIDIA TITAN GPU, note that the number of points is always rounded up to the next power-of-2 for speed of computation.

7.2.2 Building Joint Appearance Distributions on a GPU

Once the scene prior subset $\bar{\mathcal{S}}$ has been computed for the LAPS observation, the p_{IS} and its derivatives can be computed using $\bar{\mathcal{S}}$ for an arbitrary candidate pose $G_{R,W}$. The construction of these normalised histograms is computed on the GPU using the process shown highlighted in red in Figure 7.2.

The process begins by computing the active subset of $\bar{\mathcal{S}}$ which will contribute to p_{IS} and its derivatives, defined as the subset which reproject into the camera under the specified pose $G_{R,W}$. For each of these active points, the point projection Jacobians can be computed separately in parallel, as can the product of the two univariate cubic B-Splines which define the weight to assign to the bins containing the intensities of the pixels in the bounding 4×4 pixel patch in the image.

The main issue that arises is how the seven histograms representing p_{IS} and its six derivatives with respect to the minimal $\mathfrak{se}_3$ parameterisation of $G_{R,W}$ should be constructed to achieve maximum performance on a GPU, as it appears to be an intrinsically serial procedure that necessitates an atomic operation when applying each update to a bin. Intuitively, to compute a histogram in parallel, the input data should be divided into chunks and a histogram constructed separately for each chunk; finally, the histograms for each chunk should be summed together to produce the complete histogram. If the granularity was such that there was a single thread executing each chunk, then no atomic operations are required to update the bins as there is no other thread to contend with. On a CPU with a small number of cores, this approach would generally be optimal.

However, on a modern GPU there can be thousands of cores, which share memory resources in groups called ‘workgroups’ in OpenCL parlance, which are comparatively limited relative to those of a CPU and so it is often impractical / impossible for every thread to have their own copy of the histogram. As such each workgroup of threads, collectively update in parallel using atomic operations multiple copies of

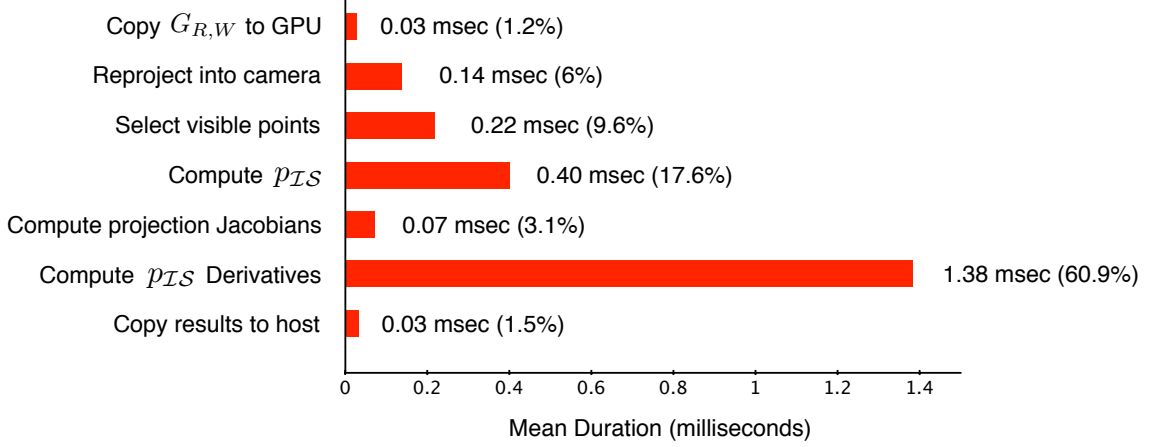


Figure 7.4: Performance of OpenCL evaluation of p_{IS} and derivatives. Mean duration of each stage in the OpenCL pipeline to evaluate of p_{IS} and its 6 derivatives with respect to the minimal parameterisation of $G_{R,W}$ in milliseconds, the mean total time is 2.27 milliseconds. Evaluated over 600 evaluations with a 768×1024 image and $|\bar{S}| = 2^{16} \approx 65.5 \cdot 10^3$ subsampled points, all evaluations performed on an NVIDIA TITAN GPU.

the histogram for different subsets of points stored in fast local memory, which is shared between them and not accessible outside the group. The final stage of each workgroup of threads is to reduce these multiple histograms in shared local memory into a single histogram for the complete workgroup. As there are many workgroups, each of which execute in parallel, this results in one histogram per workgroup which must then be further reduced in parallel to a single histogram containing all of the input data. The normalisation to convert the histogram to a PDF occurs only at the end, after all reductions have taken place to maintain precision. For the same reason Kahan summation [32], a method to significantly reduce the numerical error when summing a sequence of floating point values, should be used when reducing the histograms. This is important because GPUs are optimised for processing 32-bit floating point numbers and are typically much slower if 64-bit floating point values are used instead. Thus numerical error is a much more significant concern than on a CPU, where 64-bit floating point values are the norm.

Figure 7.4 shows the mean duration of each stage in the evaluation of p_{IS} and its

6 derivatives with respect to the minimal parameterisation of $G_{R,W}$ in milliseconds. Evaluated over 600 evaluations with a 768×1024 image and $|\bar{S}| = 2^{16} \approx 65.5 \cdot 10^3$ subsampled points, all evaluations performed on an NVIDIA TITAN GPU.

For a fast implementation, two aspects are particularly crucial, both concern memory accesses. First, the global memory accesses of the information for each point must be coalesced [81]. This means that within a workgroup of threads, where each thread is assigned an index, consecutive threads access consecutive blocks of equal size in global memory. Thus memory is processed in workgroup sized blocks, with each thread skipping: `sizeof(object) × |workgroup| × # workgroups` bytes after reading each object. Second, the copies of the histogram stored in local memory for each workgroup should be arranged such that they are all in separate memory banks [30, 81]. For local memory in modern GPUs (for both AMD & NVIDIA) consecutive 4 bytes blocks are stored in separate memory banks modulo 16, the total number of memory banks. Hence, if the local memory is used to store an array of type `float`, the first 16 values will be in separate memory banks, but the 17th value will be in the same memory bank as the first and so on. This is important for a fast implementation, because accesses to different memory banks happen in parallel, but accesses to the same bank are serialised. Hence, arranging the local histogram copies such that they are not stored in overlapping memory banks means that there is no contention between threads updating different histogram copies and thus improves performance.

In practice, dependent upon the specific GPU hardware used and the size of the histogram, it may be faster to have < 16 local copies of the histogram. In this case, it is still important to ensure that the local histogram copies are stored in non-overlapping memory banks, even if each histogram is itself spread across multiple memory banks.

7.3 Optimising NID

In order to determine the estimate of the true robot pose $\hat{G}_{R,W}$ for each LAPS observation, a BFGS optimiser is initialised at the seed pose G_{R,W_0} , minimising the L_2 norm³ of the vector of NID values for all of the synchronised cameras as shown in Equation (7.1).

$$\hat{G}_{R,W} = \arg \min_{G_{R,W}} \sum_{c=0}^{|\text{Cameras}|} \text{NID}^2(\mathcal{I}_c, \overline{\mathcal{S}}_c) \quad (7.1)$$

We use the implementation of BFGS using a cubic line search satisfying the strong Wolfe conditions (Section 2.3) in Ceres [1], which we developed and contributed as part of this work. The implementation broadly follows the method described in [79], but with modifications to improve robustness to inconsistencies in the residuals.

As the parameter space for the optimiser represents the minimal parameterisation of an $\mathbb{SE}(3)$ pose $G_{R,W}$, there are two important considerations surrounding the update step at each iteration in the optimiser, for which the general case is shown in Equation (7.2).

$$\mathbf{x}_{k+1} = \mathbf{x}_k \boxplus \Delta \mathbf{x}_k \quad (7.2)$$

$$\mathbf{g}_{R,W_{k+1}} = \mathbf{g}_{R,W_k} \boxplus \Delta \mathbf{g}_k \Rightarrow G_{R,W_{k+1}} = \Delta G_k \cdot G_{R,W_k} \quad (7.3)$$

Firstly, the $\boxplus$ update cannot be performed with straight vector addition of the current minimal pose parameterisation and a delta, as this does not reflect their $\mathbb{SE}(3)$ nature and the rules of combining transforms in $\mathbb{SE}(3)$. Instead, the param-

³As noted in Section 7.1, the use of the L_2 norm is a consequence of using the Ceres solver [1], which is at present specific to solving nonlinear least squares problems. However, we have found by comparison with a separate MATLAB implementation minimising the L_1 norm of the NID vector that this makes a negligible difference to the overall convergence performance.

eters must be converted into their $\text{SE}(3)$ form, in which a left-multiply update is performed and then finally the resulting new pose converted back to the minimal pose parameterisation $\mathbf{g}_{R,W}$ as shown in Equation (7.3). A left-multiply update is required here, so that the delta is applied in the robot frame as we estimate the pose as $G_{R,W}$, the transform from the world frame into the robot frame. This representation is natural for us as we are repeatedly transforming $\mathcal{S}$ defined in the world frame into the frame of a camera mounted on the robot.

Secondly, the nature of our problem is such that the magnitude of the orientation (rotational) components of the NID Jacobian with respect to $G_{R,W}$ will be much larger than those of the translation components. In general, such dramatic differences in parameter sensitivity are not desirable in an optimiser and a standard technique to mitigating the resulting loss of performance is to solve instead a preconditioned problem in which the sensitivity of the parameters are more equal [5]. Intuitively, this can be achieved by scaling steps taken in the more sensitive parameters such that in the preconditioned problem solved by the optimiser, a unit step in that parameter equates to a step of $s < 1$ in the original problem.

Formally, the optimiser takes steps in a preconditioned space $\mathbf{h}_{R,W}$ defined in Equation (7.4),

$$\mathbf{g}_{R,W} = S \cdot \mathbf{h}_{R,W} : S \in \mathbb{R}^{6 \times 6} \equiv \text{diag}(\mathbf{s}) \quad (7.4)$$

$$\begin{aligned} \mathbf{g}_{R,W_{k+1}} &= \mathbf{g}_{R,W_k} \boxplus \Delta \mathbf{g}_k \\ &= S \cdot \mathbf{h}_{R,W_k} \boxplus S \cdot \Delta \mathbf{h}_k \end{aligned} \quad (7.5)$$

$$\mathbf{J}_{\mathbf{h}} = \mathbf{J} \cdot S \quad (7.6)$$

where $\text{diag}()$ converts a vector into a diagonal matrix, and in this work we have used: $\mathbf{s} = [1, 1, 1, 0.1, 0.1, 0.1]$. The update at each iteration is performed in the true

space as shown in Equation (7.5) as is by necessity the cost function evaluation, and the resulting Jacobian in the true space: J is transformed into the preconditioned space: J_h using Equation (7.6) before being returned to the optimiser. The use of this preconditioning makes a substantial improvement to the overall performance of the BFGS optimiser.

Finally, we constrain the magnitude of the components of the complete update: $(G_{R,W_k}^{-1} \cdot G_{R,W_0})$ to be within some specified bounds. This is achieved by simply returning a proxy for a cost of ∞ to the optimiser, and thus forcing the line search to backtrack (i.e. reduce the step size). This is effective, in that it prevents the optimiser occasionally taking improbably large steps and moving outside the convergence basin, however a better approach would be to use the L-BFGS-B algorithm [7], an explicit bounds constrained version of BFGS.

Figure 7.5 shows the performance of the optimiser for a typical scene from Holywell Street in Oxford, showing the reprojected point images, that is the images formed by reprojecting $|\bar{\mathcal{S}}|$ into the cameras at a specified pose, at both the initial (in error) seed pose G_{R,W_0} and the converged pose $\hat{G}_{R,W}$. As can be seen from Figures 7.5c and 7.5e, the optimiser has converged to an approximately error free pose. In this case, we let the optimiser run until convergence, defined as: $\frac{|\Delta \text{NID}^2|}{\text{NID}^2} < 10^{-6}$, which took 35 iterations (≈ 250 milliseconds).

In the work presented here, we perform a single optimisation using a fixed image scale, rather than an image pyramid approach involving repeated optimisations at different scales as used in [20, 86]. This is because we have found empirically that any expansion of the convergence basin for greatly reduced scales, if present, is moderate and often moves the minima away from the true pose or otherwise distorts its shape. Furthermore, we have found that the BFGS optimiser at a single scale results in good performance and reduces the complexity of the implementation. Finally, the presence of the solution classifier (Figure 7.1 and Section 7.4) improves

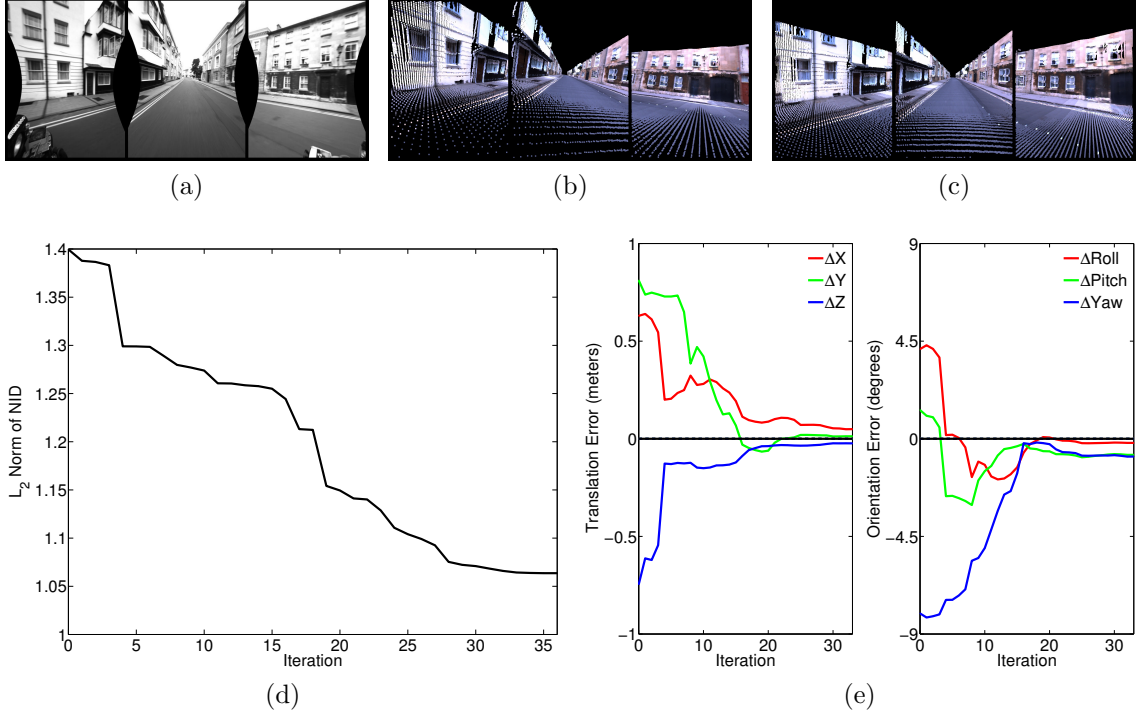


Figure 7.5: Sample BFGS solver optimisation for a typical scene in Holywell Street, Oxford. The solver was initialised with a seed pose G_{R,W_0} composed of random errors applied about ground truth to all six Euler parameters of the pose (XYZ: $[0.63, 0.81, -0.75]$ metres & Roll Pitch Yaw: $[4.13, 1.32, -8.05]$ degrees). The three input images (downscaled by 2 to 384×512 for processing) are shown in 7.5a, the images formed by reprojecting $\mathcal{S}$ into the image using G_{R,W_0} are shown in 7.5b. The evolution of the the L_2 norm of the vector of NID values for the three cameras is shown in 7.5d and the evolution of the error in each Euler dimension is shown in 7.5e. Finally, 7.5c shows the image formed by reprojecting $\mathcal{S}$ into the cameras using the pose at termination, as can be seen it aligns well with 7.5a as expected because the errors are near zero at termination. The complete optimisation took approximately 250 milliseconds using $|\bar{\mathcal{S}}| = 2^{15} \approx 32.8 \cdot 10^3$ points per camera, spread over two NVIDIA TITAN GPUs.

ΔX	ΔY	ΔZ	ΔRoll	ΔPitch	ΔYaw
$U(\pm 1.5 \text{ m})$	$U(\pm 1.5 \text{ m})$	$U(\pm 1 \text{ m})$	$U(\pm 5^\circ)$	$U(\pm 5^\circ)$	$U(\pm 10^\circ)$

Table 7.1: Distributions of Euler parameters of the offset transforms ΔG applied about ground truth: $\Delta G \cdot G_{R,W}^*$ used for LAPS convergence experiments. Note that $U(\pm\alpha)$ means uniformly distributed across the range: $[-\alpha, \alpha]$.

the robustness of the overall system to failures by independently identifying cases when the optimiser terminated and there was still a significant error in the pose.

7.3.1 Convergence Results

Before considering the solution classifier in Section 7.4, we motivate its necessity by analysing the ‘raw’ localisation performance of the output of the LAPS localiser: $\hat{G}_{R,W}$. To do this, we subsampled the the images in the Oxford A1 dataset (Section 8.1) to have at least a 1 metre separation, equivalent to ≈ 5 Hz resulting in a total of 6645 sets of images. Each set contains images from the three forward facing Ladybug2 cameras downscaled by 50% to 384×512 . These images cover a wide variety of different scene geometries and appearances observed throughout typical driving in Oxford, thus the results generated should provide a fair indication of the overall performance of the system. Using the scene prior generated from the same dataset (using all of the images), so that the ground truth poses are known, we initialised the optimiser at 5 randomly sampled offset from ground truth: $\Delta G \cdot G_{R,W}^*$. Where the Euler parameters of the applied offset transform ΔG were distributed according to Table 7.1.

It is important to note that whilst using the same images that generated the scene prior being localised against will reduce observed differences in illumination between the live images and the prior, it will not eliminate them. This is because only a single image was used to colour each point, but they will typically reproject into multiple images. Due to differences in viewing angles, for distant points accumulated

VO error in the prior, and as all experiments are outdoors and so the background illumination level and auto-exposure control of the camera are not constant, the NID will not be zero at ground truth.

Figure 7.6 shows the histograms of the translation and orientation components of both the initial offset transforms: ΔG (the initial errors) and the errors at termination of the BFGS optimiser after a maximum of 40 iterations: $G_{R,W}^* \cdot \hat{G}_{R,W}^{-1}$. The strong peaks at the origin (zero error) in Figures 7.6c and 7.6d demonstrate that the NID is in general well behaved, with a strong minima at ground truth. They also support the claim that the sample NID figures shown throughout this work, e.g. in Section 4.4.3, are indeed typical. That the peaks in Figures 7.6c and 7.6d do not occur exactly at zero, suggest that there may be some small residual errors in the calibration of the camera relative to the Robot frame. Ideally, the histogram would fall to zero outside the central peak, i.e. no observations have large errors, however there are a small number of observation which have large errors at termination, notably for X, which in the Wildcat frame corresponds to a translation along the vehicle trackline.

Figure 7.7 should be read in conjunction with Figure 7.6, as it drawn from the same data. It shows the cumulative percentage histogram of the number of BFGS optimiser iterations in Figure 7.7a, thus it can be seen that $\approx 80\%$ of the trials terminated in ≤ 35 iterations. As BFGS iterations can involve a variable number of line search steps, Figure 7.7b shows a 2D cumulative percentage histogram of the number of BFGS iterations versus the total time taken by the optimiser in seconds (for the trial). The value of any point in Figure 7.7b is the percentage of the total number of trials that took up to the specified number of iterations and seconds to terminate. For example, we can determine that $\approx 80\%$ of the trials terminated in ≤ 35 iterations and ≈ 0.22 seconds.

The relationship between the plots is that Figure 7.7a represents the sum over the

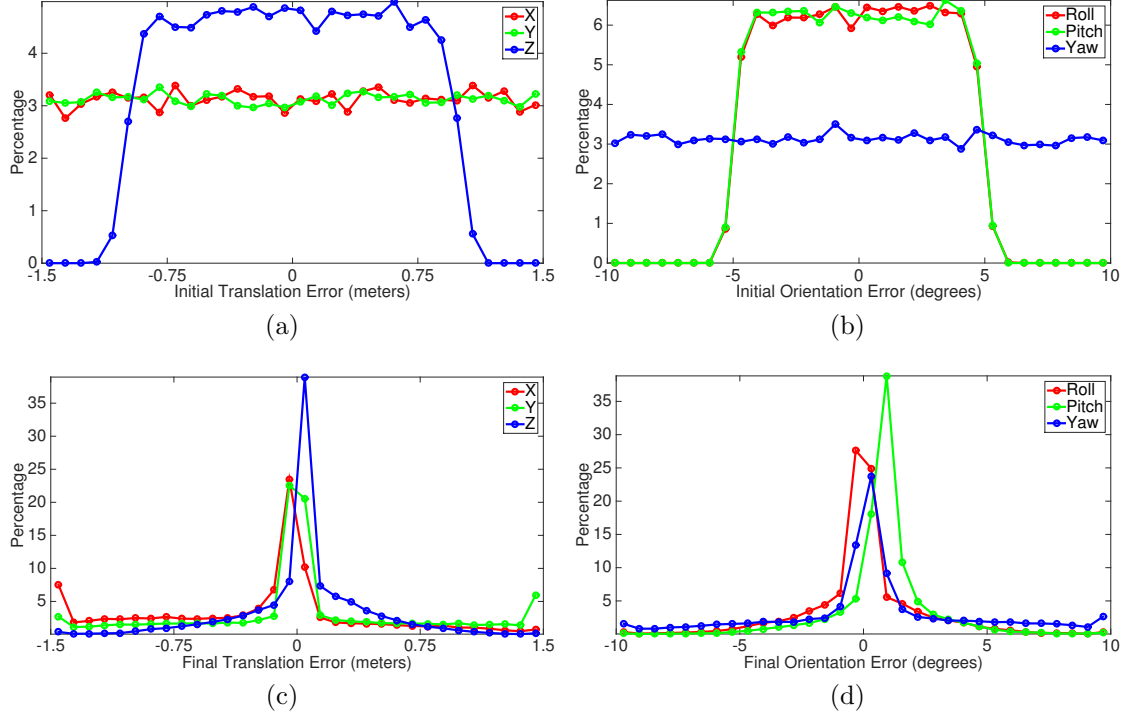


Figure 7.6: LAPS Localiser Convergence Results for Oxford A1 Dataset (Section 8.1), read in conjunction with Figure 7.7. At each of 6645 sets of images separated by ≈ 1 metre across the entirety of the Oxford A1 dataset, equivalent to ≈ 5 Hz, the LAPS optimiser was initialised at $\Delta G \cdot G_{R,W}^*$ 5 times, with ΔG distributed according to Table 7.1. Figures 7.6a and 7.6b show the histograms of the initial errors (ΔG) for translation and orientation respectively, expressed as a percentage of the total number of trials (33225). The BFGS optimiser was allowed to run for a maximum of 40 iterations, Figures 7.6c and 7.6d show respectively the histograms of the translation and orientation components of the error transform at termination: $G_{R,W}^* \cdot \hat{G}_{R,W}^{-1}$ expressed as percentages of the total number of trials.

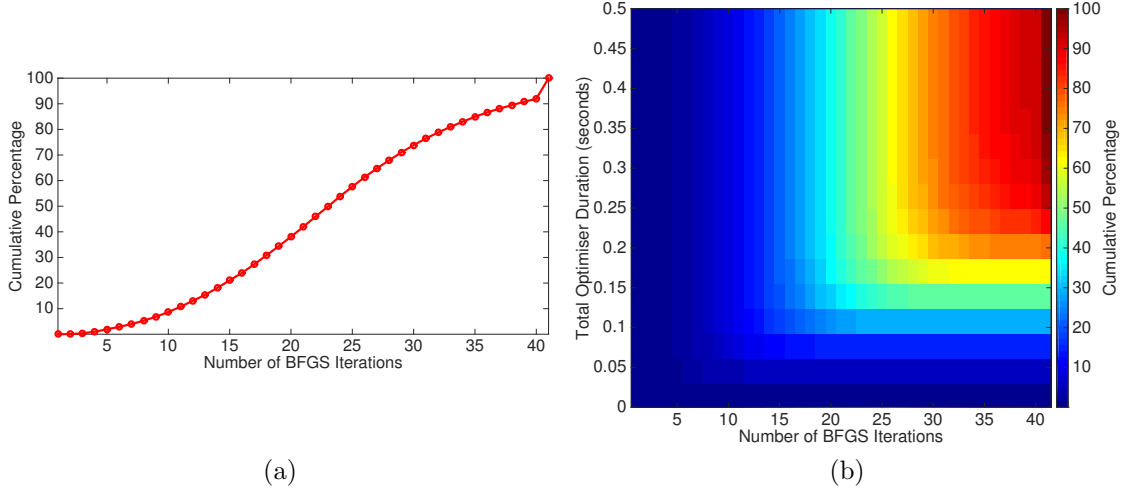


Figure 7.7: LAPS Localiser Convergence Results for Oxford A1 Dataset (Section 8.1), read in conjunction with Figure 7.6. At each of 6645 sets of images separated by ≈ 1 metre across the entirety of the Oxford A1 dataset, equivalent to ≈ 5 Hz, the LAPS optimiser was initialised at $\Delta G \cdot G_{R,W}^*$ 5 times, with ΔG distributed according to Table 7.1. Figure 7.7a shows the cumulative percentage histogram for the number of iterations taken by the optimiser before termination, where for k iterations the value of the curve denotes the percentage of trials (out of the full 33225) which took $\leq k$ iterations to terminate. Figure 7.7b shows the 2D cumulative percentage histogram for the number of iterations vs total optimisation duration for the trial. The value of the 2D histogram for a given $\{\text{num_iterations}, \text{optimiser_duration}\}$ position is the percentage of trials (out of 33225) which took $\leq \text{num_iterations}$ iterations and $\leq \text{optimiser_duration}$ seconds to complete.

rows of Figure 7.7b, i.e. the total number of trials that completed in $\leq k$ iterations, irrespective of how long they took to do so.

The core conclusion from Figure 7.6 and the motivation for our development of the localisation solution classifier described in Section 7.4 is that although our proposed LAPS localiser performs well in a wide variety of cases, it is not infallible. This is not surprising, there are scenes for which the geometry and appearance combine to make our problem ill posed, an example of which is shown in Figure 7.8. Also, the NID also has multiple minima, so if starting outside of the convergence basin of the true minima, the BFGS optimiser will likely converge to the wrong result. Thus, the convergence of the optimiser is not on its own a sufficient measure

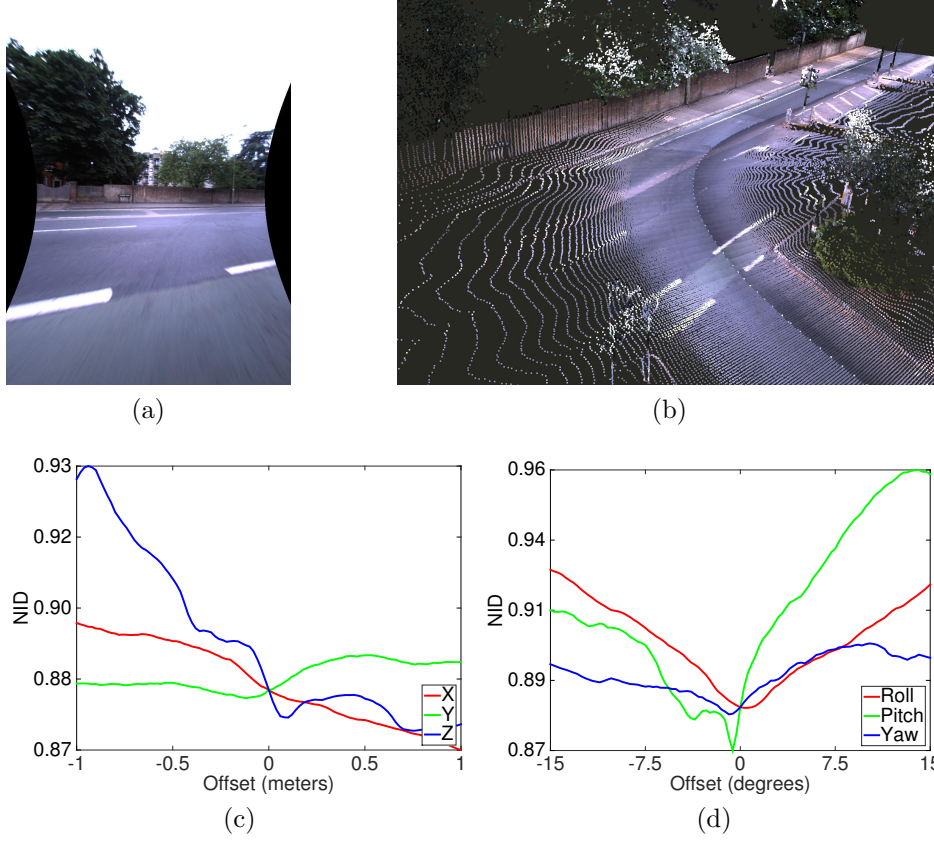


Figure 7.8: Example of a scene induced failure. For the single forward facing camera image in 7.8a and scene prior shown in 7.8b, 7.8c and 7.8d show the axes-sampled lines: traces of the NID evaluated for deltas in each Euler axis independently, about the ground-truth pose. Compared to the clear minima at ground-truth observed for all axes for the EPFL dataset shown previously in Figure 4.11, 7.8c and 7.8d have either no clear minima, offset minima or multiple minima about ground truth. As such, a BFGS optimisation would fail to correctly estimate $G_{R,W}$.

for validity of $\hat{G}_{R,W}$.

Unknowingly incorporating a failed LAPS localisation into our pose estimate is clearly undesirable, particularly because we use our current estimate as the seed pose for future LAPS observations. As a result, even a moderate error combined with a lever arm of integrated odometry between LAPS localisations could push us outside the convergence basin and prevent any future localisations, rendering us lost. Thus it is important that we are able to accurately classify whether $\hat{G}_{R,W}$ represents a valid estimate of the true pose, or a failed localisation.

7.4 Identifying Localisation Failures

A necessary prerequisite of a robust system is that it can identify when it fails; as no system is perfect, failure at some point is inevitable and so it is important to anticipate this eventuality. The results presented in Section 7.3.1 demonstrate that the proposed localisation system works for the majority of cases tested. However, as the NID does not have a single global minima, if the optimiser starts outside the convergence basin for the true minima, or if the geometry and appearance of $\mathcal{S}$ make the problem degenerate, it will converge to an incorrect local minima. This motivates our development of the optimisation solution classifier in the localisation pipeline shown in Figure 7.1 to independently assess whether the $\hat{G}_{R,W}$ returned by the optimiser is a reliable estimate of the true pose.

