Batch Continuous-Time Trajectory Estimation

by

Sean William Anderson

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto

Abstract

Batch Continuous-Time Trajectory Estimation

Sean William Anderson
Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto
2017

As the influence of autonomous mobile robots grows stronger on the lives of humans, so too does the importance of robust and accurate localization (and control). Although motion-estimation techniques utilizing passive cameras have been a core topic in robotic research for decades, we note that this technology is unable to produce reliable results in low-light conditions (which account for roughly half the day). For this reason, sensors that use active illumination, such as lidar, are an attractive alternative. However, techniques borrowed from the fields of *photogrammetry* and *computer vision* have long steered the robotics community towards a *simultaneous localization and mapping* (SLAM) formulation with a discrete-time trajectory model; this is not well suited for scanning-type sensors, such as lidar. In this thesis, we assert that a continuous-time model of the trajectory is a more natural and principled representation for robotic-state estimation. Practical robotic localization problems often involve finding the *smooth* trajectory of a mobile robot. Furthermore, we find that the continuous-time framework lends its abilities quite naturally to high-rate, unsynchronized, and scanning-type sensors. To this end, we propose novel continuous-time trajectory representations (both parametric, using weighted basis functions, and nonparametric, using Gaussian-processes) for robotic state estimation and demonstrate their use in a *batch, continuous-time trajectory estimation* framework. We also present a novel outlier rejection scheme that uses a constant-velocity model to account for motion distortion. The core algorithms are validated using data from a two-axis scanning lidar mounted on a robot, collected over a 1.1 kilometer traversal.

Dedication

To my mother and father – thank you for your unconditional love and support throughout my entire academic career, for inspiring me to pursue higher levels of education, and for pushing me to explore my talents.

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Timothy D. Barfoot. I am forever grateful for his guidance, teachings, and the time that he has invested in my understanding of modern state estimation techniques. I would not be the researcher I am today without Tim's inspirational drive to conduct cutting-edge research and (seemingly) limitless knowledge of mathematics (which he is always gracious enough to share). May he never run out of interesting problems to sink his teeth into, and keen padawans with whom to work.

My decision to delve in mobile robotics is largely due to the great influences I had during my undergraduate studies. I'd like to thank Ryan Gariepy for throwing me into the 'wild west' of computer vision with nothing but an OpenCV textbook, a webcam, and a Husky Unmanned Ground Vehicle (UGV) – my career path would likely be very different without that magical summer at Clearpath Robotics. I'd also like to thank Steven Waslander who prepared a fantastic introductory course to mobile robotics and inspired me to further develop my understanding of localization and navigation techniques. Finally, a thank you to Kirk MacTavish, for countless hours of collaboration developing our first 3D SLAM algorithm with the Microsoft Kinect, and his continued support in graduate school with many hours of white-board and scrap-paper-based discussions.

To my colleagues and friends at the Autonomous Space Robotics Lab (ASRL), it was a pleasure, and I hope we have the chance to work together again in the future. A special thanks to Paul Furgale and Chi Hay Tong who paved the road for this thesis with their pioneering work (and discussions) on continuous-time SLAM. Also, many thanks to Colin McManus, Hang Dong, and the rest of the Autonosys operators that worked on appearance-based lidar and collected the large gravel-pit dataset. To all the others I've had the joy of working along side: Peter, Chris, Jon, Francois, Mike P., Mike W., Kai, Pat, Tyler, Katarina, Braden, Erik, Rehman, Keith, Andrew, and Goran, thank you.

Lastly, I want to thank the National Science and Engineering Research Council (NSERC) of Canada for their financial support through the CGS-M and CGS-D awards.