First we will discuss the generation of the dataset used for training and testing the classifier, then we shall present the choice of feature space in which to perform classification and finally we shall discuss the classifier itself, for which we use an RBF-SVM [34, 10] and present results demonstrating its performance.

7.4.1 Classification Dataset

Before we discuss the development of the classifier and its feature space, we must first describe the dataset used to develop and evaluate it. To generate the classification dataset, similarly to the convergence experiments in Section 7.3.1, we took 600 images equally spaced in time (≈ 2.25 second period) across the entirety of both the Oxford A1 & Oxford A3 datasets (Section 8.1), which cover a wide variety of different scenes. Using the scene prior generated from the dataset so that ground truth is known, for each image we first initialised the optimiser at ground truth ($G_{R,W}^*$) and then at 50 randomly sampled offsets from ground truth: $\Delta G \cdot G_{R,W}^*$ where the Euler parameters of the applied offset transform ΔG were distributed

ΔX	ΔY	ΔZ	ΔRoll	ΔPitch	ΔYaw
$U(\pm 3 \text{ m})$	$U(\pm 3 \text{ m})$	$U(\pm 1.5 \text{ m})$	$U(\pm 10^\circ)$	$U(\pm 10^\circ)$	$U(\pm 30^\circ)$

Table 7.2: Distributions of Euler parameters of the offset transforms ΔG applied about ground truth: $\Delta G \cdot G_{R,W}^*$ to generate the classification datasets. Note that $U(\pm\alpha)$ means uniformly distributed across the range: $[-\alpha, \alpha]$.

according to Table 7.2.

The values for ΔG were deliberately chosen to be large enough that convergence would not occur for the majority of cases. This was because we expect that the space of true positives, that is cases when the optimiser correctly converges to the true pose: $\hat{G}_{R,W} \approx G_{R,W}^*$, is significantly smaller than the space of true negatives. This is because as the NID does not have a unique global minima, outside of the convergence basin about the true minima (where it exists), there will be an arbitrarily large number of local minima to which the optimiser could converge, but only one correct pose.

This resulted in 61200 samples in total, which were then coarsely divided into positive and negative examples based on the magnitudes of the components of the error transform at convergence: $\hat{G}_{R,W} \cdot (G_{R,W}^*)^{-1}$. Initially, the cases where all of the translation components of the error transform were ≤ 0.5 metres and all Euler orientation components were $\leq 5^\circ$ were marked as positive and all others were marked as negative.

The positive set was then further refined, as will be described later in Section 7.4.2 based on the selected classification features. After this process, there were 1652 positive examples and 58825 negative examples which were then randomly split into training and test sets.

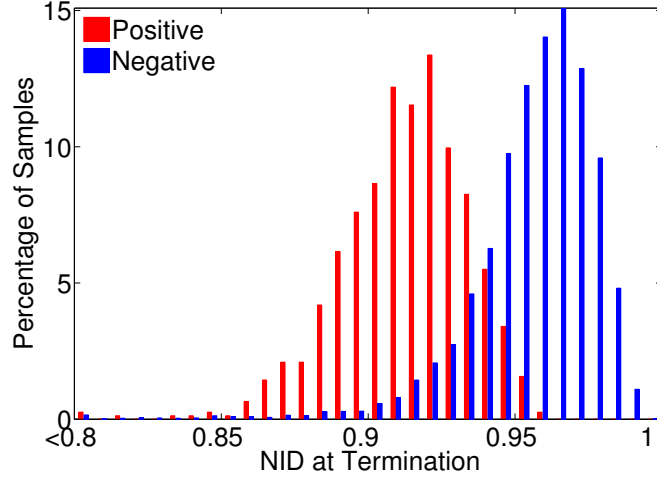


Figure 7.9: Discrimination of NID at optimiser termination for classification training set. The distributions of the values of NID at termination of the optimiser for the positive and negative examples in the classifier training set (Section 7.4.1). Whilst the NID at termination is a useful feature for the classifier, it is not on its own suitably discriminative to completely separate the positive and negative classes and thus robustly classify whether the solution from the optimiser is valid.

7.4.2 Classification Feature Space

Now that we have a classification dataset, an obvious question is whether the NID at the $\hat{G}_{R,W}$ returned by the optimiser, is on its own sufficiently discriminative to identify whether the solution is valid, i.e. close to the true pose $G_{R,W}^*$. Figure 7.9 shows the distribution of NID at termination of the optimiser for the classification dataset. From the figure, it is clear that the NID at termination whilst useful, is not on its own sufficiently discriminative to robustly separate the positive and negative classes and that additional features are required.

The motivation for our new feature is based on the observation that the *shape* of the true minima is distinctive. This characteristic shape is evident in many previous figures in this work, for example in Figures 4.9, 4.10 and 5.9. Indeed, this shape is not only common to different scenes for our task, but also more generally to mutual information based alignment tasks, although in these cases it is a maxima rather than a minima [17, 58].

To capture the shape of the NID about $\hat{G}_{R,W}$ as a feature, we use the same approach as was used to build some of our previous figures used for analysis. We reintroduce the *axes-sampled lines* for $\hat{G}_{R,W}$: $\mathcal{L}_{\hat{G}_{R,W}}$ as the set of NID values sampled for symmetric positive and negative offsets in each of the Euler transform parameters in isolation applied about $\hat{G}_{R,W}$, which were first introduced in Section 4.4.3. To recap, the *axes-sampled lines* are obtained by evaluating the NID at $\Delta G_i \cdot \hat{G}_{R,W}$, where $\Delta \mathbf{g}_i$ is zero for all but the i -th element ($i \in [0, \dots, 5]$), which is sampled evenly across the range $\pm\alpha$. For translation, we use $\alpha = 1$ metre and for orientation $\alpha = 10^\circ$ with 15 samples for each. The resulting six NID traces, one for each Euler parameter, are then collectively the axes-sampled lines for the specified $\hat{G}_{R,W}$. Some example axes-sampled lines for randomly selected positive and negative examples from the classification dataset are shown in Figure 7.10.

It is clear from Figure 7.10 that the characteristic minima evident across all axes in Figure 7.10a is a strong discriminator when compared to the negative training set samples in Figure 7.10b, confirming our intuition that the axes-sampled lines represent a strong indicator of the quality of the solution.

The functional form of the characteristic minima is not clear or intuitive, as both the NID and mutual information involve sums of $p \log(p)$ terms which do not simplify and its common form is not widely discussed in the literature. It is also not well approximated by the forms of either the Gaussian or Cauchy distributions.

We propose the extended classification feature space: $\left[\mathcal{L}_{\hat{G}_{R,W}}, \text{NID}_{\hat{G}_{R,W}} \right] \in \mathbb{R}^{91 \times 1}$ by simply concatenating the six NID traces each containing 15 samples constituting the axes-sampled line for each transform parameter in $\mathcal{L}_{\hat{G}_{R,W}}$ (each of which are separately renormalised) into a single vector. The final element in the feature space is then the NID at termination of the optimiser: $\text{NID}_{\hat{G}_{R,W}}$. Strictly speaking, it is actually the sum of $\text{NID}_{\hat{G}_{R,W}}$ for each camera normalised by the number of cameras, to allow generalisation of the classifier between systems with different numbers of

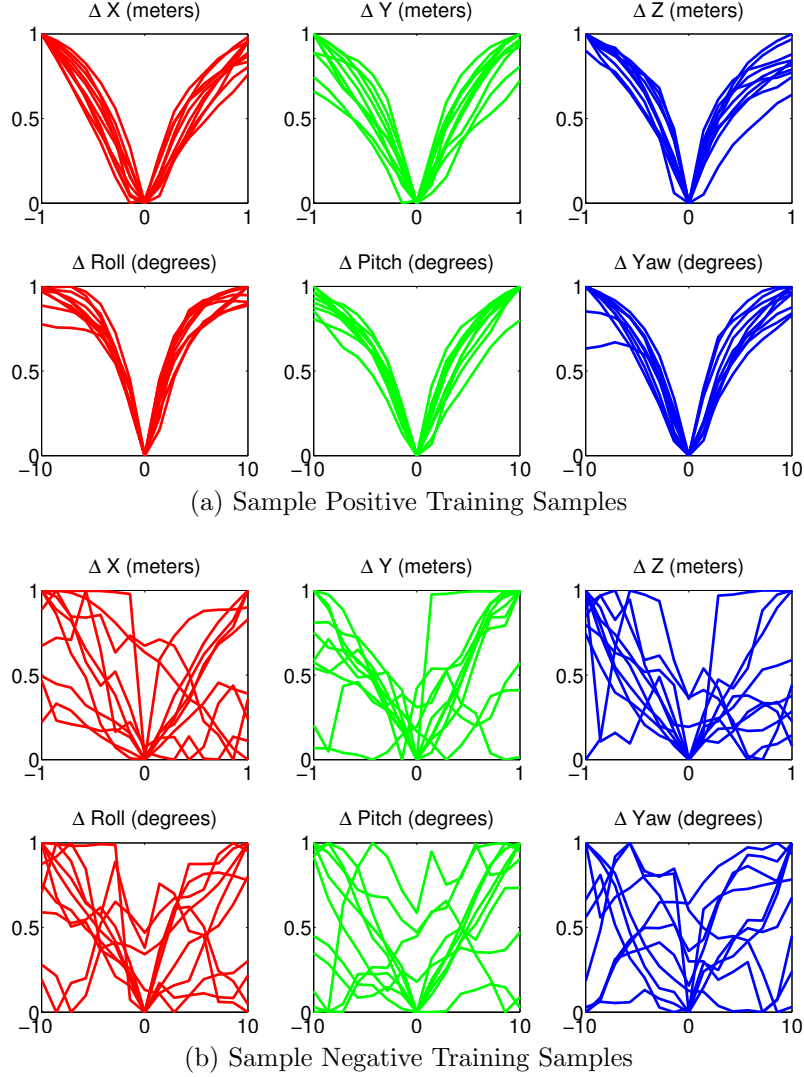


Figure 7.10: Sample axes-sampled line features from classification training data. Ten randomly selected positive (7.10a) and negative (7.10b) samples of the axes-sampled line classification features. Each set of axes-sampled line features is the set of the NID values sampled for 15 offsets along each Euler transform parameter in insolation, about the termination point of the BFGS optimiser: $\hat{G}_{R,W}$. As evident in the data, the presence of the characteristic minima sampled about all axes is a strong discriminator between the positive and negative training sets. The normalisation of each axes-sampled line is performed to improve the generalisation of the training data and thus classification performance [10] and furthermore, as the NID at termination is also used as an additional feature, an ‘absolute’ comparator is still available to the classifier.

cameras. The renormalisation which occurs in $\mathcal{L}_{\hat{G}_{R,W}}$ is a widely used technique for improving the performance of SVM classifiers [10]. But it also reflects our desire to use $\mathcal{L}_{\hat{G}_{R,W}}$ to represent the *shape* of the minima, as the relative magnitude can change for different scenes and configurations (Figure 5.9), but we posit that the shape is preserved. To further emphasise this distinction, as mentioned in Section 7.4.1, we performed a final manual pruning of the positives in the classification dataset to keep only those which displayed strong examples of the minima, centered at an offset of zero, across all parameters. The presence of the NID as the final feature means that an ‘absolute’ comparator between different samples still exists in the feature space, even though the traces in $\mathcal{L}_{\hat{G}_{R,W}}$ have been individually normalised.

Our rationale for using only the value of the NID in $\mathcal{L}_{\hat{G}_{R,W}}$ and for using 15 samples per axis are both influenced by performance considerations, as we wish to be able to maintain reasonable real-time performance for the complete localisation system. Figure 7.4 shows that computing the derivatives of $p_{\mathcal{IS}}$ is the dominant computational cost (60%) and is only required for computing the gradient of NID, thus we can minimise the performance penalty incurred the classification stage by using just the NID.

The use of 15 samples per axis for a total of 90, was chosen as a balance between providing sufficient detail to capture the shape of the minima, whilst minimising the computation cost incurred. On average, evaluating $\mathcal{L}_{\hat{G}_{R,W}}$ with this configuration takes ≈ 50 milliseconds per camera.

7.4.3 SVM Classification of Optimiser Solution

When considering the choice of classifier, a key consideration for our problem is that it be fast to evaluate, as we wish to minimise the overhead of its presence. We also require a classifier capable of modelling the expected nonlinear relations between the features and the labels, which led us to select an RBF-SVM classifier [34], for

which we use the implementation in libSVM⁴ [10].

When training an SVM, the general minimisation to be solved is given by Equation (7.7) [10],

$$\begin{aligned}
 \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_i \xi_i \\
 \text{subject to} \quad & y_i (\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \\
 & \xi_i \geq 0
 \end{aligned} \tag{7.7}$$

where $\mathbf{x}_i \in \mathbb{R}^{n \times 1}$ and $y_i \in [-1, 1]$ represents the feature space and label for the i -th training sample respectively. The SVM is parameterised by the weight vector $\mathbf{w}$, b and the Kernel function: $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$. The C parameter is a penalty term which weights the slack variables ξ_i , which represent the misclassification margin. An RBF-SVM uses the kernel function given in Equation (7.8),

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \quad \gamma > 0 \tag{7.8}$$

and thus has two hyperparameters to be tuned: C and γ also called the length-scale which controls the influence of one sample on another. If γ is large then the influence is small, conversely for small γ the influence is large.

In order to select the ‘best’ hyperparameters for our problem, we performed a grid search with 5-fold cross validation on 80% of the classification dataset to explore the space of reasonable values for the hyperparameters using the average $F_{0.5}$ measure across all folds as the ranking, the results of which are shown in Figure 7.11. We chose the $F_{0.5}$ measure as a comparator because for our problem we care more about precision than recall. As we fusing the final solution with odometry, false

⁴<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

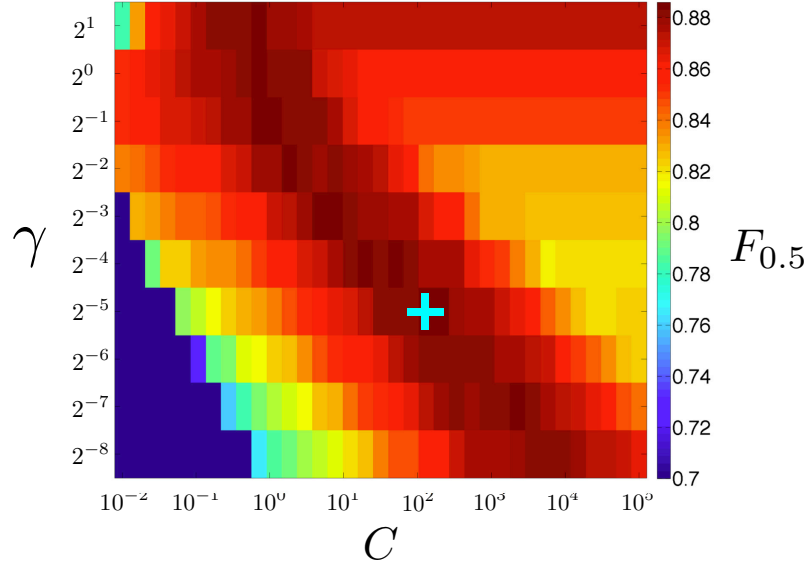


Figure 7.11: Average $F_{0.5}$ for grid search over RBF-SVM hyperparameters with 5-fold cross validation. The average $F_{0.5}$ over the 5-folds for a grid search over the RBF-SVM hyperparameters γ and C . The search space was logarithmic base 2 for γ and logarithmic base 10 for C . The values selected: $\gamma = 2^{-5}$ & $C = 150$ which yielded an average $F_{0.5} = 0.886$ are highlighted on the figure with a cyan cross and were found through an additional local search about the maxima returned from the grid search.

negatives: failing to incorporate a valid LAPS localisation, are less problematic than false positives: incorporating an incorrect localisation, as the latter may push the resulting solution outside of the convergence basin and prevent future localisations, whilst the former means relying on odometry for a longer period. We note that this is not to say that recall is not important, merely that it is less important than precision. As the odometry is only assumed to be accurate over short distances, integrating it for long periods (implied by a low recall) is not desirable either.

From Figure 7.11, there is clearly a band of hyperparameter pairs that give similar performance according to the average $F_{0.5}$ measure. We selected: $\gamma = 2^{-5}$ & $C = 150$, which maximised $F_{0.5}$ from a local search about the coarse maxima returned from the grid search in Figure 7.11. This also reflects our bias towards a smaller γ implying greater generalisation of the support vectors and a larger C to penalise more heavily any misclassifications.

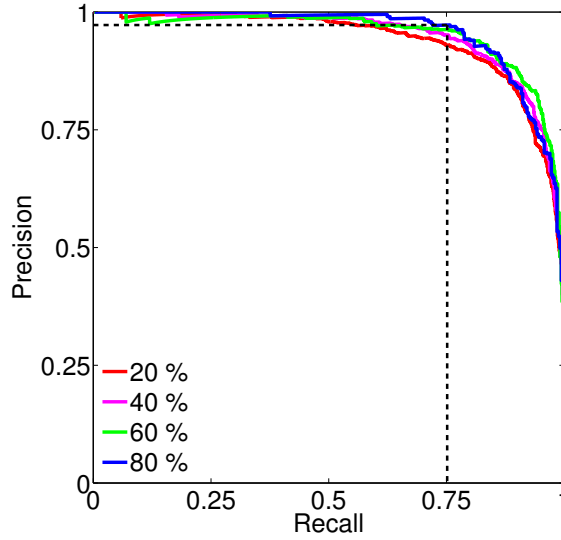


Figure 7.12: Precision - Recall (P-R) curves for the optimiser solution classifier for varying training percentages. Each P-R curve corresponds to the RBF-SVM with hyperparameters $\gamma = 2^{-5}$ & $C = 150$ trained on a different randomly selected percentage of the classification dataset and tested on the remainder. The decision boundary used in this work is shown as a dashed black line and was chosen from the 80% curve, and results in a precision of 97% and a recall of 75%

Using the selected hyperparameters, the Precision - Recall (P-R) curves for an RBF-SVM trained on varying percentages of the classification dataset and tested against the remainder are shown in Figure 7.12, along with the position of the decision boundary used in this work. That the classifier performs well, and that the percentage of training data makes little difference to the shape of the P-R curve suggests that the positive and negative classes are well separable using the selected features. It also suggests, at least within the classification dataset, that the SVM generalises well, a good indicator of the performance on new data.

The decision boundary selected for this work was chosen as the point on the 80% P-R curve in Figure 7.12 which returned a recall of 75%, which resulted in a precision of 97%. The choice of this boundary is a trade-off between our conflicting desires for high precision, to prevent incorporation of any failed localisations which would prevent the system from relocalising, and a high recall to keep the distances between

accepted LAPS localisations, over which we rely on odometry, short. An additional motivation for high recall is that the final stage in the pipeline which fuses odometry and LAPS poses will better accommodate the occasional failed LAPS localisation, provided that the errors are not too large, when there are more LAPS observations to constrain the result. It is also important to remember that the results in Figure 7.12 are really only indicative of the overall expected performance, as the classification dataset does not exhaustively capture all scenes and conditions in which the system might be used.

The complete classification pipeline, which is run live in the localisation loop using the trained RBF-SVM is shown in Figure 7.13. As mentioned in Section 7.4.2 the evaluation of $\mathcal{L}_{\hat{G}_{R,W}}$ is by far the most expensive component taking approximately 50 milliseconds per camera. For our test system with two GPUs and three cameras, two cameras are run serially on one GPU and so the total overhead of the classification pipeline is approximately 100 milliseconds. This is considerable and it pushes the realistically achievable rate of the LAPS localiser to ≤ 5 Hz from ≈ 10 Hz without any solution classifier.

To demonstrate the effectiveness of the classifier, we present again the LAPS localisation convergence results from Section 7.3.1 and Figure 7.6, but now for only those trials classified as valid at termination by the SVM classifier, shown in Figure 7.14. In total, 13207 trials out of 33225, or $\approx 40\%$ were classified as valid. The key point to note is that *all* trials classified as valid have small errors, as can be seen from Figures 7.14c and 7.14d, in contrast to the equivalent plots for the complete set of trials shown in Figures 7.6c and 7.6d. Thus the classifier is very effectively addressing the problem for which it was designed.

Although the classifier adds computational cost, it is important to reject failed localisations as our localisation problem consists of tracking against a scene prior, and during normal operation, the current output of the system is used as the seed

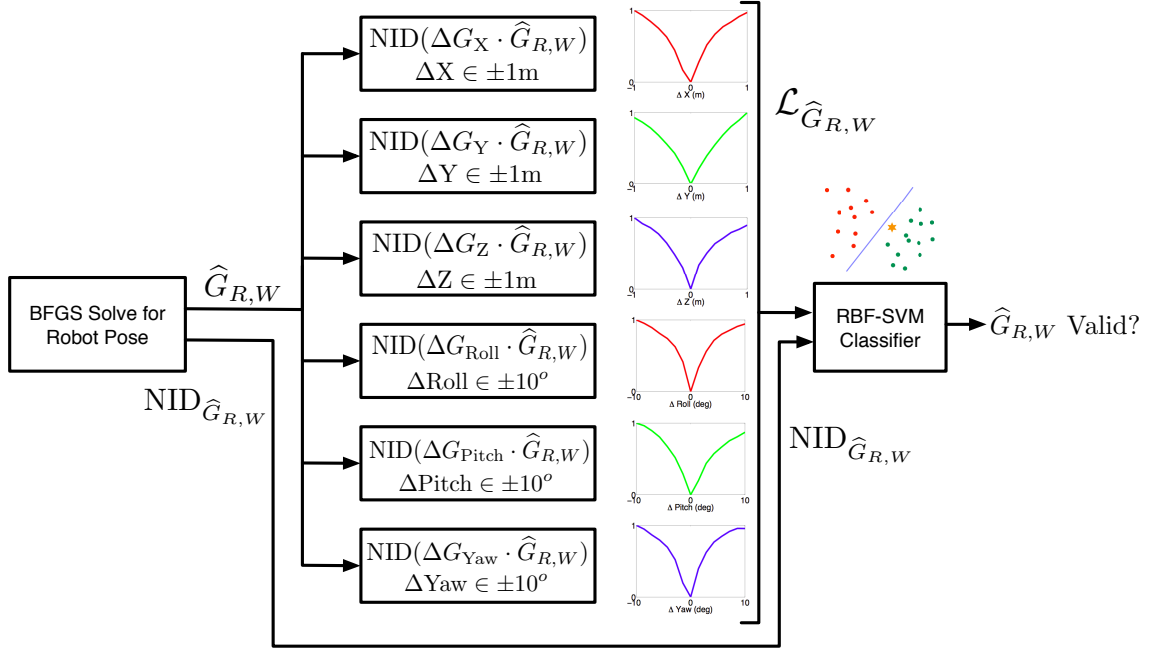


Figure 7.13: Optimiser Solution Classifier Pipeline. The inputs to the pipeline are the estimated pose: $\hat{G}_{R,W}$ and the NID at evaluated $\hat{G}_{R,W}$: $NID_{\hat{G}_{R,W}}$ at termination of the BFGS optimiser. The axes-sampled lines $\mathcal{L}_{\hat{G}_{R,W}}$ are then evaluated about $\hat{G}_{R,W}$ and combined with $NID_{\hat{G}_{R,W}}$ to form the complete feature vector. The RBF-SVM then classifies the localisation as either positive: $\hat{G}_{R,W} \approx G_{R,W}^*$ or negative. Only poses classified as positive are passed to the complete localisation pipeline which fuses LAPS and odometry poses.

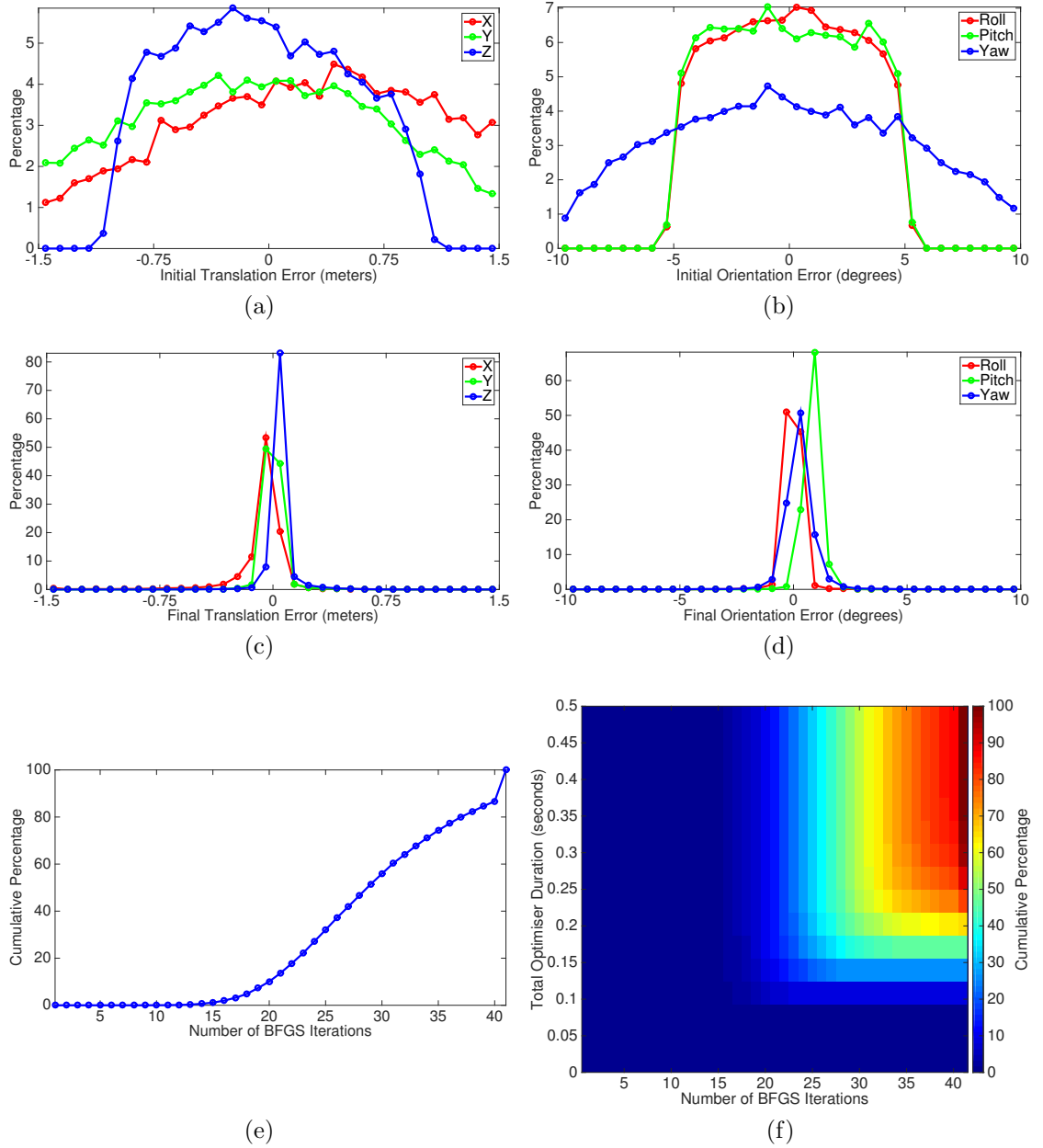


Figure 7.14: LAPS Localiser Convergence Results for Oxford A1 Dataset (Section 8.1) classified as valid by SVM. With reference to Figure 7.6, the plots show the same quantities, but for the subset of the trials shown in Figure 7.6 which the SVM classifier classifies as valid at termination. This equates to 13207 trials out of 33225, or $\approx 40\%$. Note from Figures 7.14c and 7.14d that as desired, no trials classified as valid have large errors at termination, contrast with Figures 7.6c and 7.6d. The tapering evident in X, Y & Yaw in Figures 7.14a and 7.14b as contrasted to the initial errors for the full set of trials in Figure 7.6 shows that larger errors in these axes resulted in reduced likelihood of converging to a valid, as reported by the classifier, solution.

pose for the next iteration. Thus, incorporating failed localisations which push the seed poses for subsequent observations outside the convergence basin around the true pose would cause the system to fail.

As noted earlier, when the system is first initialised or after it has *independently* declared itself lost, defined as having failed to produce a LAPS localisation which was classified as valid by the classifier for a period of time (5 seconds in the results presented in Chapter 8), it requires an external ‘oracle’ to provide an approximate (re)initialisation pose. The system cannot accurately track the robot’s location against the scene prior if it does not know roughly where it is, as otherwise which component of $\mathcal{S}$ should even be used is unknown. As the ‘oracle’ does not need to be available continuously, only when the system is started or declares itself lost and the robot must be operating in a previously mapped area, there are many possible ‘oracle’ sources besides GPS. Principally, as we require that the robot to be localised has a collection of calibrated monocular cameras (as did the survey vehicle which built the scene prior), vision-based topological localisation systems such as [15, 56, 69, 87, 75] would be a natural choice for an ‘oracle’. Particularly those robust to large changes in illumination and seasonal changes [69, 87, 75].

We stress that it is the LAPS system itself which *independently* identifies when it is lost using the solution classifier, without requiring an external ‘oracle’. Without the classifier, the system would not be able to reliably determine independently if this had happened, and thus whether the pose it is outputting is accurate. Hence we argue that the classification stage whilst computationally expensive, is not only desirable but necessary for the overall system to be robust.

7.5 Fusing LAPS with Odometry

The final stage in the localisation pipeline in Figure 7.1 is the system which fuses the LAPS localisations classified as valid solutions with the odometry source to produce the final output pose. Comparably to a standard GPS aided INS system, we have low update rate, noisy position estimates, in our case from LAPS in the INS case from GPS, which represent the best estimate of the pose at the specified time. The process that generated them did so without any consideration of the dynamical constraints of the robot, or its locally observed motion. In addition, we have a high update rate source providing purely relative local motion information, in our case from odometry, either VO or wheel odometry, and in the INS case from an IMU (and also potentially wheel odometry). Our goal is to combine these two sources to produce a single pose output at a rate comparable to that of the odometry, but which is accurately referenced in an external frame.

In the case of an INS, this external frame is a single georeferenced frame (e.g. latitude & longitude or UTM). In our case there are different external frames for different locations, as we are localising against the scene prior $\mathcal{S}$ which is composed of segments, each of which is defined in their own frame as described in Section 6.3. Although at any given instant we typically use an $\mathcal{S}$ built from multiple adjacent segments, we only ever localise against the frame of one segment (the closest) at any give time.

There are a variety of different approaches that have been used for this problem, the most prevalent being the nonlinear variants of the Kalman Filter [62]: the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF) [37, 109], which form the basis of the vast majority of commercially available GPS aided INS systems [94, 82]. An EKF was also used in [122], recent work using the same formulation as our work, but using normalised mutual information and constrained

to $\mathbb{SE}(2)$, to combine their image registration measurements with those from a high-grade IMU from an Applanix POS-LV 420 INS.

In contrast to these discrete time methods, recent work [28, 110, 53] has focussed instead on how to perform continuous time pose estimation. As noted in [28], discrete time formulations where the state expands to include the pose at every discrete time, whilst convenient, become intractably large. Intuitively, continuous time representations aim to provide a solution to this by estimating a lower dimensional parameter set which are then weighted by basis functions defined in time to produce the estimated pose at any arbitrary time in the period covered by the data.

In Gaussian Process Gauss-Newton (GPGN) introduced in [110], a Gaussian Process (GP) is used to estimate the pose of the robot within a local window and was used in [111] to perform visual odometry with a scanning lidar. The methods developed in [28, 53] were introduced to address the same problem of motion distortion, albeit for rolling shutter cameras in which each column in the image is captured sequentially. Both methods estimate control points of cubic B-Splines which represent the pose of the robot within a window parameterised by time. This allows estimates of the first and second derivatives of the pose to be computed quickly, which is particularly useful when incorporating accelerometer and gyro observations. In [28], two B-Splines are estimated separately, one for translation and one for rotations, whilst in the Spline Fusion algorithm [53], a single cubic B-Spline in the $\mathfrak{se}_3$ Lie Algebra space is estimated instead, which is more desirable.

In this work, we chose to use the Spline Fusion algorithm [53] in $\mathbb{SE}(3)$, taking as inputs the LAPS localisations and a relative motion odometry source, either from VO or from wheel odometry. The use of odometry information in this form rather than an IMU is attractive as the purely relative nature of our scene prior reference frames complicate the use of a gravity vector. An additional benefit is that all modern cars are capable of outputting wheel odometry, but an IMU of

sufficient grade for pose estimation is not yet standard equipment. In our case, as the Wildcat does not supply wheel odometry information, we use our VO system instead, although we have found empirically on one of our other platforms that the relative motion estimates from wheel odometry were broadly comparable with those from VO (in the plane) over the length of our scene prior segments (30m).

The continuous time formulation of Spline Fusion allows us to naturally handle the fact that the odometry source will not be synchronised to the cameras generating images for LAPS localisations in addition to the variable period of the LAPS localisations themselves. As only localisations classified as valid solutions will be passed to Spline Fusion and the time between valid LAPS localisations will be variable and is not knowable a priori. Finally, the non-parametric nature of Spline Fusion as solved by a nonlinear least squares optimisation allows us to easily formulate the problem as a sliding window over the last τ seconds of data and to transform the reference frame of the spline and observations to those of consecutive segments in the scene prior as the robot moves through it.

We emphasise that we are *not* reevaluating the LAPS objective for each localisation whilst optimising the spline parameterising the robot’s pose in Spline Fusion. We treat each accepted LAPS localisation as an immutable observation against which the error relative to the pose estimated by the spline at the same timestamp is computed and used as a residual in the optimisation. This is analogous to how GPS observations are treated implicitly in a Kalman Filter framework. Our arguments for this approach are firstly of practicality, reevaluating the NID and its gradient for a large number of observations simultaneously would have to be serialised per camera on any available GPUs and would require storing more images and a larger scene prior on the GPU. Secondly, the use of the BFGS solver means that any accepted LAPS localisation should already be a local minima, and in order for it to be accepted by the solution classifier it must have the local characteristics

of the true pose.

In the following sections, we first describe in detail the Spline Fusion algorithm from [53] and then detail the specific residuals used when fusing LAPS localisations with odometry.

7.5.1 Spline Fusion

The Spline Fusion algorithm presented in [53] estimates the continuous pose of a robot using a cubic B-Spline parameterised by time and defined in the $\mathfrak{se}_3$ Lie Algebra space. With reference to our previous usage of cubic B-Splines in Sections 2.2 and 5.3, in Spline Fusion the cubic B-Spline weight function is parameterised by time: $\beta() = \beta(t)$ and the control points are poses defined in $\mathbb{SE}(3)$: $G_{W,k}$ in an external frame W . Intuitively, the pose of the robot at any continuous time t : $G_{W,S}(t)$ is obtained via a weighted sum of the control point poses based on the distance in time from the control point time: t_k to the sample time.

However, we cannot simply perform a weighted sum of the matrices representing $G_{W,k}$, as the result would not be a valid $\mathbb{SE}(3)$ pose. Hence the weighting is applied to the $\mathfrak{se}_3$ representation of the poses and the addition for a standard vector space is substituted for multiplication, its analogue for $\mathbb{SE}(3)$, to combine the weighted form of the transforms. Thus the result of evaluating the spline is also in $\mathbb{SE}(3)$. Formally, starting from the general cumulative form [38] of an m -th degree B-Spline given in Equation (2.13),

$$\mathbf{p}(u) = \mathbf{p}_0 \cdot \beta_{0,m} + \sum_{k=1}^n \tilde{\beta}_{k,m}(u) \cdot (\mathbf{p}_k - \mathbf{p}_{k-1}) \quad (2.13)$$

the cubic B-Spline equation used in Spline Fusion in $\mathbb{SE}(3)$ is given by Equation (7.9),

$$G_{W,S}(t) = \exp\left(\tilde{\beta}_0(t) \cdot \log(G_{W,0})\right) \prod_{k=1}^n \exp\left(\tilde{\beta}_k(t) \cdot \log(G_{W,k-1}^{-1} G_{W,k})\right) \quad (7.9)$$

where we have dropped the second subscript denoting the degree from β for brevity as we are explicitly considering only cubic B-Splines for which $m = 3$.

The matrix exponential and matrix logarithm in Equation (7.9) are used to map: $\mathfrak{se}_3 \mapsto \mathbb{SE}(3)$ and $\mathbb{SE}(3) \mapsto \mathfrak{se}_3$ respectively, where $\mathbb{SE}(3) \in \mathbb{R}^{4 \times 4}$ and $\mathfrak{se}_3 \in \mathbb{R}^{4 \times 4}$. The $\mathfrak{se}_3 \in \mathbb{R}^{4 \times 4}$ representation can also be transformed from/to a minimal $\mathbb{R}^{6 \times 1}$ form using the hat: $(\cdot)^\wedge : \mathbb{R}^{6 \times 1} \mapsto \mathbb{R}^{4 \times 4}$ and inverse hat: $(\cdot)^\vee : \mathbb{R}^{4 \times 4} \mapsto \mathbb{R}^{6 \times 1}$ operators. The control point differences: $\mathbf{p}_k - \mathbf{p}_{k-1}$ in the general cumulative form have been replaced with the equivalent form in $\mathbb{SE}(3)$: $G_{W,k-1}^{-1} G_{W,k}$, the transform between control points.

Equation (7.9) can be simplified further, first by considering that the control points correspond to equally spaced points in time along the spline, with a period of Δt , with the time of the k -th control point $G_{W,k}$ denoted by t_k . This does not limit the generality of the approach, as the control point poses are going to be estimated via a nonlinear least squares optimisation to satisfy the observed pose constraints of the vehicle at arbitrary (potentially non-periodic) times from the observed sensor data. This allows the spline parameter to be updated to: $s(t) = \frac{t-t_0}{\Delta t}$, in which the control point times map to integer coordinates: $s_k \in [0, \dots, n]$, allowing the use of the fractional coordinate B-Spline weight equations given in Section 2.2: $\beta(s(t) - s_k)$. Second, by the local modification property of B-Splines, for cubic B-Splines (at most) 4 control points have non-zero weights, for an observation at time t , these are: $[t_{k-1}, t_k, t_{k+1}, t_{k+2}]$, where $t \in [t_k, t_{k+1})$. Using these simplifications, Equation (7.9) becomes Equation (7.11), where $\tilde{\beta}_0(s) = 1 \forall s$ and so has been omitted.

$$\begin{aligned}
 G_{W,S}(t) &= G_{W,k-1} \prod_{j=1}^3 \exp \left(\tilde{\beta}_j(s) \cdot \log(G_{W,k+(j-1)-1}^{-1} \cdot G_{W,k+(j-1)}) \right) \\
 &\Rightarrow \Omega_i \equiv \log(G_{W,k+i-1}^{-1} \cdot G_{W,k+i}) \tag{7.10}
 \end{aligned}$$

$$= G_{W,k-1} \prod_{j=1}^3 \exp \left(\tilde{\beta}_j(s) \cdot \Omega_{j-1} \right) \tag{7.11}$$

We emphasise that it is important to understand that the spline will *not* (in general) interpolate the control point poses $G_{W,k}$, just like any standard B-Spline. The control points are best thought of as more abstract quantities to be estimated via an optimisation which through a weighted combination describe the pose of the robot. They are *not* directly the pose of the robot at the observed times. The resulting Spline Fusion formulation is represented graphically in Figure 7.15.

One of the principal advantages to this approach in addition to providing a truly continuous pose estimate, is that it is C_2 continuous and thus the first and second derivatives of the pose of the robot with respect to time can be easily and quickly computed. As noted in our previous discussions of B-Splines, the derivatives of the B-Spline weight functions can be computed in closed form. Using the matrix exponential identity: $\frac{\partial \exp(\alpha X)}{\partial \alpha} = X \exp(\alpha X) = \exp(\alpha X) X$, the hat operator for $\mathfrak{se}_3$: $\hat{\cdot}: \mathbb{R}^6 \mapsto \mathbb{R}^{4 \times 4}$ [103] and the abbreviation in Equation (7.12),

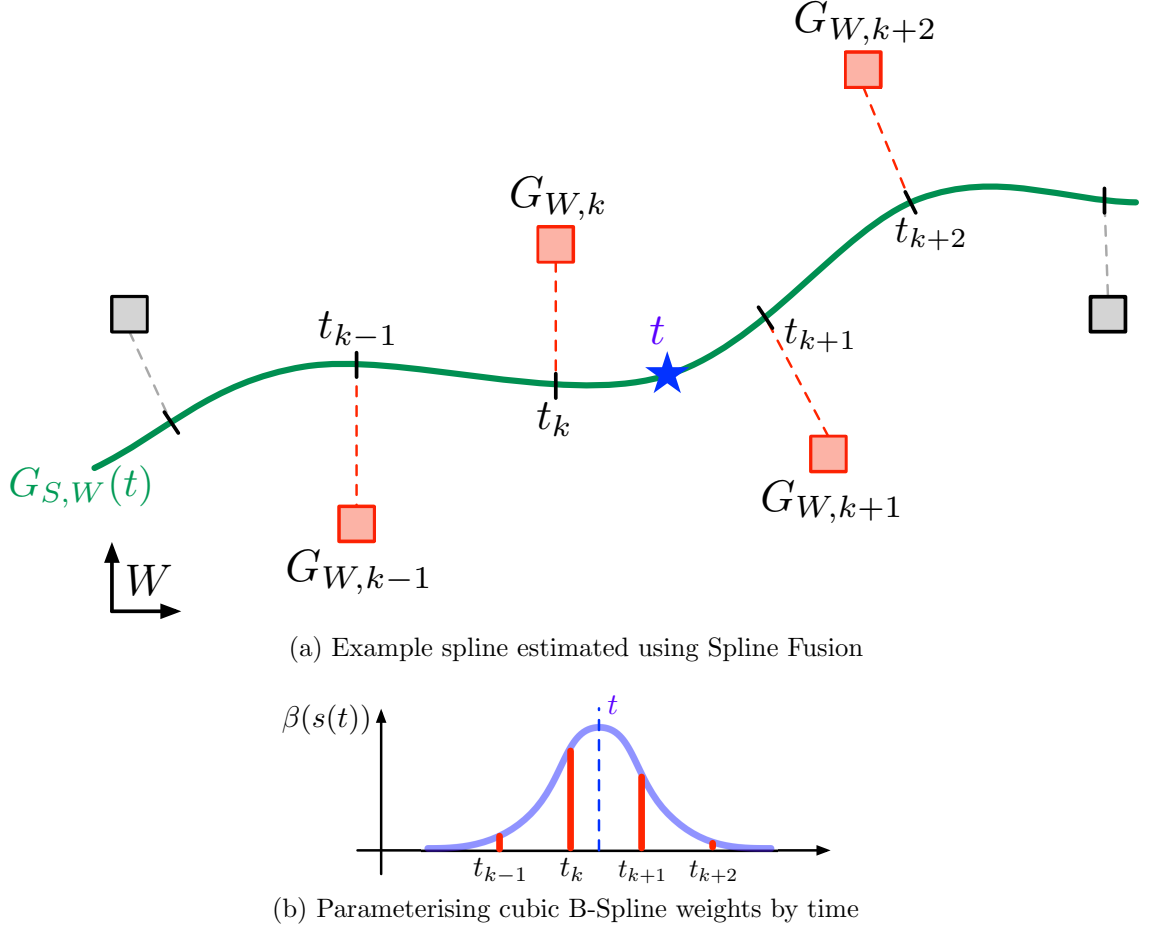


Figure 7.15: Conceptual example of Spline Fusion. A conceptual example of a spline estimating the continuously defined pose: $G_{W,S}(t)$ via Spline Fusion is shown in Figure 7.15a. The control points are shown as squares, the k -th control point is parameterised by a $\mathbb{SE}(3)$ pose defined in an external frame W : $G_{W,k}$ and influences the spline at a time t_k . For any query time t , four control points have non-zero weights: $\beta(s(t))$ according to the standard cubic B-Spline weight function parameterised by time shown in Figure 7.15b. These control points are defined to be those which influence the spline at times: $[t_{k-1}, t_k, t_{k+1}, t_{k+2}]$ where $t \in [t_k, t_{k+1})$ and are highlighted in red in Figure 7.15a. An important point to note is that the spline does *not* (in general) interpolate the control point poses $G_{W,k}$, just like any standard B-Spline does not interpolate its control points. As such, they can perhaps be more easily thought of as abstract parameters $\in \mathbb{SE}(3)$ to be estimated based on the input observations, a weighted combination of which defines the pose of the robot.

$$A_j \equiv \exp \left(\tilde{\beta}_j(s) \cdot \Omega_{j-1} \right) \quad (7.12)$$

$$\begin{aligned} \dot{A}_j &= \frac{\partial A_j}{\partial t} = A_j \hat{\Omega}_{j-1} \frac{\partial \tilde{\beta}_j(s)}{\partial t} \\ &= A_j \hat{\Omega}_{j-1} \dot{\tilde{\beta}}_j(s) \end{aligned} \quad (7.13)$$

$$\begin{aligned} \ddot{A}_j &= \frac{\partial^2 A_j}{\partial t^2} \\ &= \dot{A}_j \hat{\Omega}_{j-1} \ddot{\tilde{\beta}}_j(s) + A_j \hat{\Omega}_{j-1} \ddot{\tilde{\beta}}_j(s) \end{aligned} \quad (7.14)$$

the equations for the derivatives of Equation (7.11) can be written as Equations (7.16) and (7.17),

$$G_{W,S}(t) = G_{W,k-1} A_1 A_2 A_3 \quad (7.15)$$

$$\frac{\partial G_{W,S}(t)}{\partial t} = G_{W,k-1} \left(\dot{A}_1 A_2 A_3 + A_1 \dot{A}_2 A_3 + A_1 A_2 \dot{A}_3 \right) \quad (7.16)$$

$$\frac{\partial^2 G_{W,S}(t)}{\partial t^2} = G_{W,k-1} \left(\begin{aligned} &\ddot{A}_1 A_2 A_3 + A_1 \ddot{A}_2 A_3 + A_1 A_2 \ddot{A}_3 + \\ &2 \left(\dot{A}_1 \dot{A}_2 A_3 + \dot{A}_1 A_2 \dot{A}_3 + A_1 \dot{A}_2 \dot{A}_3 \right) \end{aligned} \right) \quad (7.17)$$

following the form of [53]. It should be noted that as $\tilde{\beta}_j(s)$ is a function of $s = s(t)$, from the chain rule, there will be an additional $\frac{1}{\Delta t}$ term multiplying the first derivative weights from Section 2.2 and an equivalent $\frac{1}{\Delta t^2}$ term multiplying the second derivative weights.

Now that we have obtained equations for the evaluating the pose of the robot and its derivatives at a query time t according to the spline, we can use these equations to form error residuals in a nonlinear least squares optimisation to estimate the control point poses: $G_{W,k}$. Note that for any residual representing the error of the spline relative to a sensor observation with a corresponding timestamp, which set

of four control points used to compute the pose according to the spline and their associated weights are fixed throughout the optimisation. This is an important benefit to parameterising the spline by time, as the timestamps of the observation and the control points are invariant and thus so are the B-Spline weights $\beta(s(t))$. Furthermore, by the local modification property of B-Splines any single residual involving a single pose or associated derivative on the spline affects only four control points and thus the resulting problem is sparse.

The choice of the control point period Δt defines the cut off of the low-pass filter represented by the spline. A small Δt means the resulting spline will be capable of representing high frequency motions, but at the cost of estimating more control points and thus of requiring more observations to constrain the spline. Conversely a large Δt results in a spline that will smooth out any high frequency motions. For the cases for which Spline Fusion was originally demonstrated of visual odometry with a free moving camera and camera-IMU calibration, a small Δt is required as high frequency camera motions are likely. However, for our problem where the robot whose pose to be estimated is a full size Ackermann vehicle weighing approximately 2 tonnes, a large Δt is appropriate as the vehicle is not (safely) capable of high frequency movements and thus any observed are sensor noise that we wish to smooth out.

A distinction between the Spline Fusion algorithm and the GPGN approach from [110] which was not discussed in the original work [53], is that of estimating covariances of $G_{W,S}(t)$ and its derivatives. A benefit of the GPGN method is that the covariances of the estimates can be directly obtained from the GP. In the case of Spline Fusion, an estimate of the covariances of the control point poses $G_{W,k}$ can be obtained at the termination of the optimiser using the Gauss-Newton approximation: $\Sigma = (J^T \Omega J)^{-1}$. However, it is not clear how these should be combined to yield the covariance of the pose estimate at a sample time t . Extending the approach

from [111] for linear interpolation of poses is problematic as the cancellations and simplifications via the Adjoint operator for the linear case do not occur for Equation (7.11). A conservative approach would be to assume independence between the control point poses, which is by definition not the case. Under this assumption, $\Sigma_{G_{W,S}(t)}$ would be computed as the weighted sum of the diagonal blocks from the covariance matrix corresponding to the $G_{W,k}$ with non-zero weights at time t , using the same $\tilde{\beta}_j(s)$ weights as were used to calculate the pose.

7.5.2 Fusing LAPS with Odometry using Spline Fusion

Now that we have introduced the formulation of Spline Fusion, we can now describe the nonlinear least squares minimisation that we perform in our work to estimate the control point pose parameters $G_{W,k}$ using LAPS localisations and odometry. For each LAPS observation classified as a valid solution, we form the $\mathfrak{se}_3$ error term: $\mathbf{e}_{\text{LAPS}}$ given in Equation (7.18),

$$\mathbf{e}_{\text{LAPS}} = \left(\log \left(\hat{G}_{R,W}^{-1} \cdot G_{W,S}(t_{\text{LAPS}}) \right) \right)^\vee \quad (7.18)$$

which is the $\mathfrak{se}_3$ form of the transformation between the spline and the LAPS localisation at the observation timestamp. Equivalently for the odometry we build $\mathbf{e}_{\text{odom}}$ given in Equation (7.19),

$$\begin{aligned} \mathbf{e}_{\text{odom}} &= \left(\log \left(\hat{G}_{R_a,R_b}^{-1} \cdot (G_{W,S}^{-1}(t_a) \cdot G_{W,S}(t_b)) \right) \right)^\vee \\ &= \left(\log \left(\hat{G}_{R_a,R_b}^{-1} \cdot G_{S_a,S_b} \right) \right)^\vee \end{aligned} \quad (7.19)$$

although in this case the error is the $\mathfrak{se}_3$ representation of the transform between

the destination frames of the relative motion transforms from t_a to t_b (by definition: $t_b > t_a$) if their source frames were coincident. Note that to emphasise the source of the terms in Equations (7.18) and (7.19), we keep the subscript S to denote the spline frame. However it should be understood that this frame is assumed to be equivalent to the robot frame R used in the LAPS and odometry observations.

Using these error terms our resulting nonlinear least squares objective is given by Equation (7.20),

$$E = \sum_i e_{\text{LAPS}}^T \cdot \Sigma_{\text{LAPS}}^{-1} \cdot e_{\text{LAPS}} + \sum_j \left(e_{\text{odom}}^T \cdot \Sigma_{\text{odom}}^{-1} \cdot e_{\text{odom}} \right) + \left(e_{\text{Level Prior}}^T \cdot \Sigma_{\text{Level Prior}}^{-1} \cdot e_{\text{Level Prior}} \right) \quad (7.20)$$

where Σ denotes the $\mathfrak{se}_3$ covariance of the relevant observation. The $e_{\text{Level Prior}}$ error term is simply a weak prior that the destination timestamp (t_b) of every odometry motion observation, the vehicle should be, with a large covariance, level: roll & pitch ≈ 0 and moving along the ground plane: Z & $\frac{\partial Z}{\partial t}$ both ≈ 0 in the external frame in which the spline is defined. As in the case of wheel odometry, the relative pose updates are really only $\mathbb{SE}(2)$, we have found that this term helps constrain the spline to feasible vehicle motions. The assumption that $Z \approx 0$ is valid because we always maintain the spline in the frame of the closest node in the scene prior against which we are localising, segments are comparatively short and defined in a purely relative frame and so constant inclines are not apparent.

Choosing sensible values for the observation covariances is unfortunately something of a dark art, particularly as their relative magnitude is what governs the preference of the optimiser when trading errors against each other. Furthermore, a standard practice for Visual Odometry systems of using the inverse of the Gauss-Newton approximate Hessian as the covariance: $\Sigma = (J^T \Omega J)^{-1}$, is a linear approx-

imation at the termination point, and even then, is only accurate if Ω is correctly chosen to rescale the errors on the residuals to be standard normals. In practice, this typically results in the covariances estimated this way being very overconfident.

For the LAPS localisations, we have experimented with using the inverse of the BFGS approximation of the Hessian at convergence as the covariance, but we found that this also resulted in overconfident estimates. There is also a larger question about whether this is theoretically sound, as the BFGS approximation is iteratively updated throughout the optimisation, it is not clear that its value at termination should be expected to be a good estimate of the covariance at termination. In the results presented in Chapter 8, we have used a fixed covariance for all LAPS localisations which is $4\times$ the magnitude of the covariance of the VO in translation and $2\times$ the magnitude in orientation.

Our implementation of Spline Fusion uses the Ceres Solver [1] to solve for the control point poses $G_{W,k}$ which minimise Equation (7.20) using Levenberg-Marquardt and also the Sophus library [104] to allow the use of Ceres’ automatic differentiation [79] to construct the Jacobians.

We maintain a sliding window containing all observations in the last T seconds, with $T = 15$ seconds in the results presented in Chapter 8, and iteratively add new control points to the front of the spline as required to cover the front of the window. Except at the start where we do not have at least T seconds worth of data, we enforce continuity of the optimised spline within the window with the preceding (now fixed) sections that occur before the start of the window by keeping the poses of the control points with non-zero weights at the window start time constant. All control point poses are initialised with the Identity. As mentioned earlier, as we always localise against the closest node frame in the scene prior we are using, we repeatedly update our reference frame W after traversing approximately one edge length ($\approx 30\text{m}$ in this work). When this occurs we transform the spline and all observations into the

new frame, as noted in [88], transforming a spline simply corresponds to applying the transform to all of the control points.

To summarise the configuration used in the results presented in Chapter 8, we used a window size of $T = 15$ seconds and a control point period $\Delta t = 2$ seconds. The VO system on the Wildcat outputs relative poses at 20 Hz, which are combined with LAPS poses at ≈ 4.9 Hz. Using 10 parallel threads to evaluate the residuals and Jacobians, solving iteratively for the control point poses takes ≈ 50 milliseconds on a 2.5GHz Intel Xeon E5-2670 v2. The spline is resolved whenever new data is available and to maintain a timely output pose, the published pose is the most recent spline pose plus the integration of any odometry which has arrived since that time. Thus the complete system outputs pose at the same rate as the odometry: 20Hz.

7.6 Conclusions

In this chapter we described in detail the real-time capable GPU-based implementation of the approach described in this work and used to generate the results presented in Chapter 8. We started by reviewing the benefits of GPUs for our problem and then described in detail how the construction of the joint appearance distributions is implemented using OpenCL. Including the use of one point per pixel subsampling. Then we described how BFGS optimisation can be applied to our problem and demonstrated the convergence properties of the resulting system on data from central Oxford. Motivated by failure cases from the convergence experiments, we then presented a classifier for identifying failure cases based on the value and shape of the NID objective about the candidate pose obtained from the optimiser and demonstrated its effectiveness. Finally, we reviewed the continuous time Spline Fusion algorithm from [53] and described how it is used in our implementation to fuse LAPS localisations with an odometry source.

Chapter 8

Localisation using LAPS

In this chapter, we present the core results which demonstrate the localisation performance of our approach on five datasets from central Oxford collected under different illumination conditions, times of day and year, using the implementation described in Chapter 7. We first present the datasets used and explain how the baseline ground-truth estimate, against which the error metrics are reported, was generated. Finally, we present our key results, which show that our system outperforms a high-grade commercial INS, and discuss particular points of interest and motivations for future work.

8.1 Oxford Datasets

We present localisation results using 5 datasets collected from two overlapping routes of Oxford collected over a year apart. The first route, Route A shown in Figure 8.1 is 6.7km long and encompasses a wide variety of scene geometries and appearances from central Oxford. This includes a collection of main roads in Oxford with various types of residential areas, several Oxford Colleges of mixed ages and aesthetics and commercial / shopping areas. The second route, Route B shown in Figure 8.2 is

broadly the same as Route A but is slightly longer at 7.5km and includes two laps of Broad Street (bottom right in Figure 8.2), a busy area in the centre of Oxford with a car park and a seemingly limitless supply of tourists and buses. Additionally, Route B goes around the front of the Ashmolean Museum (centre of Figure 8.2) instead of around the back as in Route A.

All datasets were captured with the MRG Wildcat shown in Figure 6.2, driving at speeds of 15-20mph. Datasets A1 & A2 were captured 40 minutes apart in the early morning in July 2013, when the weather was overcast. Datasets B1 & B2 were captured approximately 1.5 hours apart during late afternoon in August 2014 when the weather was only moderately cloudy (not overcast), whilst B3 was captured a week earlier on a bright sunny day shortly after 1pm. Thus, the appearance of the scene between A1 & A2 changes only slightly, as evident in Figures 8.1c and 8.1d and traffic is minimal due to the time of day. The appearance changes between B1 & B2 are more significant due to changing cloud patterns and traffic is significantly greater than for A1 & A2. Dataset B3 has very harsh lighting and strong shadows across large sections of the route which were problematic for the autoexposure controller often resulting in washed out images as shown in Figure 8.2e. Note that dataset A3 is not used in this chapter, it was used solely as a training dataset for the solution classifier developed in Section 7.4.

The performance of the Wildcat’s INS, a commercial unit costing approximately £50k with a dual-antenna GPS, is reasonable for A1 & A2, with translation errors typically $\leq 1 - 2\text{m}$ between the routes. However, the performance for B1 & B2 is markedly worse for the first part of the route, with translation errors $> 500\text{m}$ when GPS satellite coverage was lost. The INS completely failed in B3, despite being initialised before data collection commenced. This performance is broadly typical of our experiences of INS performance in central Oxford, where the narrow urban canyons and comparatively high latitude result in patchy GPS coverage.

We argue that the often poor performance of the INS makes it an unsuitable measure of ground-truth against which errors can meaningfully be reported. Instead we perform a global optimisation of the purely relative poses obtained from visual odometry for each route with metric loop closures obtained from a visual odometry step between topological loop closures identified by FABMAP [15] described in detail in Section 8.2.1.

We present results for eight experiments using the five datasets: A1, A2 & B1 - B3, in each of which different datasets are used as the prior, from which the Scene Prior was built and localisation is performed, and the traversal which is the live stream of data to be localised. The directed graph shown in Figure 8.3 represents the datasets as nodes and the experiments presented as red arrows, with the arrow pointing *from* the traversal dataset *to* the prior dataset used in the experiment. Note that although all datasets contain data from the full suite of sensors and thus can be used to construct a Scene Prior, only the Ladybug2 imagery and VO stream are used when the dataset is used as a traversal.

Where an arrow begins and ends at the same node, it means the same dataset was used as both the prior and the traversal, a ‘self-self’ experiment. As noted in Section 7.3.1 where the convergence experiments were presented using a subset of Oxford A1, these do not result in zero appearance change. This is because only a single camera is used to colour a point in the prior, but it is typically used to localise in multiple images. As all experiments are conducted outdoors, the illumination is dynamic and the autoexposure controller is constantly adjusting the camera settings.

All experiments were performed on a desktop machine with a 10-core 2.5GHz Intel Xeon E5-2670 v2 CPU and two NVIDIA TITAN GPUs. The Ladybug2 images, of which only the forward facing three cameras were processed at a resolution of 384×512 , are captured at 10 Hz. The VO runs at 20 Hz, the framerate of the Bumblebee2 stereo camera. The system runs in real-time, with the LAPS localiser



(c) Oxford A1 - Captured on 2013/07/21 at 06:04 am



(d) Oxford A2 - Captured on 2013/07/21 at 06:46 am

(e) Oxford A3 (*used for classifier training only*) - Captured on 2013/08/12 at 18:13 pm

Figure 8.1: Oxford Dataset Route A, used to collect datasets A1, A2 & A3. The route is 6.7km long and encompasses a wide variety of scene geometries and appearances from central Oxford. The top row shows the route overlaid on satellite imagery and the INS traces for the datasets, with kilometre markers shown in black relative to the start of A1. The bottom rows show sample images for the same places across the datasets. Note A3 was *only* used for training the solution classifier from Section 7.4, it is not used in the results presented in this chapter. The identification of the matching images in A2 for those in A1 was performed using the optimised poses discussed in Section 8.2.1, for A3 they were manually aligned for illustration.



(c) Oxford B1 - Captured on 2014/08/16 at 16:30



(d) Oxford B2 - Captured on 2014/08/16 at 18:05



(e) Oxford B3 - Captured on 2014/08/09 at 13:02

Figure 8.2: Oxford Dataset Route B, used to collect datasets: B1, B2 & B3. The route is 7.5km long, covering broadly the same route as Route A, but including two loops of Broad Street (bottom right) and going around the front of the Ashmolean instead of around the back (centre). The top row shows the route overlaid on satellite imagery and the INS traces for datasets and the route from 8.2b, shown as a dashed orange line all relative to the start of A1, with kilometre markers shown in black. Note the poor GPS coverage evident in the INS traces, in B3 the INS failed completely. The bottom rows show sample images for the same places across the datasets.

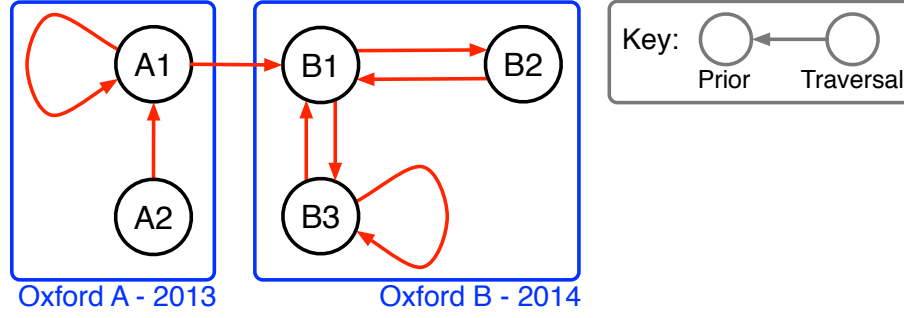


Figure 8.3: Directed graph of prior-traversal experiments presented. The datasets shown in Figures 8.1 and 8.2 are shown as nodes. The eight experiments presented are represented as red arrows, where the arrow points *from* the traversal dataset *to* the prior dataset used in the experiment. The ‘self-self’ experiments, where the same dataset is used as both the prior and the traversal are shown as loopback edges which start and end at the same node.

producing poses at an average rate of 4.9 Hz, thus processing roughly every other Ladybug2 frame. The SplineFusion solver which fuses the LAPS localisations with VO runs at an average of 13 Hz, but outputs poses at 20 Hz by integrating VO between SplineFusion solutions.

8.2 Evaluation Metrics

The core metric of interest for our results is the pose error between the LAPS estimate and ground truth: $G_{\text{Error}}^{\text{LAPS}}$ defined in Equation (8.1),

$$\begin{aligned} G_{\text{Error}}^{\text{LAPS}} &= (G_{W,R}^{\text{Ground Truth}})^{-1} \cdot G_{W,R}^{\text{LAPS}} \\ &= G_{R_{\text{Ground Truth}}, R_{\text{LAPS}}} \end{aligned} \quad (8.1)$$

and the equivalent metric for the Wildat’s INS for comparison: $G_{\text{Error}}^{\text{INS}}$. First we shall discuss how we obtained our estimate of the ground truth alignment of the different datasets, then we will explain the form of the figures presented for each experiment and how they should be understood.

8.2.1 Optimised Ground Truth

As noted previously and evident from Figure 8.2, the INS is not a suitable measure of ground-truth against which to compute $G_{\text{Error}}^{\text{LAPS}}$. Instead, we aligned all of the datasets using a global optimisation of the integrated purely relative poses obtained from VO for each dataset with metric loop closures obtained from a visual odometry step between topological loop closures identified by FABMAP [15], all performed with the stereo Bumblebee2 camera only. As there were some areas on the routes where FABMAP did not return any loop closure candidates, we manually added an additional 48 topological loop closures for which the VO was again used to estimate a metric loop closure constraint. Note that the suppression by FABMAP of loop closure candidates in these areas would generally be desirable behaviour to suppress errors due to perceptual aliasing, however for our specific use case of aligning multiple runs of the same route, it is not.

There were between 28k - 32k relative VO poses per dataset, with $\approx 11\text{k}$ FABMAP + VO metric loop closures across all of the datasets. The optimisation took 68 seconds on a 2.7 GHz Intel i7 CPU with a single thread using the Ceres Solver [1]. We emphasise that the purpose of this optimisation is to align the integrated relative VO pose chains from different datasets to each other, not to produce a globally metrically accurate estimate of the routes. Indeed the result does not meet this criterion, as it does not allow us to compute an accurate metric pose between distant points either on the same, or on different routes. However, it does allow us to accurately compute the metric pose between points on different datasets from the same local area, which is all that we require in order to compute $G_{\text{Error}}^{\text{LAPS}}$. This is in part because we are using the forward facing Bumblebee2 stereo camera to estimate metric loop closures and thus we do not obtain any loop closures when driving along the same road in the opposite direction, as the camera has no common field of view between the two.

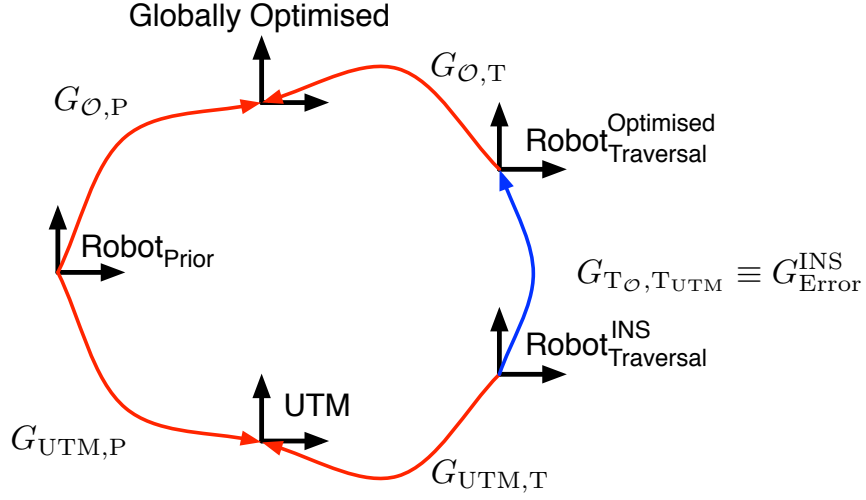


Figure 8.4: Computation of INS error: $G_{\text{Error}}^{\text{INS}}$ using the globally optimised alignment. The INS pose error relative to the globally optimised alignment is defined as the transform between the pose of the robot in the traversal as reported by the INS and the globally optimised alignment, shown in blue in the figure.

We use the local alignments between the datasets obtained from the global optimisation as the ground truth pose for any experiment where the traversal $\neq$ the prior, by which we mean the images being localised were captured at a different time to those used to colour the scene prior. For the ‘self-self’ cases, the images being localised are the same ones that were used to colour the scene prior. Therefore we already have by definition perfect ground truth and do not need to use the globally optimised alignment.

In addition to computing $G_{\text{Error}}^{\text{LAPS}}$, we can also use the optimised alignment to compute the equivalent error for the INS: $G_{\text{Error}}^{\text{INS}}$ using the approach shown in Figure 8.4. Thus for every experiment where the traversal $\neq$ prior, we present the same figures as for LAPS for the INS for comparison. For the ‘self-self’ cases the INS performance is not shown, as $G_{\text{Error}}^{\text{INS}} = \mathbf{0}$ by definition for these cases.

Our assertion is that the globally optimised alignment represents the best estimate of ground-truth readily achievable for the data, and is significantly better than that which would be achieved if the INS were used instead. Furthermore, it

allows us to meaningfully compare the performance of LAPS vs the INS without having to assume that the exact same path was driven in all datasets for a given route. As although the same route was driven for each case, placement within the lane, the path taken through wide junctions and deviations due to traffic and temporary obstructions mean the path for each dataset will differ. Finally, we note that the qualitative performance of the alignment can be seen from the sample images for each route shown in Figures 8.1c, 8.1d and 8.2c to 8.2e, the selection of which was obtained from the closest image in each dataset for each place according to the globally optimised alignment.

8.2.2 LAPS Initialisation and Restarts

At the start of each dataset, the system is initialised with the ground-truth pose obtained from the globally optimised alignment. If during the dataset LAPS fails to output a pose classified as valid by the classifier for 5 seconds, it declares itself lost and requests a restart. At each restart, the system is reset and initialised with the ground-truth pose obtained from the globally optimised alignment.

In practice, when the system is running live on a robot, the ground-truth pose for all points will not be known a priori. In that case, as noted in Section 7.4.3, an external source such as an INS (if available) or a vision-based topological localisation system such as [15, 56, 69, 87, 75] would be used to initialise the system. As these methods are purely topological, this involves matching the live imagery to that from the survey vehicle, and then reporting the metric pose in the prior of the matching survey image as the initialisation pose.

In the results presented here, by resetting the system with the ground-truth pose whenever it independently determines it is lost, we are assuming that such an initialisation is always instantaneously available. In practice, if a system such as FABMAP [15] were used to restart LAPS, this would not always be the case;

restarts could not necessarily occur as soon as LAPS determines that it is lost, as FABMAP itself may also be lost. However, we have taken this approach as it is the performance of LAPS which we wish to evaluate, independently of, and thus unbiased by, the performance of any other system on each dataset.

We emphasise that it is LAPS that determines when it is lost, and to request a restart and that it does so without any knowledge of the pose error relative to ground-truth. It relies instead solely upon the result of the classifier to identify this situation, thus it is possible for LAPS to declare itself lost, even if G_{Error} is in fact small. During the 5 seconds until LAPS declares itself lost, but has not output a new pose classified as valid, the system is running purely on integrated odometry.

8.2.3 Explanation of Figures

For each experiment shown in Figure 8.3 we obtain three sets of poses: the LAPS localisations classified as valid by the classifier from in Section 7.4; the output from SplineFusion which fuses the positive LAPS localisations with odometry as detailed in Section 7.5 and finally the poses from the INS. For the fused pose and the INS, we compute the errors relative to the globally optimised alignment at 10 Hz, the framerate of the Ladybug2 and thus the maximum rate at which LAPS localisations can be computed, even though both sources output at higher rates (20 Hz and 100 Hz respectively) to allow unbiased comparisons. For the LAPS localisations, the error was computed for every pose, the mean rate of which was ≈ 5 Hz. We present these error metrics in two different ways for each experiment:

- 2D Histograms of distance travelled vs crosstrack & heading error for the fused LAPS + Odometry and INS poses.
- Histograms of Euler components of G_{Error} for all pose sources.

The latter has already been seen earlier in this work in Section 7.3.1 and Figure 7.6 and the form is the same as in that case, with the number of samples in each bin in the histogram expressed as a percentage of the total number of samples and the translation and orientation components shown separately. These plots capture the gross performance of the source and any persistent biases, for a perfect system, there would be a Kronecker delta at zero error for each Euler component.

An example of the 2D histograms of distance travelled vs crosstrack & heading error is shown in Figure 8.5 for the crosstrack error case. For every pose in the dataset with an associated G_{Error} , the cumulative distance travelled as reported by integrating the odometry from the start is computed. The crosstrack and heading error in the robot frame are obtained from G_{Error} , where the crosstrack error is the distance from the associated ground truth robot pose to the estimated pose measured perpendicular to the direction of motion of the robot. The poses are then grouped into consecutive, non-overlapping 30 metre sections in distance travelled. For each 30 m section, separate histograms with 63 evenly spaced bins are computed for crosstrack and heading error using all of the poses within the section. The full 2D histogram of each error term across the entire dataset is then constructed by stacking the individual histograms for each section in order for each error term. Thus in the final result, each *column* represents the individual histogram of the error term for that section. As with all histograms in this work, the perimeter bins also contain all samples with errors outside the specified range.

As all of the individual histograms are represented as percentages of the data in that section, each *column* sums to 100 % and the value of each cell is represented using colour, with white being zero. In Figure 8.5 the individual histogram for the highlighted section (column in the main figure) is shown as an inset. The speed of the Wildcat was between 15-20 mph except for infrequent stops at traffic lights, of which there are 20 on Route A and 22 on Route B and on average each dataset

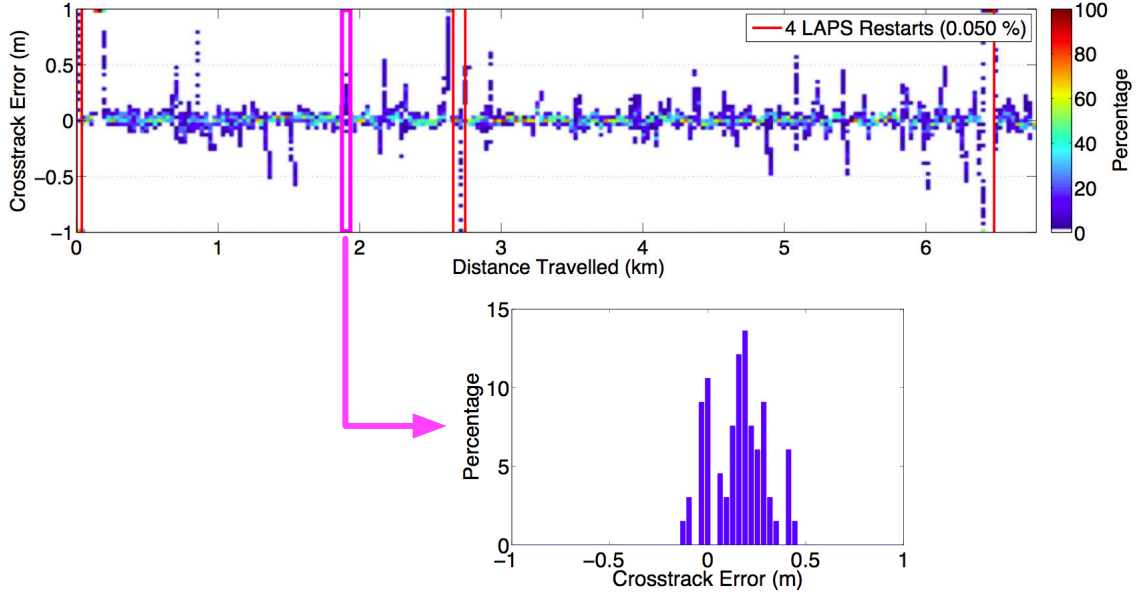


Figure 8.5: Sample 2D Histogram of Distance Travelled vs Crosstrack Error, showing inset histogram for a single column. For each experiment, the distance travelled for all poses from the specified source with an associated G_{Error} is computed by integrating the odometry source. Also computed are the crosstrack and heading error relative to the globally optimised ground-truth alignment. The data is then divided into consecutive, non-overlapping 30 metre windows in distance travelled and a separate histogram of the error term (crosstrack or heading respectively, crosstrack in the sample figure) is computed for each using all of the data in that section. A sample histogram for a single section is shown at the bottom of the figure as an inset. Note that the percentages in this histogram are expressed in terms of the number of samples in the section and so it sums to 100 %. The complete 2D histogram for the entire dataset is produced by stacking the histograms for each section together in order, where each *column* represents the histogram for a single section. The column corresponding to the inset histogram is highlighted in pink. The 2D histogram is *not* renormalised, and thus each column sums to 100 %. The colour of each cell denotes its percentage value, with white denoting zero (no samples). As with all histograms in this work, the boundary bins also contain all samples that fall outside the bounds shown. Also shown overlaid on the figure are the points at which LAPS requested a restart, shown as vertical red lines, with the total number of restarts, also expressed as a percentage of the total number of LAPS poses given in the legend.

stops at approximately half of them. As such, the number of samples per column in the 2D distance vs error histograms are approximately constant, and on average there are $\approx 12.5\text{k}$ samples in total across all sections in each figure.

The purpose of the 2D distance vs error histograms is to represent the profile of the errors most important for driving a robot throughout each experiment. For a perfect system, they would be a dark red line across the centre of the figure, indicating that all of the samples for all of the sections had zero error.

Also shown overlaid on each 2D distance vs error histogram are the location of each LAPS restart as discussed in Section 8.2.2, shown as vertical red lines with the total number of restarts, also expressed as a percentage of the number of LAPS localisations, given in the legend as shown in Figure 8.5. The locations of all restarts, and the last valid LAPS pose before a restart, are also shown on the overhead INS trace of each route.

Finally, as noted earlier, the 2D distance vs error histograms are only shown for the INS for the experiments where the traversal $\neq$ prior, as for the ‘self-self’ cases the INS error is zero by definition.

8.3 Results

Before presenting the detailed results for each experiment, we summarise the overall performance of the system across the experiments and highlight particular points of interest. Readers are advised to read the summary results section (Section 8.3.1) first and then move on to Section 8.4 which highlights points of interest from the detailed figures, and to use these sections as indexes into the detailed results themselves.

8.3.1 Summary Results

The core results presented in this work are summarised in Figures 8.6 and 8.7 and support our core claim that the system presented in this work provides performance greater than that of a high-grade commercial INS.

Figure 8.6 shows the aggregate crosstrack and heading errors for up to three sources for each experiment, LAPS localisations classified as positive by the classifier from Section 7.4, the pose estimate from fusing positive LAPS localisations with odometry and finally the INS, which is only shown for cases where the traversal $\neq$ prior. The errors shown in Figure 8.6 are not on their own sufficient to compare the performance of the system in different experiments as it does not capture the number of LAPS restarts, or the percentage of LAPS localisations classified as positive. These results are presented in Figure 8.7.

The two most important points from Figure 8.6 are that the heading error for the fused pose output is broadly comparable or smaller than that of the INS; and that the crosstrack error is significantly smaller across all experiments. As these are the two key measures of performance for a pose source for a ground vehicle, this supports our claim that our system outperforms the INS. The errors for the fused output are larger than those of the positive LAPS localisations because they include integrated odometry for the period when LAPS failed to return a positively classified pose; particularly in the periods leading up to a restart, these poses are often in error.

Broadly, the errors are proportional to the degree of appearance change between the traversal and the prior. The ‘self-self’ experiments which have the smallest appearance change produce the lowest errors with the fewest number of LAPS restarts and the highest percentage of positively classified LAPS poses. This is irrespective of the conditions, as whilst the conditions for A1 were fully overcast, with flat lighting with soft shadows, B3 was captured on a bright sunny day with few clouds resulting

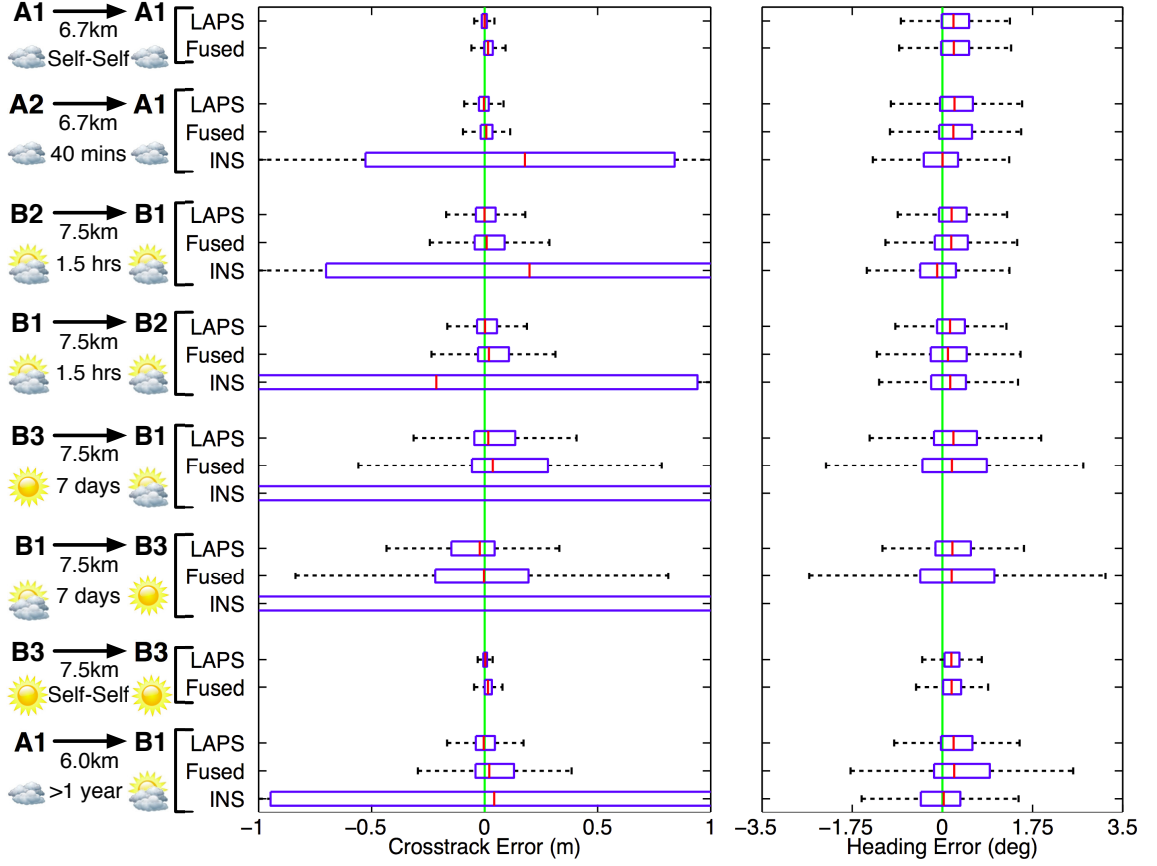
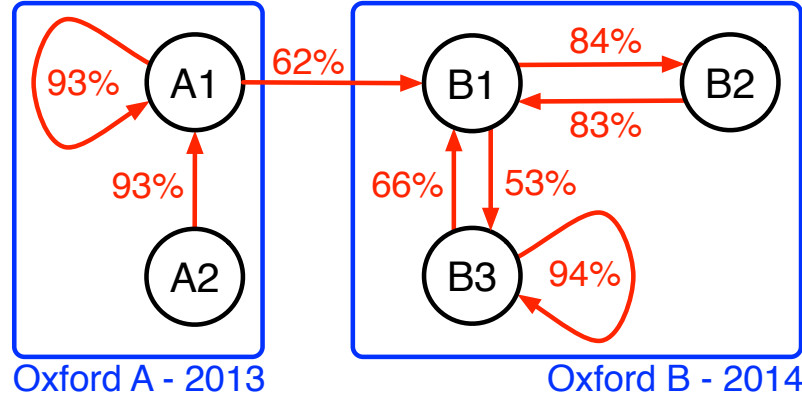
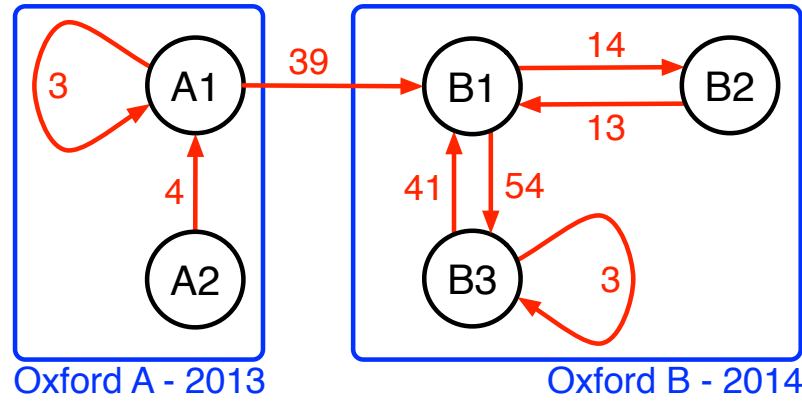


Figure 8.6: Summary cross-track and heading errors for all experiments presented. For each experiment shown on the left, where the arrow points *from* the traversal dataset *to* the prior dataset as per Figure 8.3, box plots of the cross-track (left) and heading (right) errors for up to three pose sources: LAPS, Fused and INS are shown. Respectively these are, the LAPS localisations classified as positive; the output from fusing positive LAPS localisations with odometry using SplineFusion and finally the output from the INS. The central red line for each box is the median, the edges of the box are the 25th and 75th percentiles and the whiskers extend to the most extreme points not considered outliers¹, a vertical green line highlights the origin (zero error). The INS is only shown for cases where the traversal $\neq$ prior, as it is zero by definition for the ‘self-self’ cases. In the experiments involving B3, the INS errors are very large, as the INS failed completely shortly after the start. The key point to note is that the heading error for the fused output is broadly the same, or slightly better than that of the INS, but that the cross-track error is *significantly* better across all experiments. The errors for the fused output are larger than those of the positive LAPS localisations because it includes integrated odometry for the periods when LAPS fails to return a positively classified pose and particularly in the period leading up to a restart, these poses are often in error.



(a) Percentage of LAPS poses classified as positive for each experiment



(b) Number of LAPS restarts for each experiment

Figure 8.7: Percentage of LAPS poses classified as positive by SVM classifier and the number of LAPS restarts for each experiment. Annotated versions of Figure 8.3, where the directed edges representing each experiment are annotated with respectively 8.7a: the percentage of LAPS poses produced which were classified as positive (valid) by the SVM classifier detailed in Section 7.4; and 8.7b: the number of LAPS restarts. Only the positive LAPS poses are fused into the final solution and shown in the figures in this chapter. For example, considering the arrow on the top right we can determine that for the experiment with B1 as the traversal and B2 as the prior ($B1 \rightarrow B2$) the percentage of positively classified LAPS poses was 84% and the number of LAPS restarts was 14.

in harsh lighting, overexposed images and strong shadows. The ‘self-self’ results for B3 are however less generalisable, because the dominant shadows can provide strong cues for alignment on what might otherwise be uniform appearance regions (e.g. roads), but these cues are transient and do not reflect the true appearance profile of the scene.

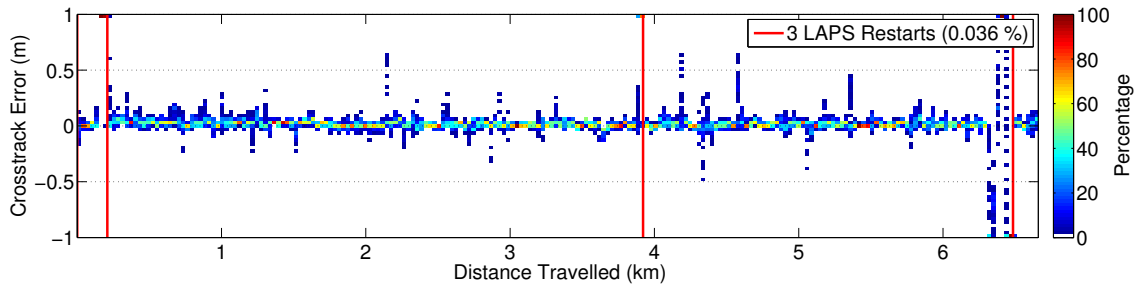
For the cases where the traversal $\neq$ prior, $A2 \rightarrow A1$ which were captured 40 minutes apart in the early morning and so has the smallest appearance change and minimal traffic has performance comparable to that of the A1 ‘self-self’ case. By contrast, the performance is lower, with larger extrema errors for $B2 \rightarrow B1$, which were captured 1.5 hours apart in the afternoon on an only partially cloudy day with typical traffic and so are only broadly similar in appearance.

Due to this similarity and because B1 & B2 do not in general have the harsh lighting and dominant shadows as in B3, which is used as the prior and the traversal has minimal affect on the performance, as evident from comparing $B2 \rightarrow B1$ to $B1 \rightarrow B2$. By contrast, the results for $B3 \rightarrow B1$ show a marked reduction in the number of LAPS restarts and increase in the percentage of positively classified LAPS localisations versus $B1 \rightarrow B3$, even though their errors appear broadly comparable in terms of magnitude from Figure 8.6. This should be expected, because as the illumination conditions were more favourable for B1, the Scene Prior built from it more faithfully captures the appearance of the scene, providing a better segmentation.

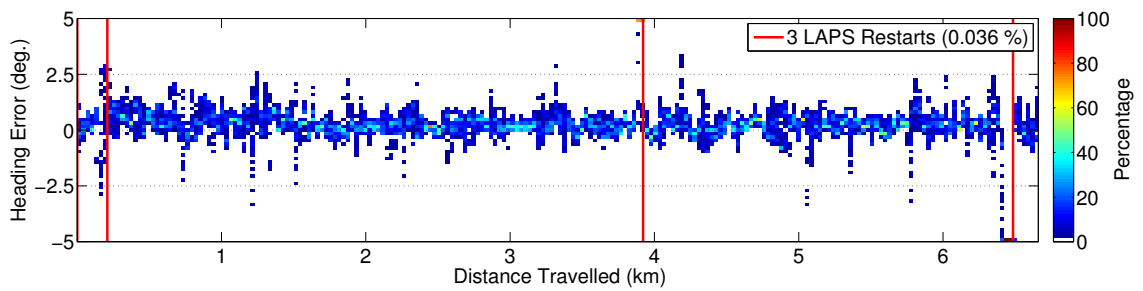
The final experiment: $A1 \rightarrow B1$, demonstrates the performance when localising using a prior that was captured over a year and over 10 hours (6am vs 4:30pm) apart. It was performed this way round, with the traversal dataset $<$ prior dataset in time to allow for the largest contiguous run, due to the differences between Routes A & B. For the same reason, it terminates where the routes diverge before the Ashmolean Museum and is thus $\approx 0.7\text{km}$ shorter than the other experiments using Route A.

The performance is better than for the $B3 \rightarrow B1$ case, which were captured only a week apart, but with harsh illumination changes, which should be expected as the resulting image quality is better in A1. Conversely, the performance is worse than that of the $B2 \Rightarrow B1$ experiments captured on the same day, which is particularly evident in the number of LAPS restarts and the percentage of positively classified LAPS poses. However, the crosstrack error is still significantly better than that of the INS and the heading error is comparable.

8.3.2 Traversal: Oxford A1 $\rightarrow$ Prior: Oxford A1



(a) LAPS Fused with Odometry: Crosstrack Error



(b) LAPS Fused with Odometry: Heading Error

Figure 8.8: 2D Distance travelled vs crosstrack: 8.8a and heading: 8.8b error respectively for LAPS fused with odometry for A1 $\rightarrow$ A1. Please refer to Section 8.2.3 for a detailed explanation of how to interpret these figures.

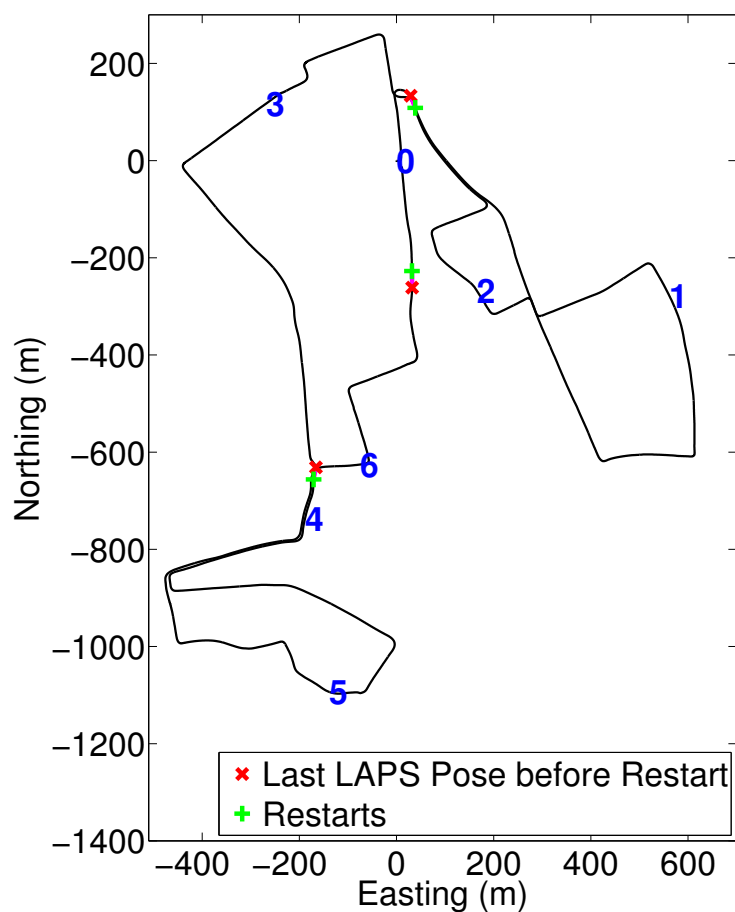


Figure 8.9: Overhead view of Route A showing the locations at which LAPS requested a restart for $A1 \rightarrow A1$. The points at which LAPS requested a restart after failing to produce a positively classified pose for 5 seconds are denoted by a green plus. Also shown as red crosses are the locations along the route of the last positively classified LAPS pose before the subsequent restart, by definition 5 seconds earlier, connected to the restart by a magenta line. The kilometre markers are also shown in blue.

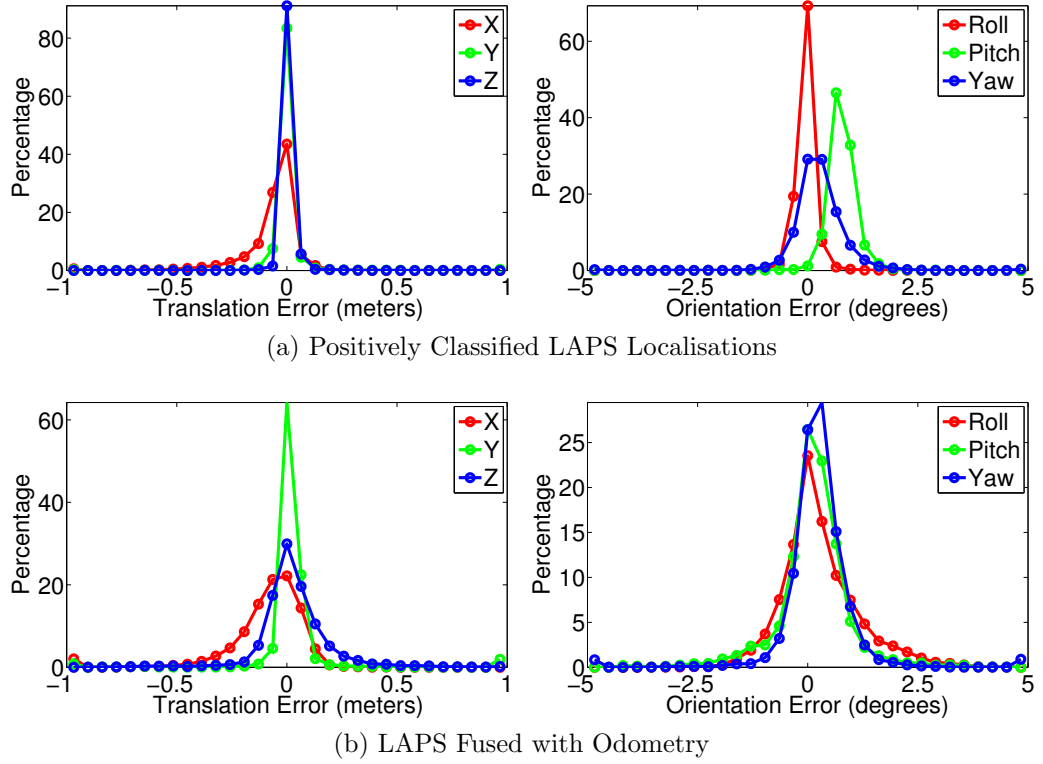
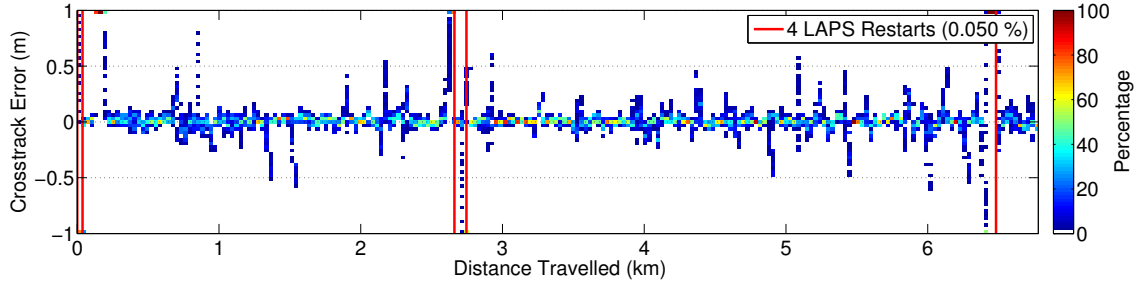
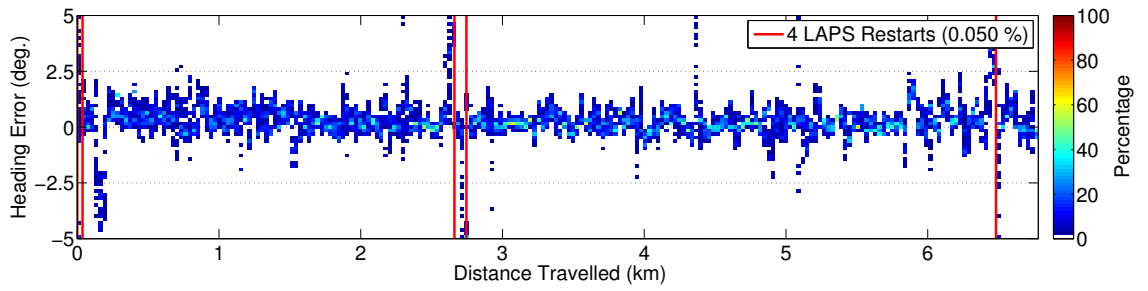


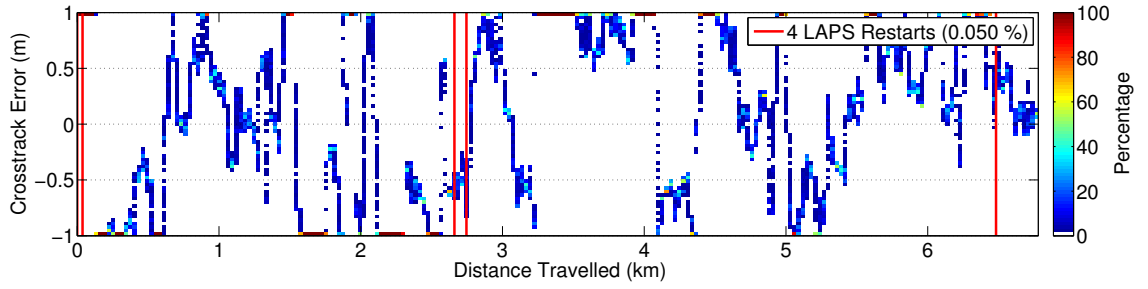
Figure 8.10: Histograms of Euler components of G_{Error} for LAPS and the Fused output for $A1 \rightarrow A1$. The histograms of G_{Error} are expressed in terms of the percentage of samples, for the LAPS localisations classified as positive: 8.10a, and the output after fusing these with odometry using SplineFusion: 8.10b.

8.3.3 Traversal: Oxford A2 $\rightarrow$ Prior: Oxford A1

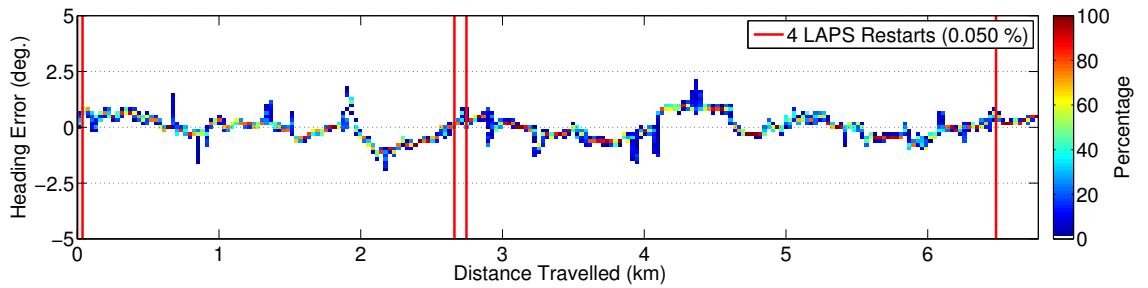
(a) LAPS Fused with Odometry: Crosstrack Error



(b) LAPS Fused with Odometry: Heading Error



(c) INS: Crosstrack Error



(d) INS: Heading Error

Figure 8.11: 2D Distance travelled vs crosstrack: Figure 8.11a and heading: Figure 8.11b error respectively for LAPS fused with odometry and the corresponding figures for the INS: Figures 8.11c and 8.11d for A2 $\rightarrow$ A1. Please refer to Section 8.2.3 for a detailed explanation of how to interpret these figures.

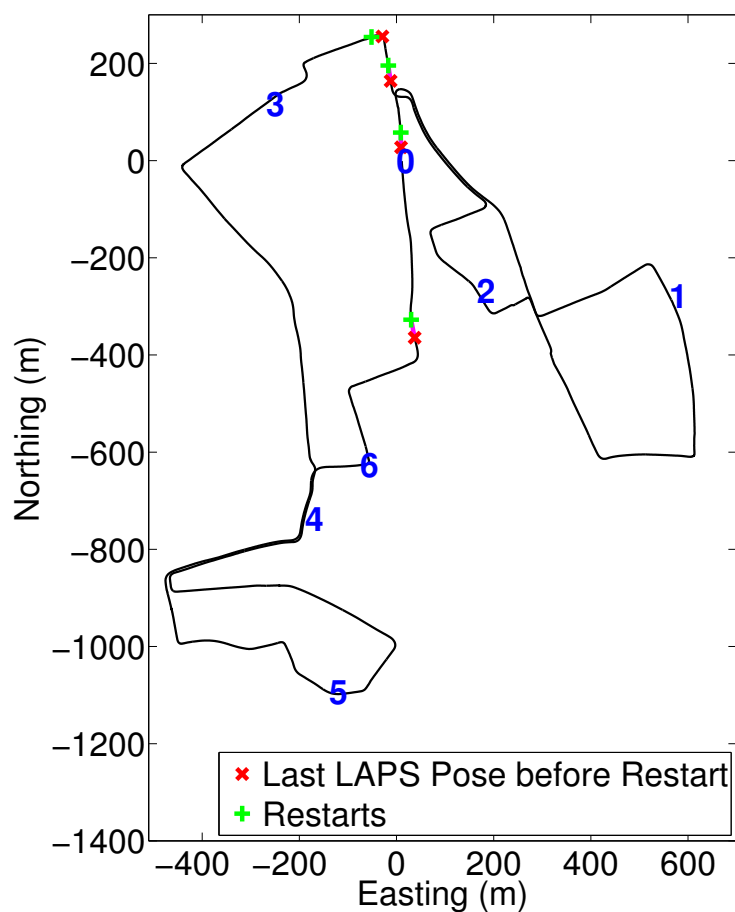
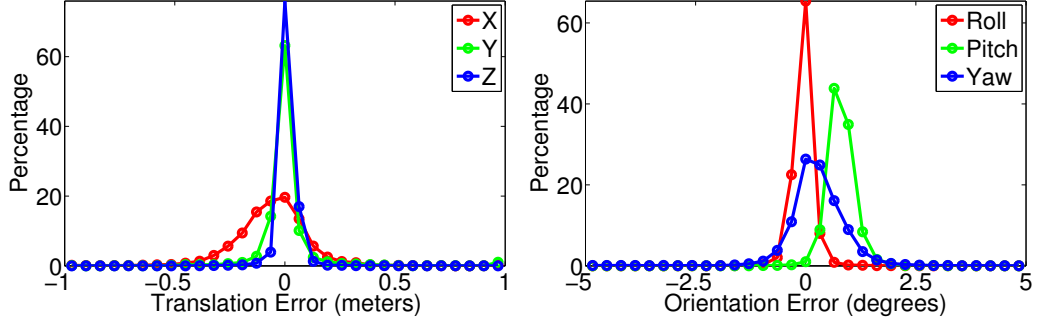
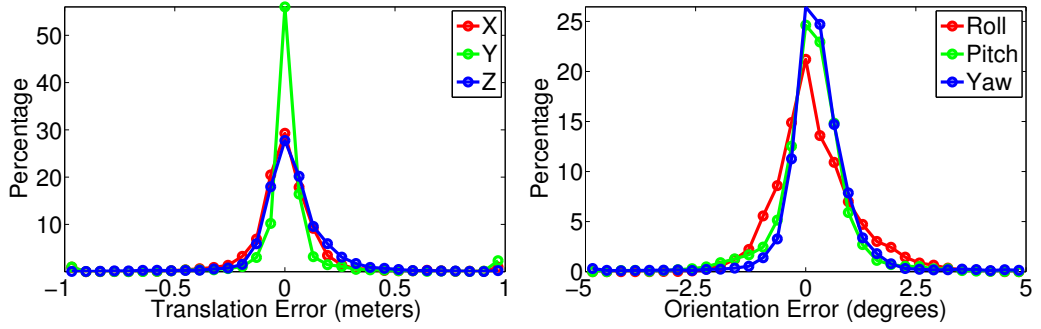


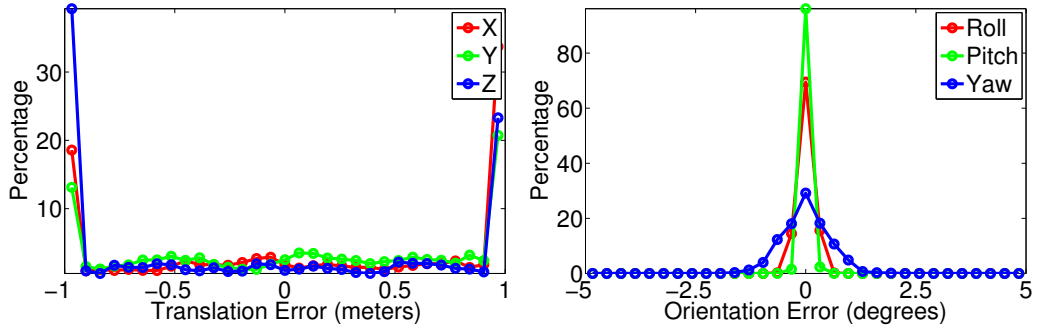
Figure 8.12: Overhead view of Route A showing the locations at which LAPS requested a restart for $A2 \rightarrow A1$. The points at which LAPS requested a restart after failing to produce a positively classified pose for 5 seconds are denoted by a green plus. Also shown as red crosses are the locations along the route of the last positively classified LAPS pose before the subsequent restart, by definition 5 seconds earlier, connected to the restart by a magenta line. The kilometre markers are also shown in blue.



(a) Positively Classified LAPS Localisations

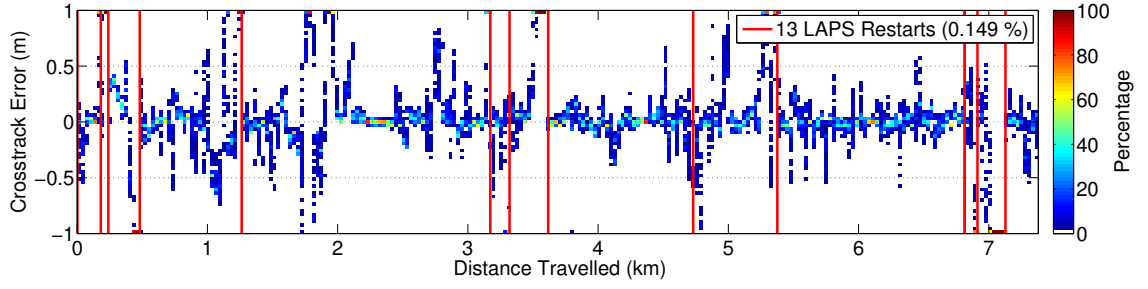


(b) LAPS Fused with Odometry

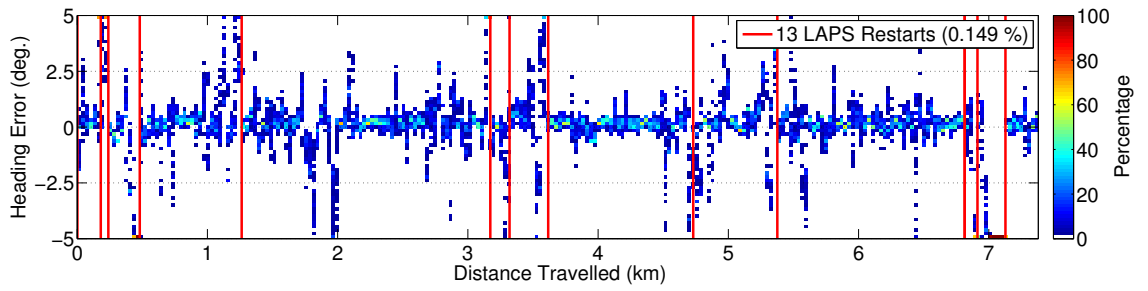


(c) INS

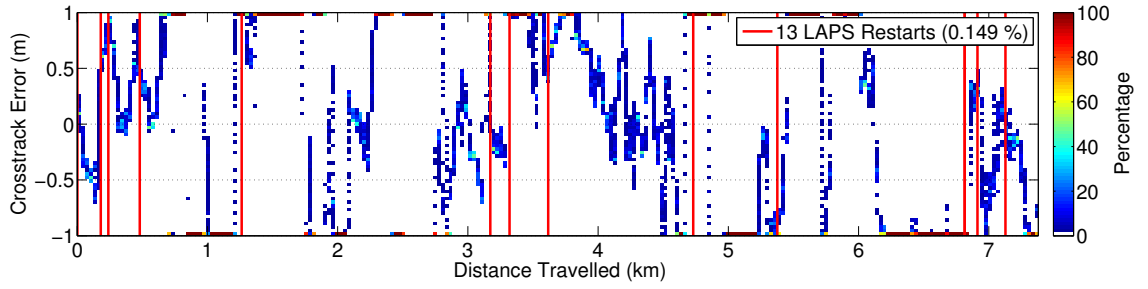
Figure 8.13: Histograms of Euler components of G_{Error} for LAPS, the Fused output and the INS for $A2 \rightarrow A1$. The histograms of G_{Error} are expressed in terms of the percentage of samples, for the LAPS localisations classified as positive: Figure 8.13a, the output after fusing these with odometry using SplineFusion: Figure 8.13b and for the INS: Figure 8.13c.

8.3.4 Traversal: Oxford B2 $\rightarrow$ Prior: Oxford B1

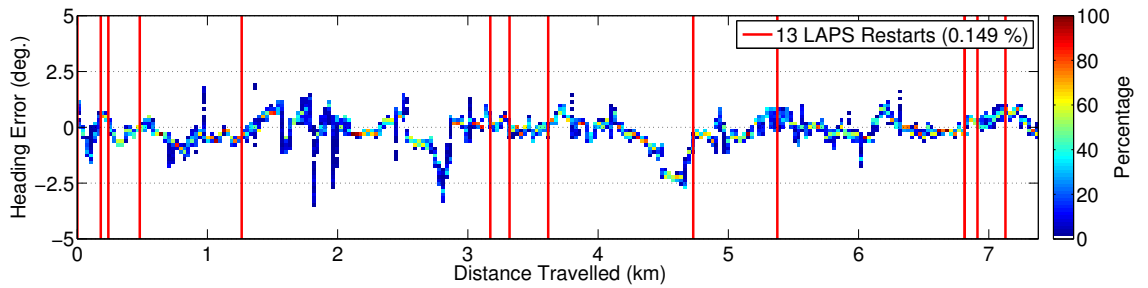
(a) LAPS Fused with Odometry: Crosstrack Error



(b) LAPS Fused with Odometry: Heading Error



(c) INS: Crosstrack Error



(d) INS: Heading Error

Figure 8.14: 2D Distance travelled vs crosstrack: Figure 8.14a and heading: Figure 8.14b error respectively for LAPS fused with odometry and the corresponding figures for the INS: Figures 8.14c and 8.14d for B2 $\rightarrow$ B1. Please refer to Section 8.2.3 for a detailed explanation of how to interpret these figures.

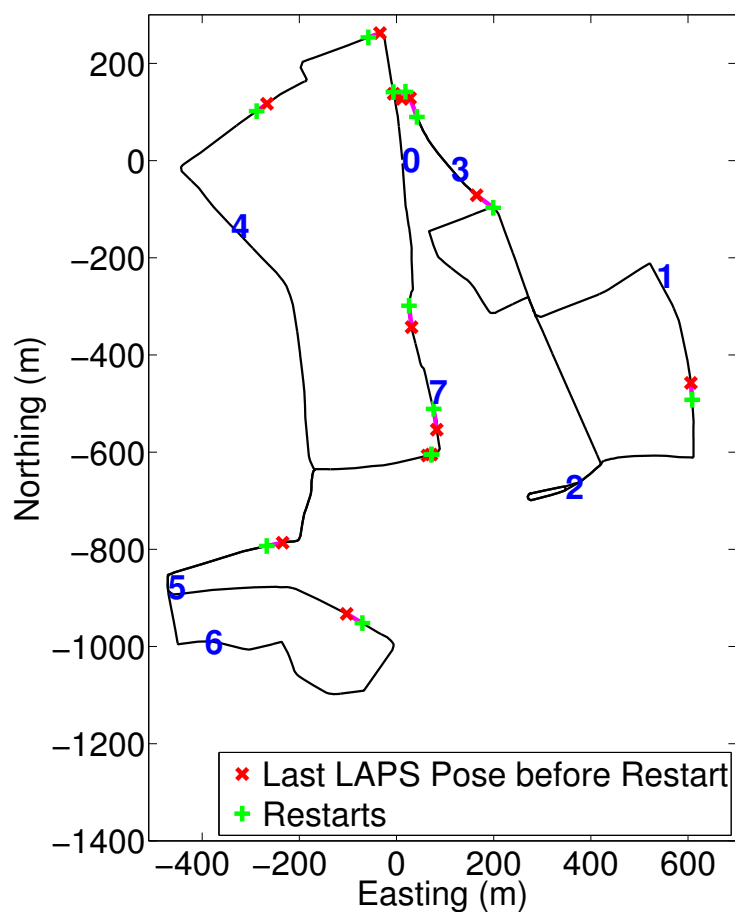
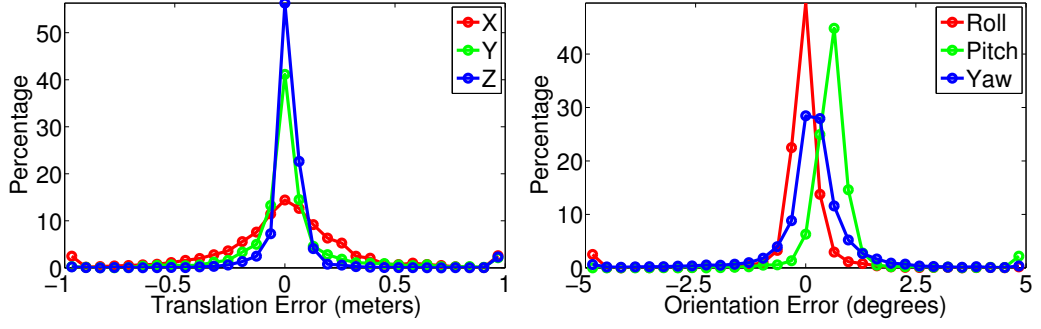
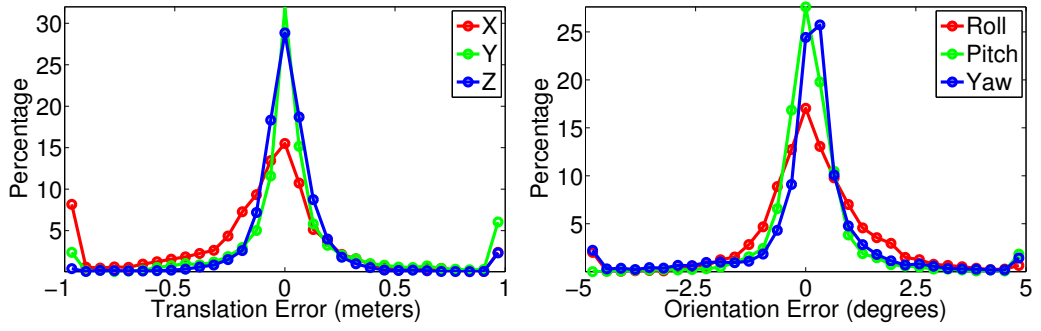


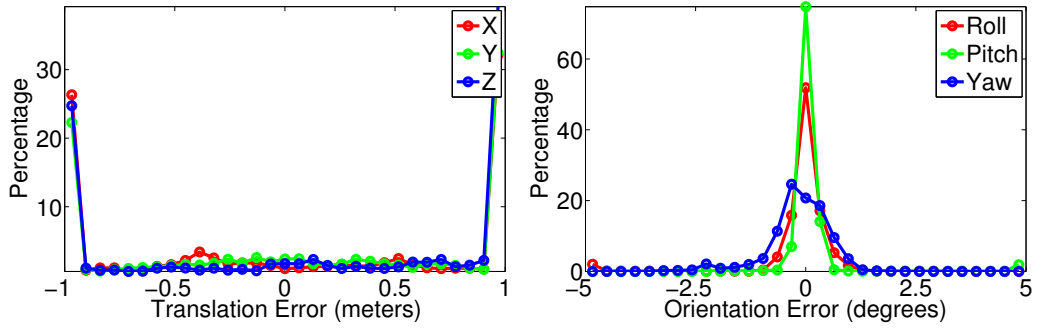
Figure 8.15: Overhead view of Route B showing the locations at which LAPS requested a restart for B2 $\rightarrow$ B1. The points at which LAPS requested a restart after failing to produce a positively classified pose for 5 seconds are denoted by a green plus. Also shown as red crosses are the locations along the route of the last positively classified LAPS pose before the subsequent restart, by definition 5 seconds earlier, connected to the restart by a magenta line. The kilometre markers are also shown in blue.



(a) Positively Classified LAPS Localisations

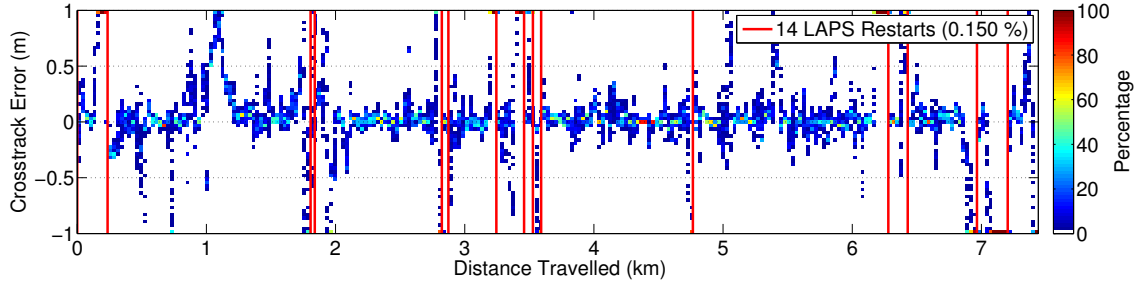


(b) LAPS Fused with Odometry

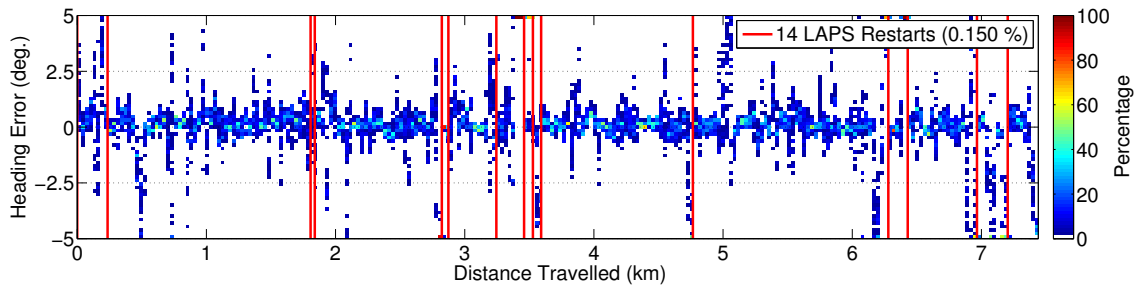


(c) INS

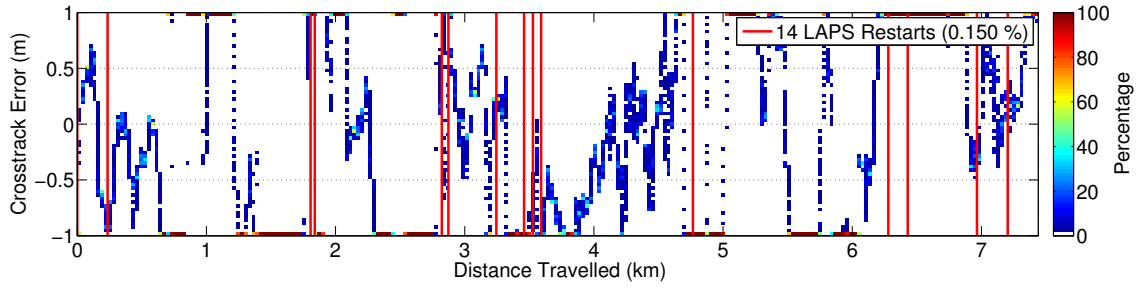
Figure 8.16: Histograms of Euler components of G_{Error} for LAPS, the Fused output and the INS for $B2 \rightarrow B1$. The histograms of G_{Error} are expressed in terms of the percentage of samples, for the LAPS localisations classified as positive: Figure 8.16a, the output after fusing these with odometry using SplineFusion: Figure 8.16b and for the INS: Figure 8.16c.

8.3.5 Traversal: Oxford B1 $\rightarrow$ Prior: Oxford B2

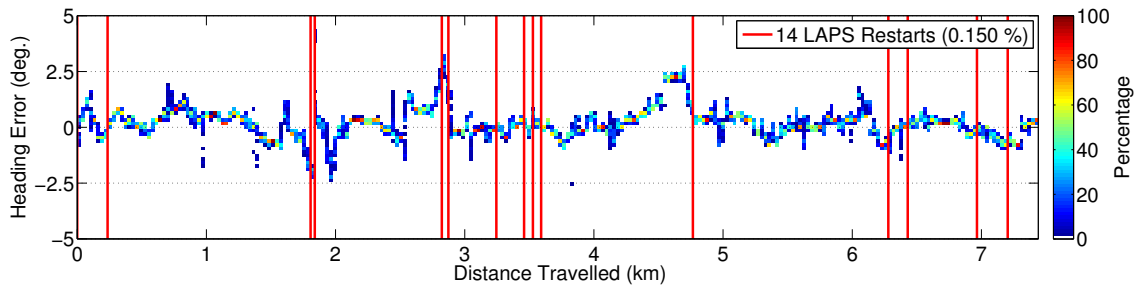
(a) LAPS Fused with Odometry: Crosstrack Error



(b) LAPS Fused with Odometry: Heading Error



(c) INS: Crosstrack Error



(d) INS: Heading Error

Figure 8.17: 2D Distance travelled vs crosstrack: Figure 8.17a and heading: Figure 8.17b error respectively for LAPS fused with odometry and the corresponding figures for the INS: Figures 8.17c and 8.17d for B1 $\rightarrow$ B2. Please refer to Section 8.2.3 for a detailed explanation of how to interpret these figures.

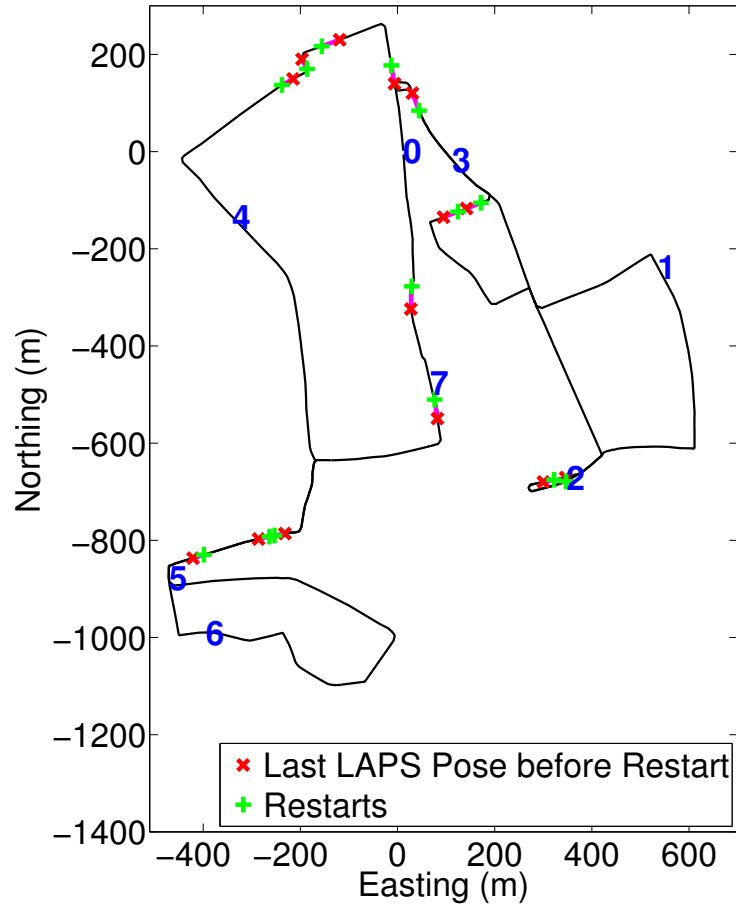
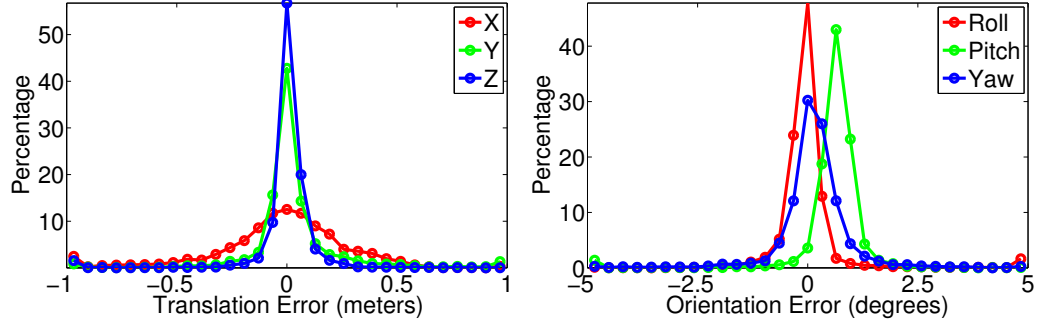
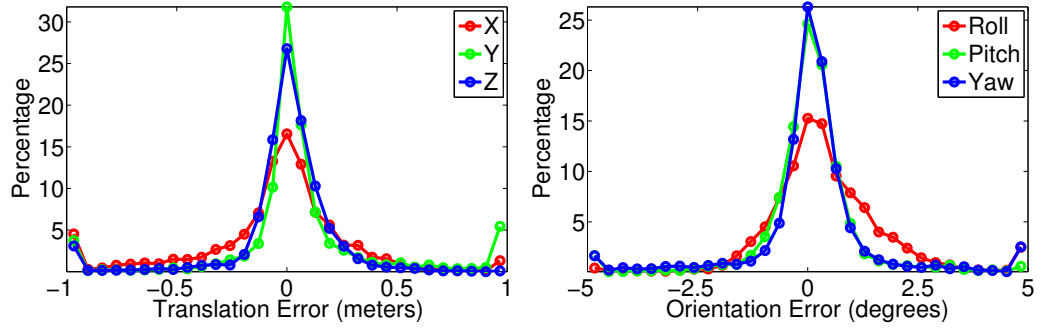


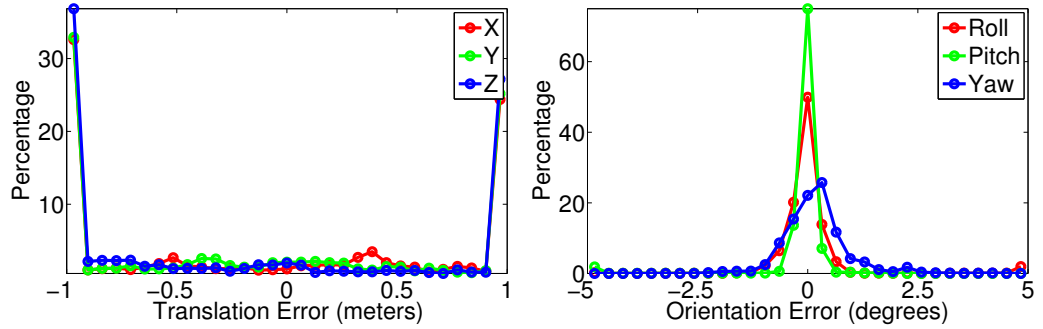
Figure 8.18: Overhead view of Route B showing the locations at which LAPS requested a restart for $B1 \rightarrow B2$. The points at which LAPS requested a restart after failing to produce a positively classified pose for 5 seconds are denoted by a green plus. Also shown as red crosses are the locations along the route of the last positively classified LAPS pose before the subsequent restart, by definition 5 seconds earlier, connected to the restart by a magenta line. The kilometre markers are also shown in blue.



(a) Positively Classified LAPS Localisations

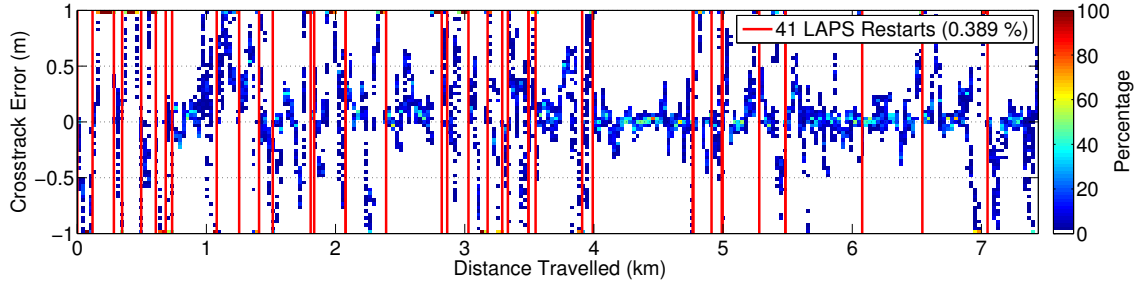


(b) LAPS Fused with Odometry

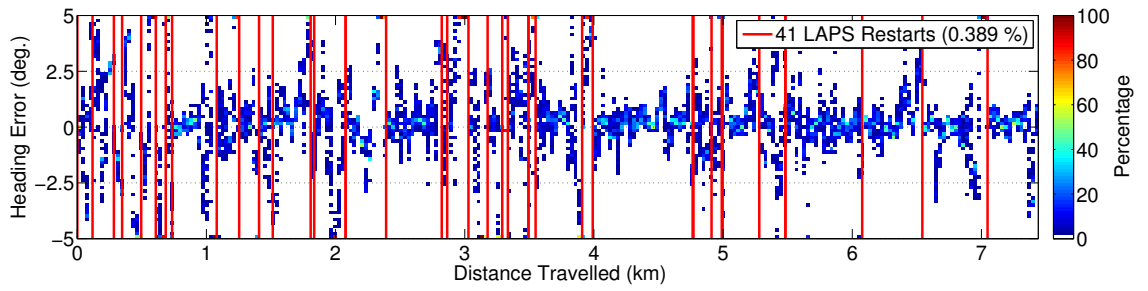


(c) INS

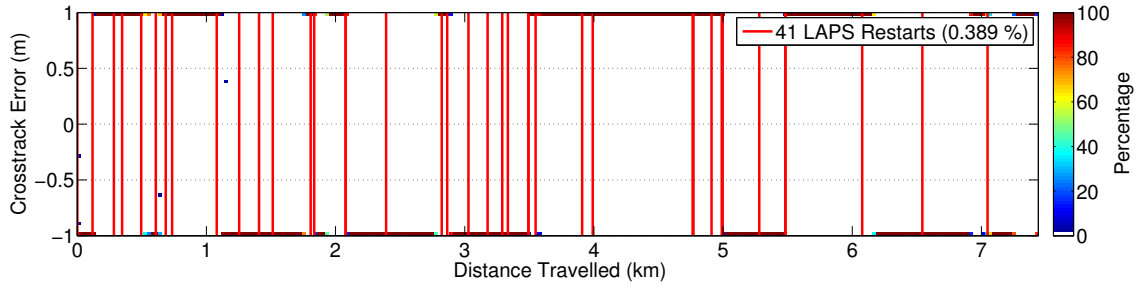
Figure 8.19: Histograms of Euler components of G_{Error} for LAPS, the Fused output and the INS for $B1 \rightarrow B2$. The histograms of G_{Error} are expressed in terms of the percentage of samples, for the LAPS localisations classified as positive: Figure 8.19a, the output after fusing these with odometry using SplineFusion: Figure 8.19b and for the INS: Figure 8.19c.

8.3.6 Traversal: Oxford B3 $\rightarrow$ Prior: Oxford B1

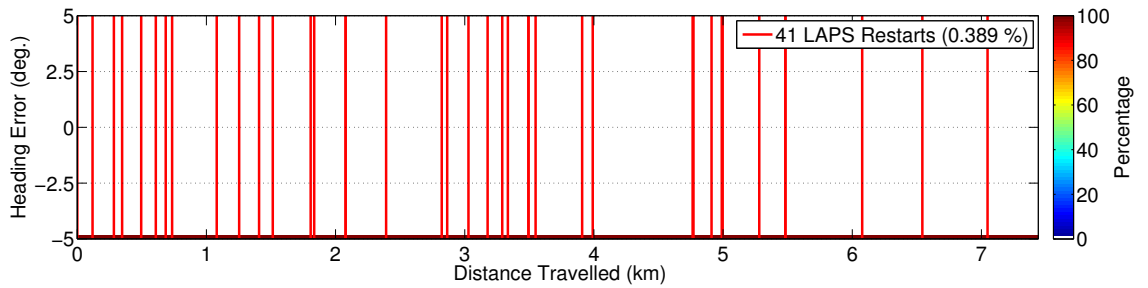
(a) LAPS Fused with Odometry: Crosstrack Error



(b) LAPS Fused with Odometry: Heading Error



(c) INS: Crosstrack Error



(d) INS: Heading Error

Figure 8.20: 2D Distance travelled vs crosstrack: Figure 8.20a and heading: Figure 8.20b error respectively for LAPS fused with odometry and the corresponding figures for the INS: Figures 8.20c and 8.20d for B3 $\rightarrow$ B1. Please refer to Section 8.2.3 for a detailed explanation of how to interpret these figures. Note that for B3, the INS failed shortly after the start.

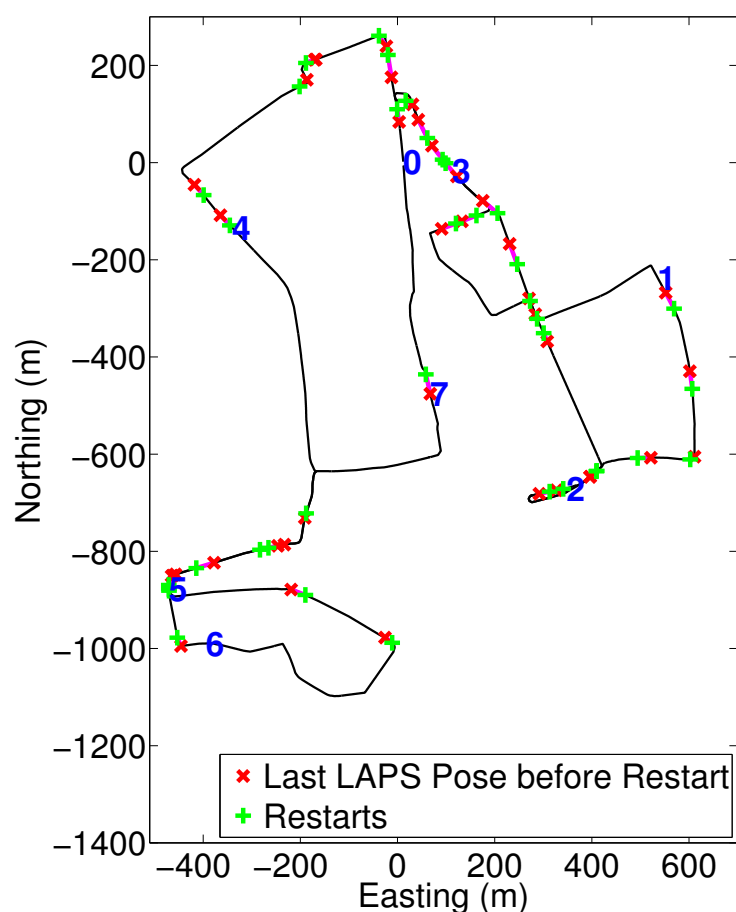
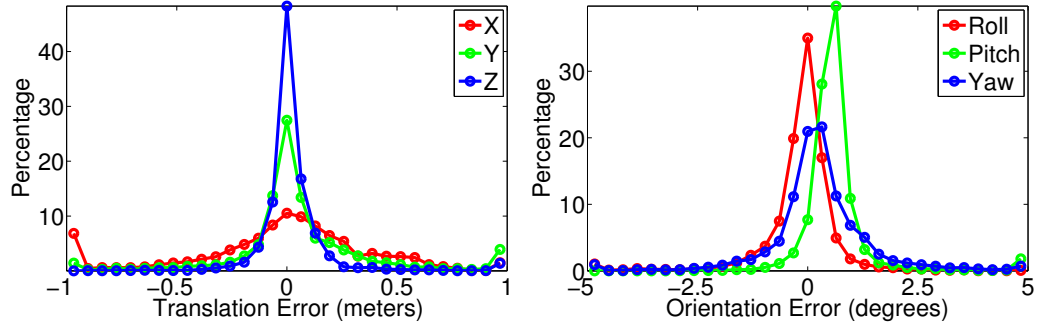
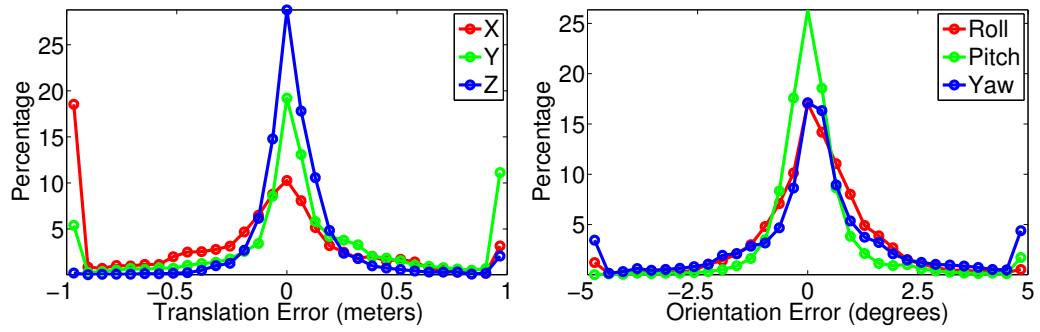


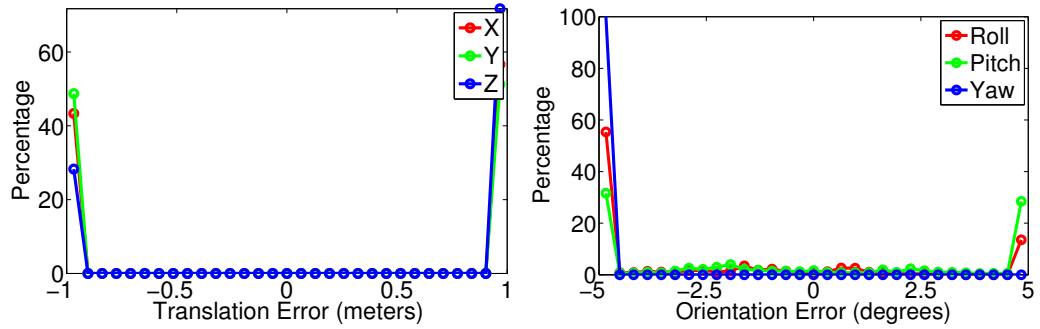
Figure 8.21: Overhead view of Route B showing the locations at which LAPS requested a restart for B3 $\rightarrow$ B1. The points at which LAPS requested a restart after failing to produce a positively classified pose for 5 seconds are denoted by a green plus. Also shown as red crosses are the locations along the route of the last positively classified LAPS pose before the subsequent restart, by definition 5 seconds earlier, connected to the restart by a magenta line. The kilometre markers are also shown in blue.



(a) Positively Classified LAPS Localisations

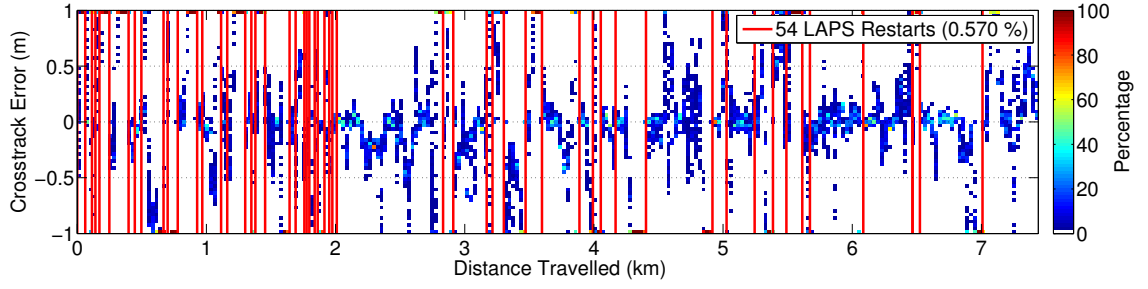


(b) LAPS Fused with Odometry

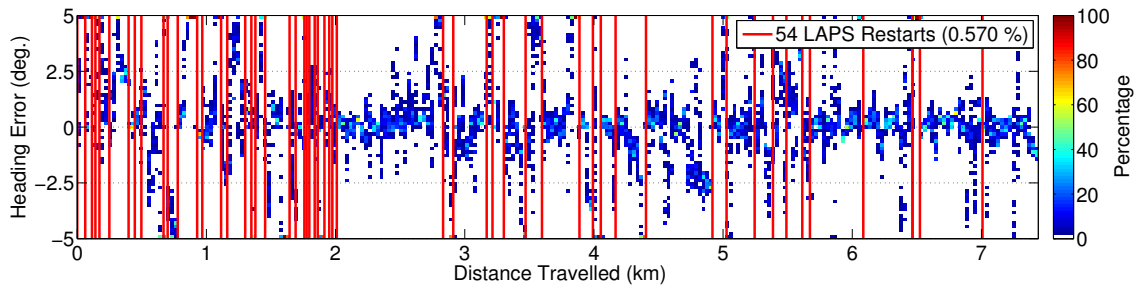


(c) INS

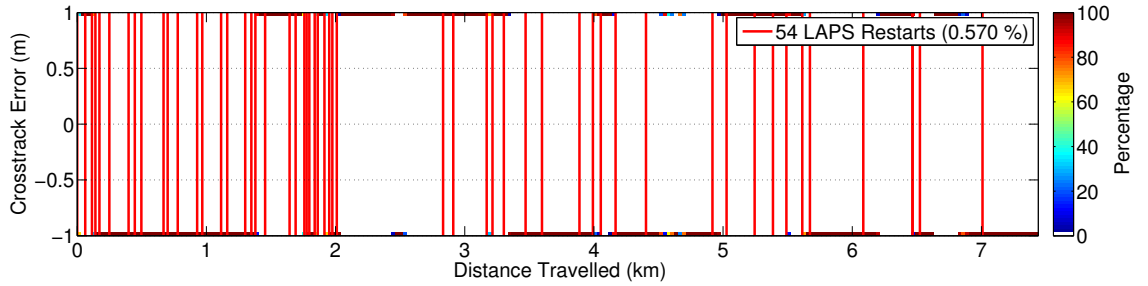
Figure 8.22: Histograms of Euler components of G_{Error} for LAPS, the Fused output and the INS for $B3 \rightarrow B1$. The histograms of G_{Error} are expressed in terms of the percentage of samples, for the LAPS localisations classified as positive: Figure 8.22a, the output after fusing these with odometry using SplineFusion: Figure 8.22b and for the INS: Figure 8.22c. Note that for B3, the INS failed shortly after the start.

8.3.7 Traversal: Oxford B1 $\rightarrow$ Prior: Oxford B3

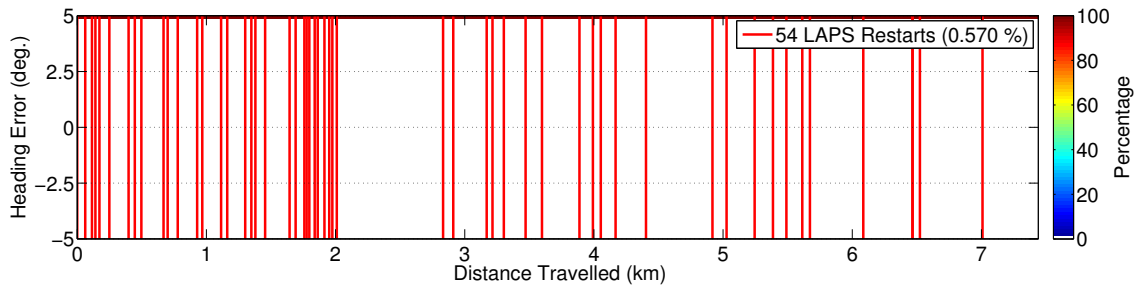
(a) LAPS Fused with Odometry: Crosstrack Error



(b) LAPS Fused with Odometry: Heading Error



(c) INS: Crosstrack Error



(d) INS: Heading Error

Figure 8.23: 2D Distance travelled vs crosstrack: Figure 8.23a and heading: Figure 8.23b error respectively for LAPS fused with odometry and the corresponding figures for the INS: Figures 8.23c and 8.23d for B1 $\rightarrow$ B3. Please refer to Section 8.2.3 for a detailed explanation of how to interpret these figures. Note that for B3, the INS failed shortly after the start.

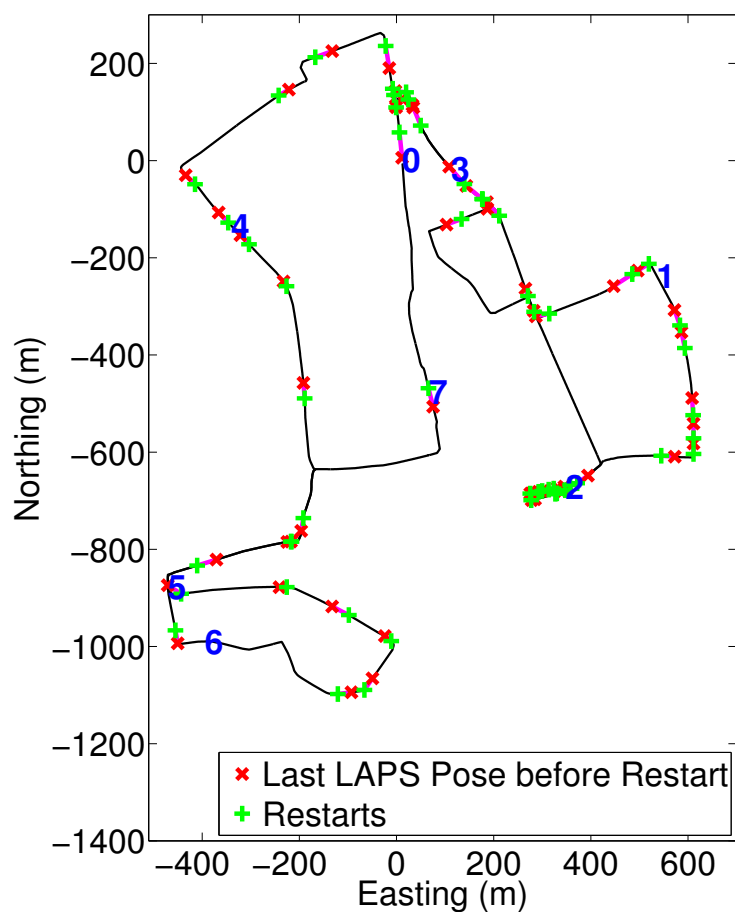
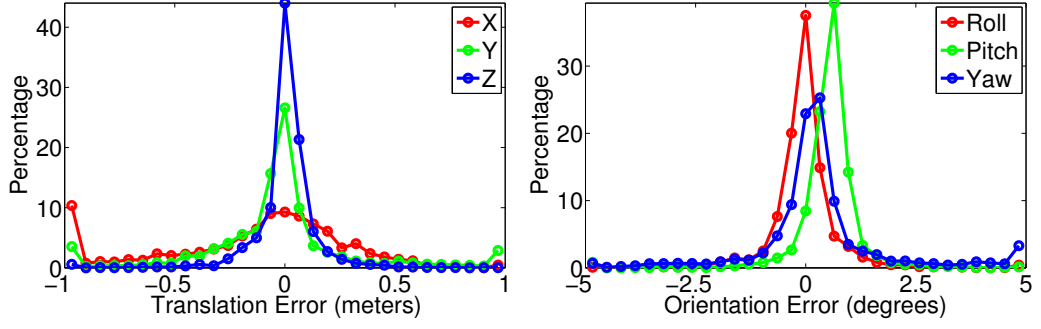
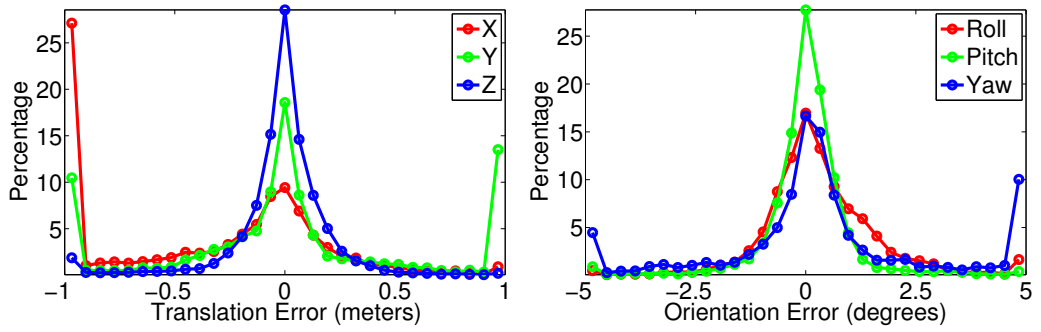


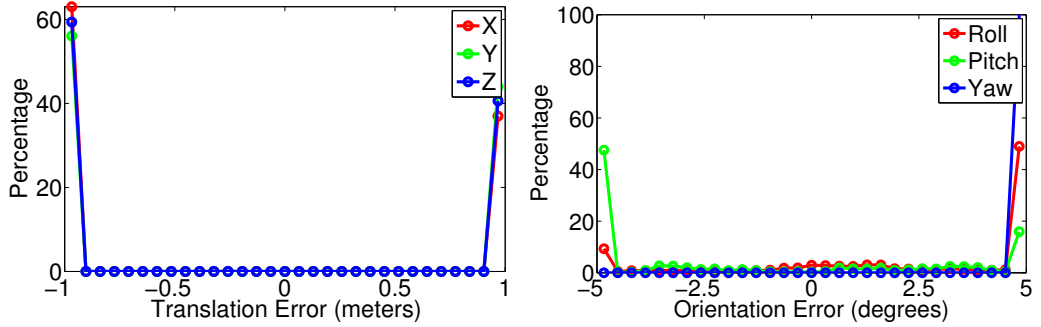
Figure 8.24: Overhead view of Route B showing the locations at which LAPS requested a restart for B1 $\rightarrow$ B3. The points at which LAPS requested a restart after failing to produce a positively classified pose for 5 seconds are denoted by a green plus. Also shown as red crosses are the locations along the route of the last positively classified LAPS pose before the subsequent restart, by definition 5 seconds earlier, connected to the restart by a magenta line. The kilometre markers are also shown in blue.



(a) Positively Classified LAPS Localisations

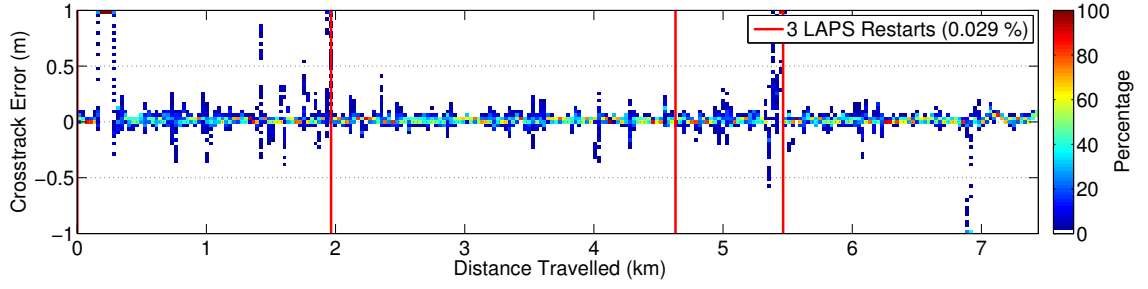


(b) LAPS Fused with Odometry

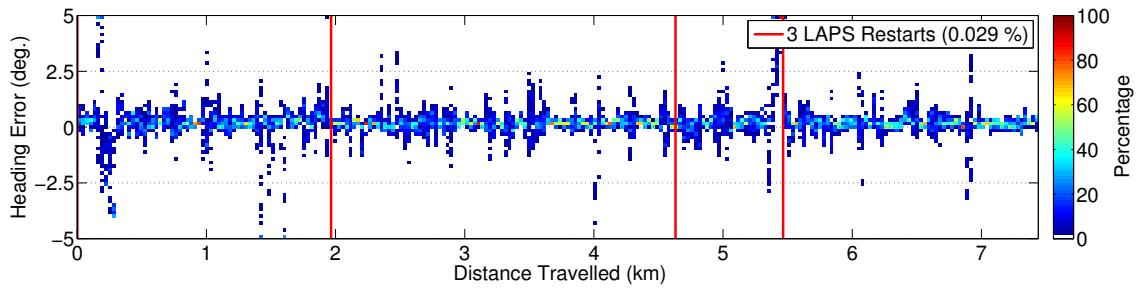


(c) INS

Figure 8.25: Histograms of Euler components of G_{Error} for LAPS, the Fused output and the INS for B1 $\rightarrow$ B3. The histograms of G_{Error} are expressed in terms of the percentage of samples, for the LAPS localisations classified as positive: Figure 8.25a, the output after fusing these with odometry using SplineFusion: Figure 8.25b and for the INS: Figure 8.25c. Note that for B3, the INS failed shortly after the start.

8.3.8 Traversal: Oxford B3 $\rightarrow$ Prior: Oxford B3

(a) LAPS Fused with Odometry: Crosstrack Error



(b) LAPS Fused with Odometry: Heading Error

Figure 8.26: 2D Distance travelled vs crosstrack: Figure 8.26a and heading: Figure 8.26b error respectively for LAPS fused with odometry for B3 $\rightarrow$ B3. Please refer to Section 8.2.3 for a detailed explanation of how to interpret these figures.

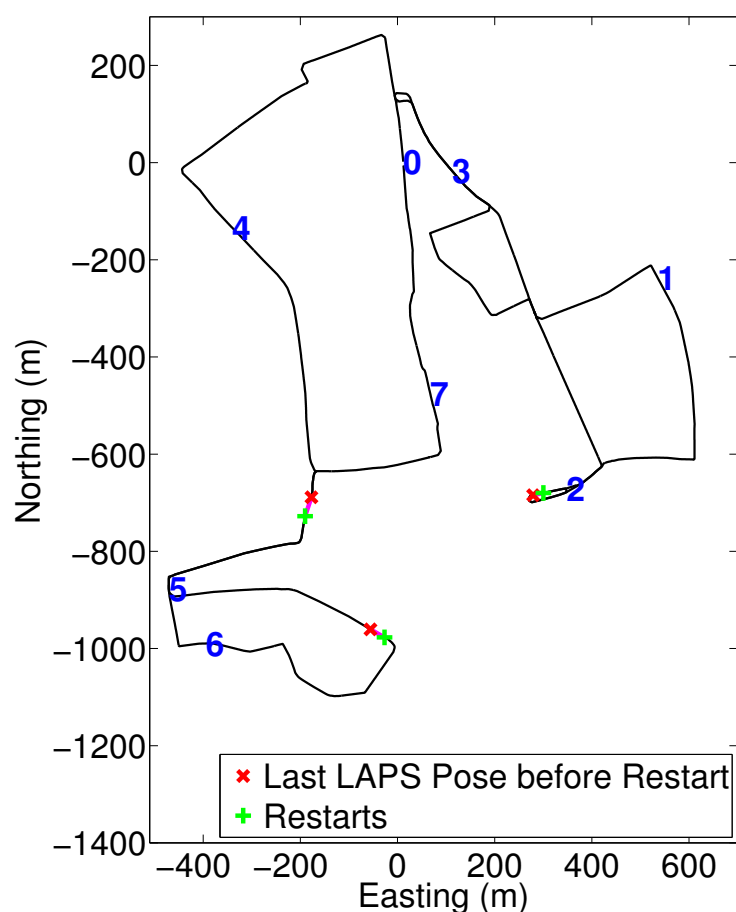


Figure 8.27: Overhead view of Route B showing the locations at which LAPS requested a restart for $B3 \rightarrow B3$. The points at which LAPS requested a restart after failing to produce a positively classified pose for 5 seconds are denoted by a green plus. Also shown as red crosses are the locations along the route of the last positively classified LAPS pose before the subsequent restart, by definition 5 seconds earlier, connected to the restart by a magenta line. The kilometre markers are also shown in blue.

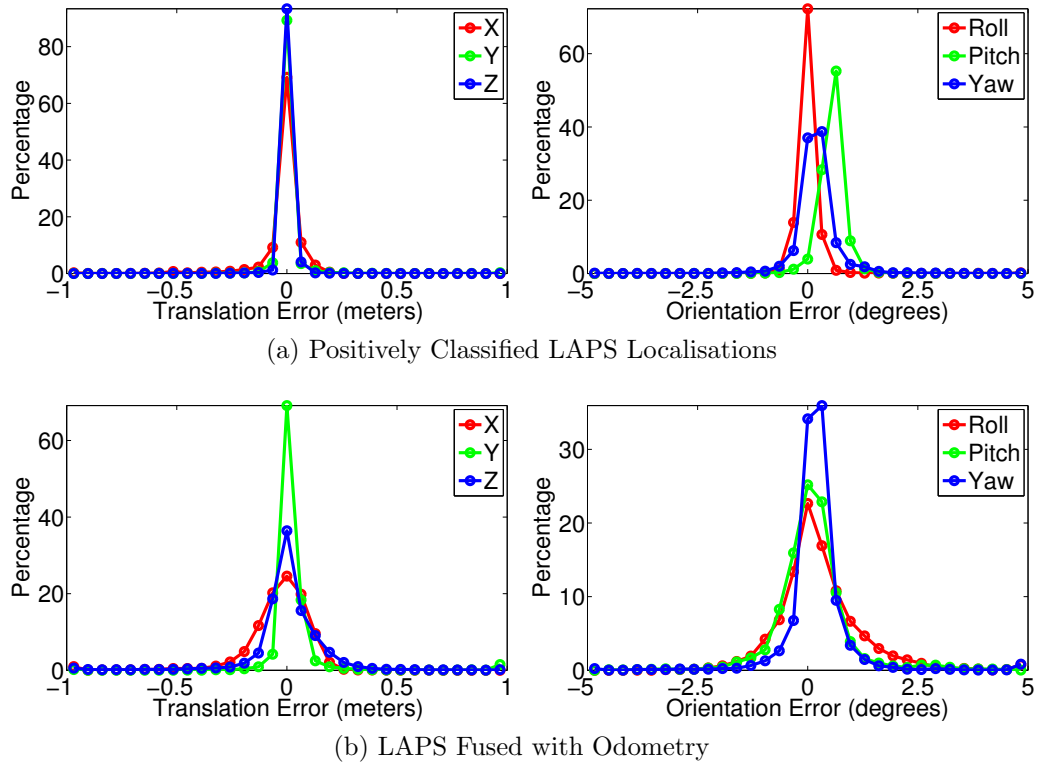
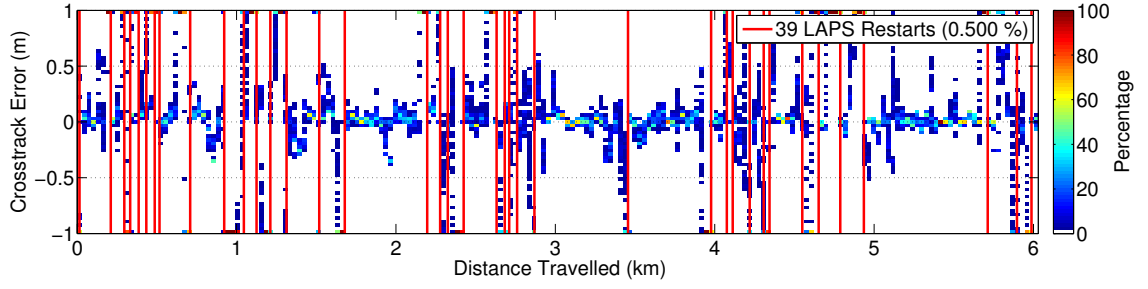
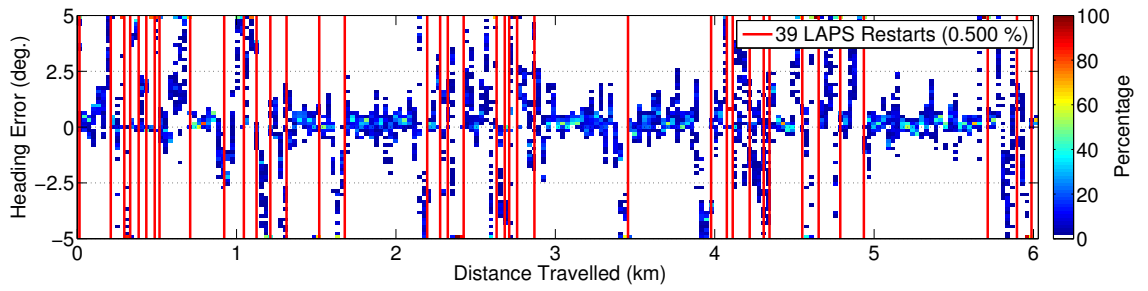


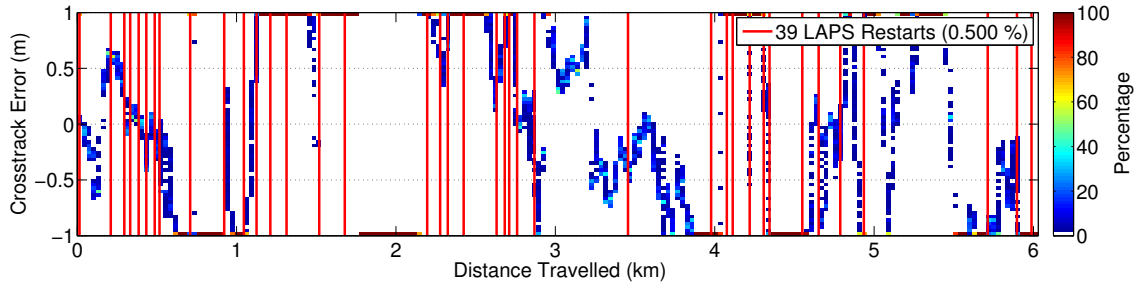
Figure 8.28: Histograms of Euler components of G_{Error} for LAPS and the Fused output for $B3 \rightarrow B3$. The histograms of G_{Error} are expressed in terms of the percentage of samples, for the LAPS localisations classified as positive: Figure 8.28a, and the output after fusing these with odometry using SplineFusion: Figure 8.28b.

8.3.9 Traversal: Oxford A1 $\rightarrow$ Prior: Oxford B1

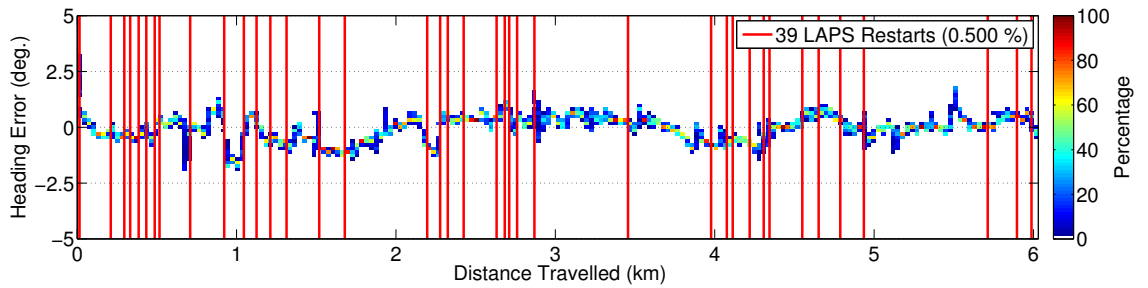
(a) LAPS Fused with Odometry: Crosstrack Error



(b) LAPS Fused with Odometry: Heading Error



(c) INS: Crosstrack Error



(d) INS: Heading Error

Figure 8.29: 2D Distance travelled vs crosstrack: Figure 8.29a and heading: Figure 8.29b error respectively for LAPS fused with odometry and the corresponding figures for the INS: Figures 8.29c and 8.29d for A1 $\rightarrow$ B1. Please refer to Section 8.2.3 for a detailed explanation of how to interpret these figures.

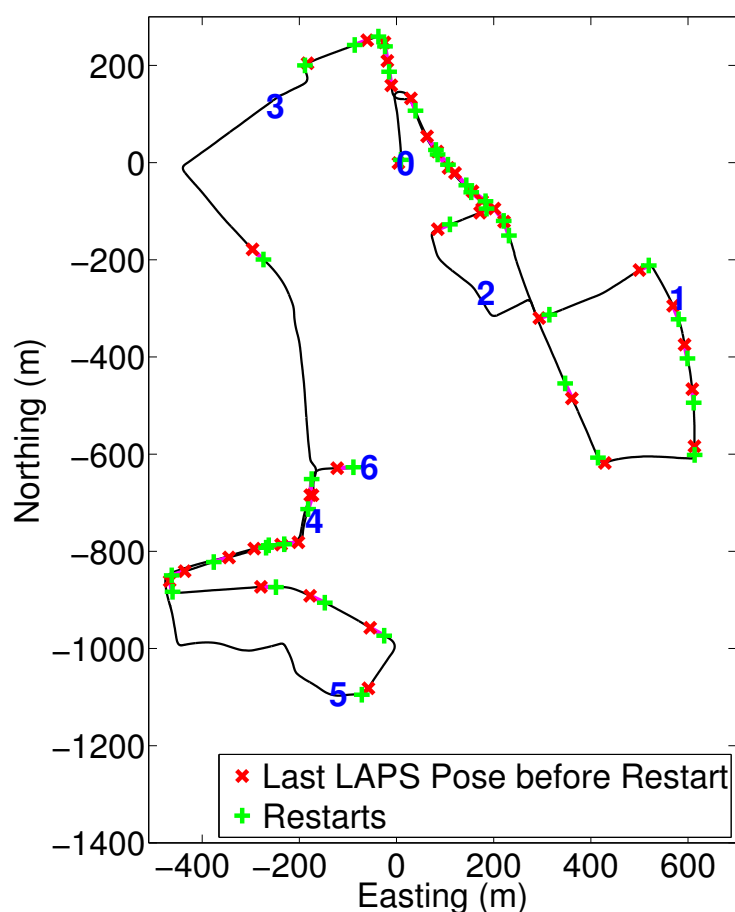
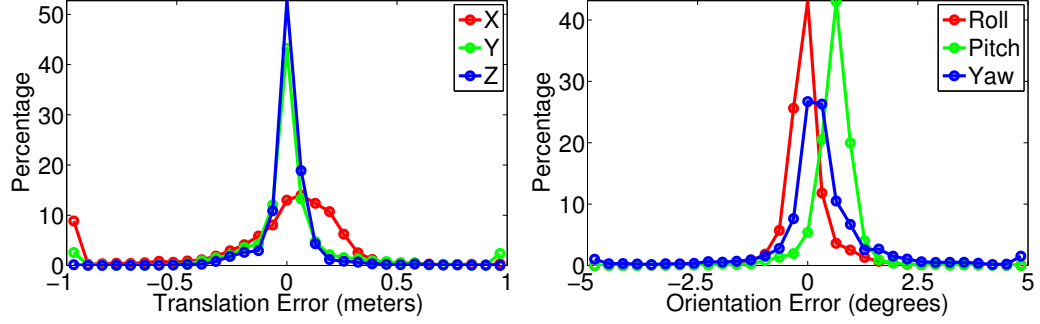
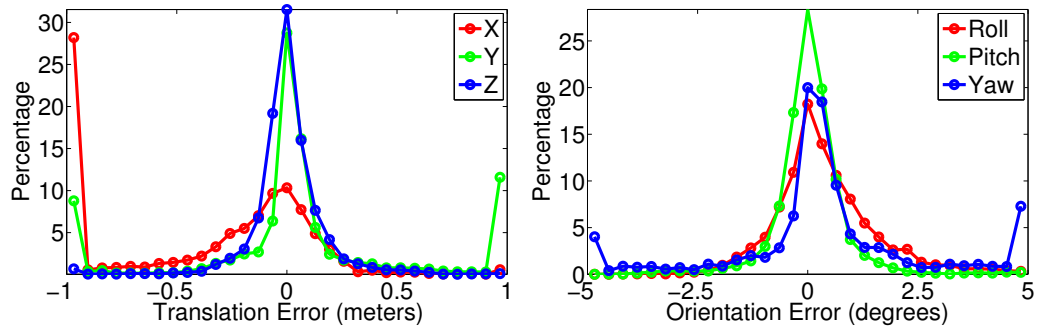


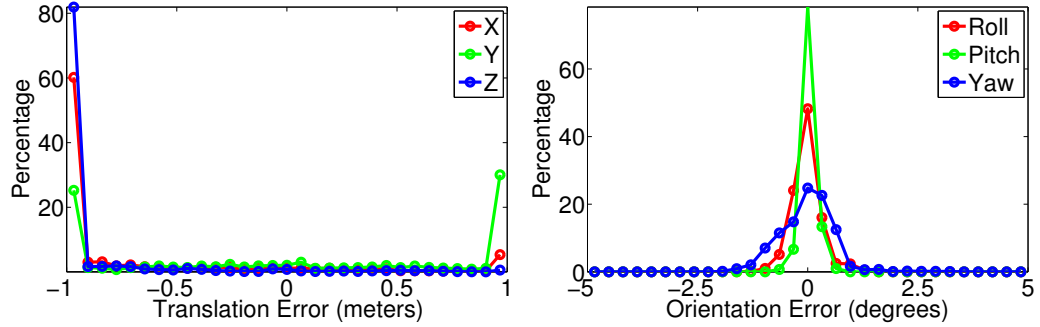
Figure 8.30: Overhead view of longest contiguous section of Route A with Route B, showing the locations at which LAPS requested a restart for A1 $\rightarrow$ B1. The points at which LAPS requested a restart after failing to produce a positively classified pose for 5 seconds are denoted by a green plus. Also shown as red crosses are the locations along the route of the last positively classified LAPS pose before the subsequent restart, by definition 5 seconds earlier, connected to the restart by a magenta line. The kilometre markers are also shown in blue.



(a) Positively Classified LAPS Localisations



(b) LAPS Fused with Odometry



(c) INS

Figure 8.31: Histograms of Euler components of G_{Error} for LAPS, the Fused output and the INS for $A1 \rightarrow B1$. The histograms of G_{Error} are expressed in terms of the percentage of samples, for the LAPS localisations classified as positive: Figure 8.31a, the output after fusing these with odometry using SplineFusion: Figure 8.31b and for the INS: Figure 8.31c.

8.4 Discussion

In this section, we review particular points of interest from the detailed experimental results in Sections 8.3.2 to 8.3.9. We advise the reader to read the summary results in Section 8.3.1 before reading this section and to use this section as an index into the detailed results figures.

8.4.1 LAPS Localiser Performance

We have seen from Figure 8.6 that the aggregate performance of the LAPS system is greater than that of the INS, particularly in terms of crosstrack error. Consider now the G_{Error} Euler component histograms for the positively classified LAPS localisations shown in Figures 8.13a, 8.16a and 8.22a, and contrast them with the equivalent figures for the fused solution and the INS given in Figures 8.13b, 8.16b and 8.22b and Figures 8.13c, 8.16c and 8.22c respectively.

The results for the positive LAPS localisations show strong, narrow peaks about zero error, with correspondingly small tails for the Y & Z translation components and all of the orientation components for all experiments. The peak for the X translation component has a lower maximum and is correspondingly wider, but still shows a clearly defined shape centered at zero error. The peaks for pitch and to a lesser extent yaw are offset by a small persistent bias of $\approx 0.65^\circ$ and $\approx 0.2^\circ$ respectively. In our definition of the Robot coordinate frame, X points forward along the vehicle trackline. As noted earlier in this work, it is common for urban scenes to result in limited visual change when moving along the road, for example see Figure 5.11. Thus as LAPS inherently uses differences in the appearance of the environment to localise, reduced confinement in the X-axis of the Robot frame should be expected.

As the small biases present in pitch & yaw are common to all experiments, irrespective of the datasets used, they are best described by residual errors in the

extrinsic calibration between the Ladybug2 and Bumblebee2 cameras and the LMS-151 Lidar on the Wildcat. This is an interesting result, which we shall return to in Section 9.1 as it suggests that our method could also be used for calibration, akin to that of [72, 85].

The results for the fused solutions show slightly wider peaks in all axes relative to the LAPS localiser results and for the datasets with lower overall performance, significant mass in the tails, for example for X in Figure 8.31b. As discussed previously in Section 8.3.1, this is because the fused solution results contains integrated odometry for the period when LAPS failed to return a positively classified pose. Particularly in the periods leading up to a restart, these poses are often in error. As LAPS only declares itself lost after failing to produce a positively classified pose for 5 seconds, integrating odometry for this period can produce a large lever arm that with only a small initial pose error results in significant pose errors along the arm. As our robot frame is defined such that X points forward along the vehicle trackline, if the last positively classified LAPS pose has a yaw error then integrating odometry will drive the vehicle off the true path. Thus resulting in a negative error in X by virtue of the definitions of G_{Error} and our $\text{SE}(3)$ convention from Section 2.1. This can also be seen from Figure 8.29a, where although the bulk of the crosstrack error is within $\pm 0.5\text{m}$, there are clear spikes before the LAPS restarts.

The absence of the pitch bias evident in the LAPS results from the fused results is a consequence of the prior imposed on the odometry observations that Z, roll and pitch be zero mean discussed in Section 7.5.2. The increase in the width of the peak in Z is also a consequence of this prior.

Reinforcing our assertion from Section 8.3.1 that our system outperforms the INS, the corresponding figures for the INS translation errors show no peak at all, with most of the mass at the boundaries, which aggregate all samples with errors larger than the specified range. This is particularly notable in Figure 8.11c, contrast with

Figure 8.11a, for datasets A1 & A2 which were collected only 40 minutes apart. The INS orientation errors are however broadly comparable to LAPS, with the heading error showing a lower variance across the datasets as evident from Figures 8.11b and 8.11d.

8.4.2 Solution Classifier Performance

The performance of the solution classifier, whilst not independently visible in the histograms of the Euler components of G_{Error} such as Figure 8.31a, is implied by the clear peaks at zero across all components and the heavy suppression of tail errors. From Figure 8.7 we can determine that the percentage of positively classified LAPS localisations is only 62% for $A1 \rightarrow B1$, thus the classifier has performed significant pruning of the data to produce Figure 8.31a. The question that remains, is whether the classifier was overzealous and removed correct LAPS poses thus reducing the overall performance, or whether it was correctly removing failed LAPS localisations and thus improving the performance. Figure 8.32 demonstrates that it is the latter, it extends Figure 8.31a by also showing the results computed before the classifier was applied in Figure 8.32a.

Comparing Figure 8.32b and Figure 8.32a, it is clear from the heavy suppression of the tail errors in Figure 8.32b that the LAPS localisations removed by the classifier after being classified as negative, were those with large errors. This demonstrates the generalisability and utility of the classifier developed in Section 7.4 and justifies its use, despite the additional computational cost this incurs.

8.4.3 Failure Cases

From the 2D distance vs error histograms and the maps showing the locations of the LAPS restarts on the traces of the routes, for example Figures 8.20a and 8.21, it is

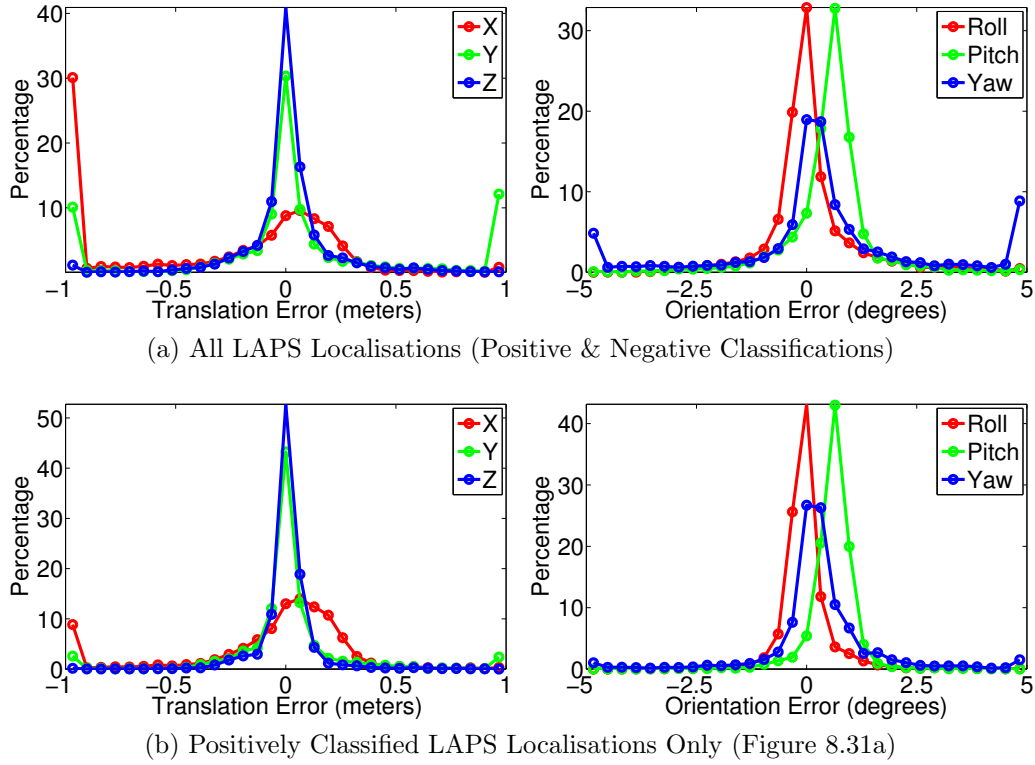


Figure 8.32: Demonstration of the effective suppression of failed LAPS localisations by the classifier for the $A1 \rightarrow B1$ experiment. Figure 8.32a shows the Euler components of G_{Error} for *all* of the LAPS localisations, i.e. those classified as negative, in addition to those classified as positive. Whilst there are clear peaks at zero, there are significant tail errors at the boundaries, particularly for X, Y & yaw. By comparison, Figure 8.32b (a reproduction of Figure 8.31a) which shows the results for only those classified as positive, the tail errors have been heavily suppressed. This demonstrates that the classifier is successfully classifying samples with large errors as negatives as desired, using a feature space formed solely by sampling the NID about the candidate solution.



Figure 8.33: Examples of areas where the Scene Prior provides poor structural confinement, taken from dataset B1. Both of the areas shown are areas in which LAPS frequently fails to produce positively classified poses and thus trigger LAPS restarts. The common theme is that there are no buildings sampled in the prior, and thus the only structure above the ground plane are trees which often have a homogeneous appearance. This absence of strong appearance cues for alignment on the structure results in either poor confinement, or no clear minima in the NID. In some cases, as in Figure 8.33b, this problem is partly a function of traffic. In the case shown, the buses are blocking the view of structure which would otherwise be useful for localisation.

clear that the LAPS restarts are not uniformly distributed. Indeed there are areas that often trigger restarts across multiple datasets, for example the Keble Road - Banbury Road junction and St Giles', located above the 0km marker and in the vicinity of the 7km marker in Figure 8.2b respectively. The Scene Prior built from dataset B1 for these areas is shown in Figure 8.33. The common theme is that the prior does not capture any buildings, and so the only significant structure above the ground plane consists of trees. As these have a homogenous appearance, as does the road, there are not sufficient appearance cues to constrain the solution which results in either poor confinement, or no clear minima in the NID. In the case of Figure 8.33b, structure that would be visible is hidden behind a wall of buses.

An additional source of LAPS restarts for experiments using Route B, particularly those involving B3, is areas which were more open and were thus more susceptible to significant illumination changes and shadows; especially those with alternating

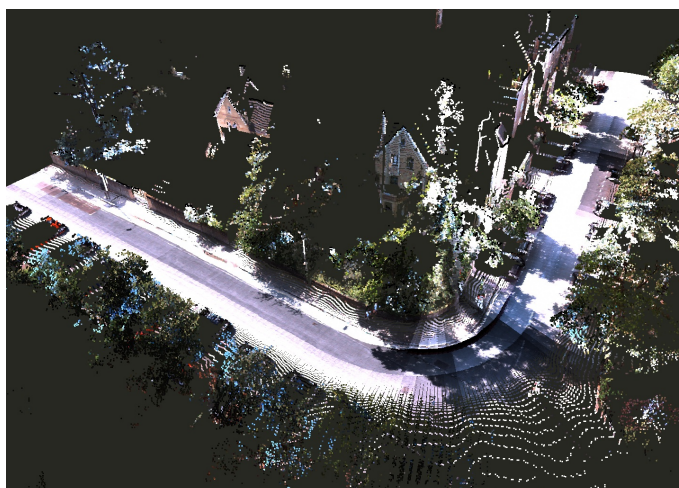


Figure 8.34: Example of the problem of overexposed images and strong shadows in dataset B3 distorting the appearance of the resulting Scene Prior for turning into Bevington Road from Banbury Road. As noted in Section 6.2, we use a custom software autoexposure controller for the Ladybug2 to improve its performance outdoors. Although it’s performance notably exceeds that of the hardware controller, in very sunny conditions where the vehicle is constantly driving into and out of direct sunlight the performance can degrade, resulting in over exposed images. As evident from the figure, large areas of the road and structure have been washed out. This removes important appearance cues for localisation and so results in poor confinement, or no clear minima in the NID.

shadows and bright sunshine. In these areas the autoexposure controller for the Ladybug2 often failed to respond adequately, typically overexposing the image thus removing important appearance cues for localisation. An example of this problem is shown in Figure 8.34 for the Banbury Road - Bevington Road junction which occurs at the top of Route B above the 0km marker in Figure 8.2b, built using dataset B3. This problem could be substantially reduced by a better autoexposure controller which preserved at least some of the appearance cues under large illumination changes. We experimented with an alternative approach in [55] combining LAPS with the Illumination Invariant colourspace transform, an example of which is shown in Figure 4.14. However, this requires that the images not be under or overexposed, and thus would not on its own mitigate the problem highlighted here.

Heavy traffic can also sometimes cause LAPS to fail to produce a positively

classified pose and request a restart, although in general we have found the system to be robust to traffic, as evident by the performance in the $B1 \rightleftharpoons B2$ experiments which were conducted in the afternoon with typical Oxford traffic. In general the relative height of the Ladybug2 on the Wildcat means that the structure is typically still visible over the tops of cars; buses and heavy tourist traffic, common hazards on Broad Street located at the 2km marker in Figure 8.2b, can however almost entirely block the view of the prior, resulting in a failure.

8.5 Conclusions

In this chapter we have demonstrated that the performance of the LAPS localisation system running in real-time over 57km of driving in central Oxford under a variety of different conditions up to a year apart consistently exceeds that of a high-grade commercial INS. We have also shown that the solution classifier developed in Section 7.4 generalises well, even to datasets captured under very different conditions to those of the training data. Furthermore, we have shown that the classifier maintains a high precision and so effectively prunes failed LAPS localisations from being incorporated into the fused solution, thus improving performance. Additionally, it acts as an effective mechanism to allow the system to independently determine when it is lost, an important quality in any practical localisation system.

Chapter 9

Conclusions

This chapter summarises the work described in this thesis and discusses some possible avenues for future work.

9.1 Summary of Contributions

This thesis has made the following contributions:

- **An information theoretic approach to monocular, metric 6-DoF localisation over large scales outdoors, leveraging a 3D coloured point-cloud Scene Prior** (Chapters 3 to 5). Our approach explicitly optimises for the pose which harmonises the information in a Shannon sense, about the appearance of the scene from the live images conditioned on the pose, with that of the prior. This enables low-cost localisation of a robot equipped with only monocular cameras by exploiting a Scene Prior captured beforehand by a survey vehicle equipped with expensive 3D imaging sensors. Our use of a Shannon information metric as an objective enables an intuitive, concise and robust formulation with a strong theoretical basis. We discuss in detail how such information metrics can be optimised efficiently and demonstrate their

inherent robustness. We also demonstrate an important differentiation in the utility of different information metrics and the relationships between them.

- **A solution classifier to independently identify localisation failures by classifying not only the value of the information metric objective at termination of the optimisation, but also its true shape obtained from sampling** (Section 7.4). This exploits the bounded metric property of our objective, the Normalised Information Distance (NID) and the characteristic response of information theoretic alignment tasks. We have demonstrated the effectiveness of this classifier and its generalisability to a vast array of different scene geometries and appearances under different conditions in central Oxford and the resulting improvement in the performance of the overall system from its use.
- **A real-time capable implementation of our approach using commodity GPU hardware** (Chapter 7). We described in detail the development of our real-time implementation on commodity hardware and how we use the Spline Fusion algorithm [53] to fuse LAPS observations with odometry in a continuous time framework. We demonstrated that the performance of the resulting system exceeds that of a high-grade commercial INS system on 57km of driving in central Oxford over a range of different conditions, times of day and year. We also described in detail the approach by which the coloured 3D pointcloud Scene Priors used throughout this work are generated.

9.2 Future Work

There are a wide variety of avenues for developing the work described in this thesis; some particular areas of interest include:

- **Sufficiency of the Scene Prior.** It is clear from the results in Chapter 8 that not all priors are equally good for localisation. In particular, some areas may have insufficient combined geometric and appearance cues, resulting in no clear minima. Either from the inherent nature of the environment or because the appearance captured by the prior does not provide a representative segmentation. Thus an important question is whether the sufficiency of a prior can be accurately assessed beforehand; whether offline before it is used for localisation, or ideally online as it is being captured. This would be important for determining the level of coverage of an environment and efficiently updating it.
- **Selection & Management of Scene Priors.** Although the NID is a robust metric, the system still performs better when the appearance of the prior is closer to that of the live imagery. Following from the notion of ‘Experience’ based navigation from [11], rather than trying to characterise how the appearance of an environment changes with time in a generative fashion, multiple appearance profiles could be stored for a given Scene Prior. Which profile was used online for any given case could then be optimised to achieve the best performance, and indeed how this similarity should be quantified in advance is an interesting question. The results from this work could then be used as a stepping stone towards the development of generative models to allow generalisation of expected appearance changes to environments which have not been revisited by the survey vehicle.
- **Point importance weighting using online semantic information.** The point importance weighting introduced in Section 5.6 was not used in the results presented in Chapter 8. This is because although the system described in Chapter 7 currently has preliminary support for weight images, a real-time

implementation of the distraction suppression system from [64] is currently in development, and so has not yet been integrated into the system. A key benefit of the approach described using weight image is that it allows semantic information determined separately online from the live camera imagery to be used to aid localisation. For example, the use of the distraction suppression mask from [64] would provide improved robustness to dynamic objects and a semantic segmentation of the scene such as that obtained from [35] would allow components expected to be more useful for localisation to be weighted more heavily.

- **Use of alternative and mixed appearance modalities.** In the results presented in Chapter 8 we used exclusively the standard gray colourspace for localisation. In [55], we successfully demonstrated the use of an illumination invariant colourspace instead. A significant advantage of our choice of an information metric objective is that the appearance spaces of the prior and live data do not have to be the same. The exploration of the use of other colourspace, sensing modalities and their cross combinations is a potentially rich seam of research. Indeed recent work [122], is already investigating this though the use of lidar intensity compared to grayscale imagery.
- **Use of Dense Mesh Scene Prior models.** In this work we used 3D point-clouds as our Scene Prior representation, which are attractive as they represent the native form in which the data is captured and are easy to manipulate. However, they can be inefficient in terms of storage and mean that the appearance prior is constrained to the sampling resolution of the lidar. The latter can result in irregular coverage of parts of the scene, particularly around sharp corners for our experimental setup, as shown in Figure 6.4. An alternative representation would be to use a textured mesh, an approach taken by [9].

This has the disadvantage of changing the points used at each iteration in the optimisation, resulting in a non-smooth objective. However it natively captures the one point per pixel subsampling described in Section 5.5 and also allows for a potentially more compact Scene Prior, reducing the overall storage requirements.

- **Extension to pure lidar navigation using synthetic images.** Using the approach described in this work, an alternative laser based localisation system inspired by [3] could be formulated where the robot to be localised had a relatively low-cost 2D lidar. Using an odometry source, the robot could build a local textured mesh describing a local 3D ‘swathe’ of the scene, from which synthetic camera views could be synthesised and used to localise against a Scene Prior using the method described in this thesis. This would have the attraction that the synthetic camera views would only have to be generated once per swathe, rather than at every iteration of the optimisation.
- **Target-free Extrinsic Calibration.** As noted earlier in Section 8.4, the results in Chapter 8 highlight a small error in the extrinsic calibration of the sensors on the Wildcat. This suggests that a variation of our approach could also be used for target-free extrinsic calibration. Indeed, the formulation for lidar-camera calibration using the mutual information presented in [84, 85] is inherently closely related to the localisation formulation in this work, as the problems are, in some sense, duals.
- **Use of rolling shutter cameras.** As noted in Section 3.1.1, we implicitly assumed the use of global shutter cameras in this thesis, in line with most work to date using vision for metric localisation. However, there is no conceptual limitation that prevents our approach from being applied to rolling shutter cameras which are typically cheaper and more prevalent. This would

complicate the reprojection of the points in the prior into the cameras, as the columns (or rows) of the images are captured sequentially in time rather than instantaneously. However, provided that the odometry pose source could be interpolated over the capture period, something aided by our use of Spline Fusion, the prior could still be accurately reprojected into the image and the p_{IS} , and thus the NID, could be computed. However depending upon the application, the increased complexity and computational requirements for using rolling shutter cameras might outweigh the additional initial cost of global shutter cameras.

9.3 Concluding Remark

This thesis set out to demonstrate that by exploiting 3D coloured pointcloud Scene Priors of the environment captured by a survey vehicle using expensive sensors, a subsequent robot could successfully localise using only a collection of low-cost monocular cameras. The information theoretic approach to monocular, metric 6-DoF localisation described in this work has been shown to outperform a high-grade commercial INS system in dense urban areas in central Oxford under a variety of conditions. We hope that the development of our approach detailed here will be useful to future researchers in the field and contribute to the wider goal of fully autonomous robots.

Bibliography

- [1] S. Agarwal, K. Mierle, and Others, “Ceres solver,” <http://ceres-solver.org>.
- [2] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (SLAM): Part II,” *Robotics & Automation Magazine, IEEE*, vol. 13, no. 3, pp. 108–117, 2006.
- [3] I. Baldwin and P. Newman, “Road vehicle localization with 2D push-broom lidar and 3D priors,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 2611–2617.
- [4] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” in *Computer Vision ECCV 2006*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds. Springer Berlin / Heidelberg, 2006, vol. 3951, pp. 404–417.
- [5] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.
- [6] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O’Reilly Media, 2008.
- [7] R. H. Byrd, P. H. Lu, J. Nocedal, and C. Y. Zhu, “A limited memory algorithm for bound constrained optimization,” *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, Sep. 1995.
- [8] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, “BRIEF: Computing a local binary descriptor very fast,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 7, pp. 1281–1298, July 2012.
- [9] G. Caron, A. Dame, and E. Marchand, “Direct model based visual pose estimation: the mutual information feature,” *Image and Vision Computing*, 2014.
- [10] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, Apr. 2011.
- [11] W. Churchill and P. Newman, “Experience-based navigation for long-term localisation,” *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1645–1661, Jan. 2014.

- [12] W. Churchill, “Experience Based Navigation: Theory, Practice and Implementation,” Ph.D. dissertation, University of Oxford, 2012.
- [13] A. Comport, E. Malis, and P. Rives, “Real-time quadrifocal visual odometry,” *The International Journal of Robotics Research*, vol. 29, no. 2-3, pp. 245–266, 2010.
- [14] P. Corke, R. Paul, W. Churchill, and P. Newman, “Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 2085–2092.
- [15] M. Cummins and P. Newman, “Appearance-only SLAM at large scale with FAB-MAP 2.0,” *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, Aug. 2011.
- [16] A. Dame and E. Marchand, “Accurate real-time tracking using mutual information,” in *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*, Oct 2010, pp. 47–56.
- [17] A. Dame and E. Marchand, “Mutual information-based visual servoing,” *Robotics, IEEE Transactions on*, vol. 27, no. 5, pp. 958–969, Oct 2011.
- [18] A. Dame and E. Marchand, “A new information theoretic approach for appearance-based navigation of non-holonomic vehicle,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 2459–2464.
- [19] A. Dame and E. Marchand, “Second-order optimization of mutual information for real-time image registration,” *Image Processing, IEEE Transactions on*, vol. 21, no. 9, pp. 4190–4203, Sept 2012.
- [20] A. Dame, “A unified direct approach for visual servoing and visual tracking using mutual information,” Ph.D. dissertation, Université de Rennes 1, 2011.
- [21] A. Dame and E. Marchand, “Using mutual information for appearance-based visual path following,” *Robotics and Autonomous Systems*, vol. 61, no. 3, pp. 259 – 270, 2013.
- [22] A. Davison, “Real-time simultaneous localisation and mapping with a single camera,” in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, Oct 2003, pp. 1403–1410 vol.2.
- [23] A. Davison, I. Reid, N. Molton, and O. Stasse, “MonoSLAM: Real-time single camera SLAM,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 6, pp. 1052–1067, June 2007.

- [24] N. Dowson and R. Bowden, “A unifying framework for mutual information methods for use in non-linear optimisation,” in *Computer Vision ECCV 2006*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 3951, pp. 365–378.
- [25] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: Part I,” *Robotics & Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99–110, 2006.
- [26] E. Eade and T. Drummond, “Scalable monocular SLAM,” in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, June 2006, pp. 469–476.
- [27] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-scale direct monocular SLAM,” in *Computer Vision ECCV 2014*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2014, vol. 8690, pp. 834–849.
- [28] P. Furgale, T. Barfoot, and G. Sibley, “Continuous-time batch estimation using temporal basis functions,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 2088–2095.
- [29] P. Furgale and T. D. Barfoot, “Visual teach and repeat for long-range rover autonomy,” *Journal of Field Robotics*, vol. 27, no. 5, pp. 534–560, 2010.
- [30] B. Gaster, L. Howes, P. Mistry, D. R. Kaeli, and D. Schaa, *Heterogeneous Computing with OpenCL*. Elsevier, 2011.
- [31] A. Glover, W. Maddern, M. Milford, and G. Wyeth, “FAB-MAP+ RatSLAM: Appearance-based SLAM for multiple times of day,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 3507–3512.
- [32] D. Goldberg, “What every computer scientist should know about floating-point arithmetic,” *ACM Comput. Surv.*, vol. 23, no. 1, pp. 5–48, Mar. 1991.
- [33] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.
- [34] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. Springer, 2009.
- [35] H. He and B. Upcroft, “Nonparametric semantic segmentation for 3D street scenes,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 3697–3703.
- [36] K. Ho and P. Newman, “Detecting loop closure with scene sequences,” *International Journal of Computer Vision*, vol. 74, no. 3, pp. 261–286, 2007.

- [37] A. Kelly, *Mobile Robotics: Mathematics, Models, and Methods*. Cambridge University Press, 2013.
- [38] M.-J. Kim, M.-S. Kim, and S. Y. Shin, “A general construction scheme for unit quaternion curves with simple high order derivatives,” in *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’95. ACM, 1995, pp. 369–376.
- [39] G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces,” in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1–10.
- [40] G. Klein and D. W. Murray, “Full-3D edge tracking with a particle filter.” in *BMVC*, 2006, pp. 1119–1128.
- [41] S. Klein, M. Staring, and J. Pluim, “Evaluation of optimization methods for nonrigid medical image registration using mutual information and B-splines,” *Image Processing, IEEE Transactions on*, vol. 16, no. 12, pp. 2879–2890, Dec 2007.
- [42] K. Konolige and M. Agrawal, “FrameSLAM: From bundle adjustment to real-time visual mapping,” *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1066–1077, Oct 2008.
- [43] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua, “View-based maps,” *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 941–957, 2010.
- [44] A. Kraskov, H. Stögbauer, R. G. Andrzejak, and P. Grassberger, “Hierarchical clustering based on mutual information,” *eprint arXiv:q-bio/0311039*, Nov. 2003.
- [45] J. Lapresté and Y. Mezouar, “A hessian approach to visual servoing,” in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1, Sept 2004, pp. 998–1003 vol.1.
- [46] J. Levinson and S. Thrun, “Robust vehicle localization in urban environments using probabilistic maps,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 4372–4378.
- [47] J. Levinson, “Automatic laser calibration, mapping, and localization for autonomous vehicles,” Ph.D. dissertation, Stanford University, 2011.
- [48] J. Levinson and S. Thrun, “Automatic calibration of cameras and lasers in arbitrary scenes,” in *Proc. 13th IFRR Intl Symp. Experimental Robotics (ISER12), Quebec City, Canada*, 2012.

- [49] M. Li and A. I. Mourikis, “High-precision, consistent EKF-based visual-inertial odometry,” *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, Jun. 2013.
- [50] M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang, “An information-based sequence distance and its application to whole mitochondrial genome phylogeny,” *Bioinformatics*, vol. 17, no. 2, pp. 149–154, 2001.
- [51] M. Li, X. Chen, X. Li, B. Ma, and P. Vitanyi, “The similarity metric,” *Information Theory, IEEE Transactions on*, vol. 50, no. 12, pp. 3250 – 3264, 2004.
- [52] S. Lovegrove, A. Davison, and J. Ibanez-Guzman, “Accurate visual odometry from a rear parking camera,” in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, June 2011, pp. 788–793.
- [53] S. Lovegrove, A. Patron-Perez, and G. Sibley, “Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras,” in *British Machine Vision Conference (BMVC13)*, 2013.
- [54] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.
- [55] W. Maddern, A. Stewart, and P. Newman, “LAPS-II: 6-DoF day and night visual localisation with prior 3D structure for autonomous road vehicles,” in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, June 2014, pp. 330–337.
- [56] W. Maddern, M. Milford, and G. Wyeth, “CAT-SLAM: probabilistic localisation and mapping using a continuous appearance-based trajectory,” *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 429–451, 2012.
- [57] W. Maddern, A. Stewart, C. McManus, B. Upcroft, W. Churchill, and P. Newman, “Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles,” in *Visual Place Recognition in Changing Environments Workshop, Robotics and Automation (ICRA), 2014 IEEE International Conference on*, Hong Kong, China, May 2014.
- [58] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens, “Multimodality image registration by maximization of mutual information,” *Medical Imaging, IEEE Transactions on*, vol. 16, no. 2, pp. 187–198, April 1997.
- [59] F. Maes, D. Vandermeulen, and P. Suetens, “Medical image registration using mutual information,” *Proceedings of the IEEE*, vol. 91, no. 10, pp. 1699–1722, Oct 2003.

- [60] F. Maes, D. Vandermeulen, and P. Suetens, “Comparative evaluation of multiresolution optimization strategies for multimodality image registration by maximization of mutual information,” *Medical Image Analysis*, vol. 3, no. 4, pp. 373 – 386, 1999.
- [61] J. Mason, S. Ricco, and R. Parr, “Textured occupancy grids for monocular localization without features,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 5800–5806.
- [62] P. Maybeck, *Stochastic Models, Estimation, and Control*, ser. Mathematics in Science and Engineering. Elsevier Science, 1982.
- [63] C. McManus, W. Churchill, W. Maddern, A. Stewart, and P. Newman, “Shady dealings: Robust, long-term visual localisation using illumination invariance,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 901–906.
- [64] C. McManus, W. Churchill, A. Napier, B. Davis, and P. Newman, “Distraction suppression for vision-based pose estimation at city scales,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 3762–3769.
- [65] C. McManus, P. Furgale, B. Stenning, and T. D. Barfoot, “Lighting-invariant visual teach and repeat using appearance-based lidar,” *Journal of Field Robotics*, vol. 30, no. 2, pp. 254–287, 2013.
- [66] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, “A constant time efficient stereo SLAM system,” in *Proceedings of the British Machine Vision Conference (BMVC)*, 2009.
- [67] 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*, vol. 94, no. 2, pp. 198–214, 2011.
- [68] M. Meilă, “Comparing clusterings by the variation of information,” in *Learning Theory and Kernel Machines*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, vol. 2777, pp. 173–187.
- [69] M. Milford, “Visual route recognition with a handful of bits,” in *Robotics: Science and Systems Proceedings*, Sydney, Australia, July 2012.
- [70] M. Milford and G. Wyeth, “Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 1643–1649.
- [71] R. Murray, Z. Li, S. Sastry, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Taylor & Francis, 1994.

- [72] A. Napier, P. Corke, and P. Newman, “Cross-calibration of push-broom 2D lidars and cameras in natural scenes,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 3679–3684.
- [73] A. Napier and P. Newman, “Generation and exploitation of local orthographic imagery for road vehicle localisation,” in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, June 2012, pp. 590–596.
- [74] A. Napier, G. Sibley, and P. Newman, “Real-time bounded-error pose estimation for road vehicles using vision,” in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, Sept 2010, pp. 1141–1146.
- [75] T. Naseer, L. Spinello, W. Burgard, and C. Stachniss, “Robust visual robot localization across seasons using network flows,” in *Proceedings of the AAAI National Conference on Artificial Intelligence (AAAI)*, 2014.
- [76] P. Neubert, N. Sunderhauf, and P. Protzel, “Appearance change prediction for long-term navigation across seasons,” in *Mobile Robots (ECMR), 2013 European Conference on*, Sept 2013, pp. 198–203.
- [77] R. A. Newcombe, S. Lovegrove, and A. Davison, “DTAM: Dense tracking and mapping in real-time,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*, Nov 2011, pp. 2320–2327.
- [78] H. Nguyen, *GPU Gems 3*. Addison-Wesley Professional, 2007.
- [79] J. Nocedal and S. Wright, *Numerical Optimization*, ser. Springer Series in Operations Research and Financial Engineering. Springer, 2006.
- [80] S. Nuske, J. Roberts, and G. Wyeth, “Robust outdoor visual localization using a three-dimensional-edge map,” *Journal of Field Robotics*, vol. 26, no. 9, pp. 728–756, Sep. 2009.
- [81] NVIDIA Corporation, *NVIDIA OpenCL Best Practices Guide*.
- [82] Oxford Technical Solutions Limited, *OxTS RT User Manual*, 2014.
- [83] G. Pandey, J. McBride, S. Savarese, and R. Eustice, “Extrinsic calibration of a 3D laser scanner and an omnidirectional camera,” in *7th IFAC Symposium on Intelligent Autonomous Vehicles*, 2010.
- [84] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, “Automatic targetless extrinsic calibration of a 3D lidar and camera by maximizing mutual information,” in *Proceedings of the AAAI National Conference on Artificial Intelligence*, Toronto, Canada, July 2012, pp. 2053–2059.

- [85] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, “Automatic extrinsic calibration of vision and lidar by maximizing mutual information,” *Journal of Field Robotics*, 2014.
- [86] G. Panin and A. Knoll, “Mutual information-based 3D object tracking,” *International Journal of Computer Vision*, vol. 78, no. 1, pp. 107–118, 2008.
- [87] E. Pepperell, P. Corke, and M. Milford, “All-environment visual place recognition with SMART,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 1612–1618.
- [88] L. Piegl and W. Tiller, *The NURBS Book*. Springer Verlag, 1997.
- [89] J. Pluim, J. Maintz, and M. Viergever, “Mutual-information-based registration of medical images: A survey,” *Medical Imaging, IEEE Transactions on*, vol. 22, no. 8, pp. 986–1004, Aug 2003.
- [90] K. Qin, “General matrix representations for B-splines,” *The Visual Computer*, vol. 16, no. 3-4, pp. 177–186, 2000.
- [91] D. Ruijters, B. M. ter Haar Romeny, and P. Suetens, “Efficient GPU-based texture interpolation using uniform B-splines,” *Journal of Graphics, GPU, and Game Tools*, vol. 13, no. 4, pp. 61–69, 2008.
- [92] D. Ruijters and P. Thvenaz, “GPU prefilter for accurate cubic B-spline interpolation,” *The Computer Journal*, 2010.
- [93] D. Scaramuzza, A. Martinelli, and R. Siegwart, “A toolbox for easily calibrating omnidirectional cameras,” in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct 2006, pp. 5695–5701.
- [94] B. M. Scherzinger, “Precise robust positioning with inertial/GPS RTK,” *Proc ION GPS-2000. Salt Lake City, September*, pp. 19–22, 2000.
- [95] C. E. Shannon, “A Mathematical Theory of Communication,” *The Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.
- [96] M. Sheehan, A. Harrison, and P. Newman, “Continuous vehicle localisation using sparse 3D sensing, kernelised Rényi distance and fast Gauss transforms,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 398–405.
- [97] M. Sheehan, A. Harrison, and P. Newman, “Automatic self-calibration of a full field-of-view 3D n-laser scanner,” in *Experimental Robotics*, ser. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, 2014, vol. 79, pp. 165–178.

- [98] G. Sibley, C. Mei, I. Reid, and P. Newman, “Vast-scale outdoor navigation using adaptive relative bundle adjustment,” *The International Journal of Robotics Research*, vol. 29, no. 8, p. 958, 2010.
- [99] G. Silveira, E. Malis, and P. Rives, “An efficient direct approach to visual SLAM,” *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 969–979, Oct 2008.
- [100] G. Silveira and E. Malis, “Unified direct visual tracking of rigid and deformable surfaces under generic illumination changes in grayscale and color images,” *International Journal of Computer Vision*, vol. 89, no. 1, pp. 84–105, 2010.
- [101] F. Steinbrucker, J. Sturm, and D. Cremers, “Real-time visual odometry from dense RGB-D images,” in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, Nov 2011, pp. 719–722.
- [102] A. Stewart and P. Newman, “LAPS - localisation using appearance of prior structure: 6-DoF monocular camera localisation using prior pointclouds,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 2625–2632.
- [103] H. Strasdat, “Local accuracy and global consistency for efficient SLAM,” Ph.D. dissertation, Imperial College London, 2012.
- [104] H. Strasdat and S. Lovegrove, “Sophus: C++ implementation of Lie Groups using Eigen,” 2011–2014, retrieved on 05/09/14. [Online]. Available: <https://github.com/stevenlovegrove/Sophus>
- [105] H. Strasdat, J. Montiel, and A. J. Davison, “Scale drift-aware large scale monocular SLAM,” in *Robotics: Science and Systems*, vol. 2, no. 3, 2010, p. 5.
- [106] C. Strecha, W. von Hansen, L. Van Gool, P. Fua, and U. Thoennessen, “On benchmarking camera calibration and multi-view stereo for high resolution imagery,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, June 2008, pp. 1–8.
- [107] C. Studholme, D. Hill, and D. Hawkes, “An overlap invariant entropy measure of 3D medical image alignment,” *Pattern Recognition*, vol. 32, no. 1, pp. 71 – 86, 1999.
- [108] P. Thevenaz and M. Unser, “Optimization of mutual information for multiresolution image registration,” *Image Processing, IEEE Transactions on*, vol. 9, no. 12, pp. 2083–2099, Dec 2000.
- [109] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT press, 2005.
- [110] C. H. Tong, P. Furgale, and T. D. Barfoot, “Gaussian Process Gauss-Newton for non-parametric simultaneous localization and mapping,” *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 507–525, May 2013.

- [111] C. H. Tong, S. Anderson, H. Dong, and T. D Barfoot, "Pose interpolation for laser-based visual odometry," *Journal of Field Robotics*, vol. 31, no. 5, pp. 731–757, Aug. 2014.
- [112] M. Unser, "Splines: a perfect fit for signal and image processing," *Signal Processing Magazine, IEEE*, vol. 16, no. 6, pp. 22–38, Nov 1999.
- [113] Velodyne, *Velodyne HDL-64E: A High Definition Lidar Sensor for 3-D Applications*, 2007.
- [114] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, Dec. 2010.
- [115] P. Viola and W. I. Wells, "Alignment by maximization of mutual information," in *Computer Vision, 1995. Proceedings., Fifth International Conference on*, 1995, pp. 16–23.
- [116] D. Wang, I. Posner, and P. Newman, "What could move? finding cars, pedestrians and bicyclists in 3D laser data," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 4038–4044.
- [117] M. Warren, D. McKinnon, and B. Upcroft, "Online calibration of stereo rigs for long-term autonomy," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 3692–3698.
- [118] E. W. Weisstein. Frobenius norm. Retrieved on 05/10/14. [Online]. Available: <http://mathworld.wolfram.com/FrobeniusNorm.html>
- [119] E. W. Weisstein. Metric. Retrieved on 25/07/14. [Online]. Available: <http://mathworld.wolfram.com/Metric.html>
- [120] E. W. Weisstein. Secant method. Retrieved on 05/10/14. [Online]. Available: <http://mathworld.wolfram.com/SecantMethod.html>
- [121] E. W. Weisstein. Woodbury formula. Retrieved on 24/08/14. [Online]. Available: <http://mathworld.wolfram.com/WoodburyFormula.html>
- [122] R. Wolcott and R. Eustice, "Visual localization within lidar maps for automated urban driving," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, Sept 2014, pp. 176–183.
- [123] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1330–1334, Nov 2000.
- [124] J. Ziegler, H. Lategahn, M. Schreiber, C. Keller, C. Knoppel, J. Hipp, M. Haueis, and C. Stiller, "Video based localization for bertha," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, June 2014, pp. 1231–1238.

Appendix A

Levenberg-Marquardt Equivalence to Gradient Descent for a Single Residual

Starting from the standard Levenberg-Marquardt update step for a nonlinear least squares problem given in Equation (A.1), consider the case where there is a single scalar residual $r \in \mathbb{R}^1$ and the objective to be minimised is $r^T r = r^2$. The parameter space is defined by $\mathbf{x} \in \mathbb{R}^{N \times 1}$ and thus the Jacobian of the residuals: $J_k \in \mathbb{R}^{1 \times N}$.

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (J_k^T J_k + \lambda_k I)^{-1} J_k^T r_k \quad (\text{A.1})$$

In this case, by definition $\text{rank}(J_k^T J_k) = 1$, however we want to show explicitly that Equation (A.1) is exactly scaled gradient descent for this case. Considering the approximate inverse Hessian term: $(J_k^T J_k + \lambda_k I)^{-1}$, using the Woodbury Formula [121] given in Equation (A.2), this can be written as Equation (A.3).

$$(A + UCV)^{-1} \equiv A^{-1} - A^{-1}U (C^{-1} + VA^{-1}U)^{-1} VA^{-1} \quad (\text{A.2})$$

$$\begin{aligned}
(J_k^T J_k + \lambda_k I)^{-1} &= \frac{1}{\lambda_k} I - \frac{1}{\lambda_k} I J_k^T \left(I + \frac{J_k I J_k^T}{\lambda_k} \right)^{-1} J_k I \frac{1}{\lambda_k} \\
&= \frac{1}{\lambda_k} \left(I - \frac{J_k^T}{\lambda_k} \left(1 + \frac{J_k J_k^T}{\lambda_k} \right)^{-1} J_k \right) \\
&= \frac{1}{\lambda_k} \left(I - \frac{1}{\lambda_k + J_k J_k^T} J_k^T J_k \right)
\end{aligned} \tag{A.3}$$

Now, left multiplying Equation (A.3) by $J_k^T r_k$ from Equation (A.1), results in Equation (A.4) which can be simplified to Equation (A.5), using the fact that $J_k J_k^T$ is a scalar.

$$(J_k^T J_k + \lambda_k I)^{-1} J_k^T r_k = \frac{1}{\lambda_k} \left(I - \frac{1}{\lambda_k + J_k J_k^T} J_k^T J_k \right) J_k^T r_k \tag{A.4}$$

$$\begin{aligned}
&= \frac{r_k}{\lambda_k} \left(J_k^T - \frac{1}{\lambda_k + J_k J_k^T} J_k^T J_k J_k^T \right) \\
&= \frac{r_k}{\lambda_k} J_k^T \left(1 - \frac{1}{\lambda_k + J_k J_k^T} J_k J_k^T \right) \\
&= \left(\frac{r_k}{\lambda_k} \left(1 - \frac{1}{\lambda_k + J_k J_k^T} J_k J_k^T \right) \right) J_k^T
\end{aligned} \tag{A.5}$$

Thus, substituting Equation (A.5) into Equation (A.1), we can write the Levenberg-Marquardt update step for a single residual as Equation (A.6), where ω_k is a scalar, which by definition is exactly scaled gradient descent as stated.

$$\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{x}_k - \left(\frac{r_k}{\lambda_k} \left(1 - \frac{1}{\lambda_k + J_k J_k^T} J_k J_k^T \right) \right) J_k^T \\
&= \mathbf{x}_k - \omega_k J_k^T
\end{aligned} \tag{A.6}$$