Contents

Notation

$\mathbb{R}^{M\times N}$	:	The real coordinate (vector) space of $M \times N$ matrices.
a	:	Symbols in this font are real scalars, $a \in \mathbb{R}^1$.
$\mathbf{a}$	:	Symbols in this font are real column vectors, $\mathbf{a} \in \mathbb{R}^N$.
$\mathbf{A}$	:	Symbols in this font are real matrices, $\mathbf{A} \in \mathbb{R}^{M\times N}$.
$\mathbf{1}$	:	The identity matrix, $\mathbf{1} \in \mathbb{R}^{N\times N}$.
$\mathbf{0}$	:	The zero matrix, $\mathbf{0} \in \mathbb{R}^{M\times N}$.
$\underrightarrow{\boldsymbol{\mathcal{F}}}_a$	:	A reference frame in three dimensions.
$\mathbf{p}_a^{c,b}$	:	A vector from point b to point c (denoted by the superscript) and expressed in $\underrightarrow{\boldsymbol{\mathcal{F}}}_a$ (denoted by the subscript).
$\boldsymbol{p}_a^{c,b}$	:	The vector $\mathbf{p}_a^{c,b}$ expressed in homogeneous coordinates.
$SO(3)$	:	The special orthogonal group.
$\mathfrak{so}(3)$	:	The Lie algebra associated with $SO(3)$.
$\mathbf{C}_{b,a}$	:	The 3×3 rotation matrix that transforms vectors from $\underrightarrow{\boldsymbol{\mathcal{F}}}_a$ to $\underrightarrow{\boldsymbol{\mathcal{F}}}_b$: $\mathbf{v}_b^{c,b} = \mathbf{C}_{b,a}\mathbf{v}_a^{c,b}$, $\mathbf{C}_{b,a} \in SO(3)$.
$SE(3)$	:	The special Euclidean group.
$\mathfrak{se}(3)$	:	The Lie algebra associated with $SE(3)$.
$\mathbf{T}_{b,a}$	:	The 4×4 transformation matrix that transforms homogeneous points from $\underrightarrow{\boldsymbol{\mathcal{F}}}_a$ to $\underrightarrow{\boldsymbol{\mathcal{F}}}_b$: $\boldsymbol{p}_b^{c,b} = \mathbf{T}_{b,a}\boldsymbol{p}_a^{c,a}$, $\mathbf{T}_{b,a} \in SE(3)$.
$(\cdot)^\wedge$	:	The overloaded operator that transforms a vector, $\boldsymbol{\phi} \in \mathbb{R}^3$, into a 3×3 (skew-symmetric) member of $\mathfrak{so}(3)$, and a vector, $\boldsymbol{\xi} \in \mathbb{R}^6$, into a 4×4 member of $\mathfrak{se}(3)$.
$(\cdot)^\vee$	:	The inverse operator of $(\cdot)^\wedge$.
$E[\cdot]$	:	The expectation operator.
$p(\mathbf{x})$	:	The *a priori* probability density of $\mathbf{x}$.
$p(\mathbf{x}\|\mathbf{y})$	:	The *posterior* probability density of $\mathbf{x}$, given evidence $\mathbf{y}$.
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	:	A Gaussian probability density with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.
$\mathcal{GP}(\boldsymbol{\mu}(t), \boldsymbol{\Sigma}(t,t'))$	:	A Gaussian process with mean function $\boldsymbol{\mu}(t)$ and covariance function $\boldsymbol{\Sigma}(t,t')$.
$\check{(\cdot)}$	:	An a priori quantity: e.g. $p(\mathbf{x}) = \mathcal{N}(\check{\mathbf{x}}, \check{\mathbf{P}})$.
$\hat{(\cdot)}$	:	A posterior quantity: e.g. $p(\mathbf{x}\|\mathbf{y}) = \mathcal{N}(\hat{\mathbf{x}}, \hat{\mathbf{P}})$.
$\bar{(\cdot)}$	:	An estimated quantity that acts as an operating point, or *best guess*, during nonlinear state estimation.

Chapter 1

Introduction

The dream of having autonomous mobile robots safely traverse complex civilian settings is becoming a reality. At the present time, computing power, sensing capability, and the algorithms to take advantage of them, have peaked at a point where industry has begun to adopt many of the techniques developed by the academic robotics community. The most anticipated result of this 'collaboration' is the arrival of self-driving cars (see Figure 1.1), which are predicted to be only a few years away from commercialization. Several companies have been very public about their development (and testing) of autonomous vehicle technology; most notably, Google Inc. (2015) reports having 48 autonomous vehicles actively self-driving the public roads of Mountain View, California, and Austin, Texas, with a collective 2 million autonomously driven kilometres since 2009. In order to enable this kind of autonomous navigation in a *safe* manner, these vehicles must be able to operate in challenging dynamic situations, all weather and lighting conditions, and a variety of environments (e.g., open plains, tunnels, urban canyons, etc.).

(a) An advertisement for the Central Power and Light Company in 1957. The tag line suggested that travel would one day be made more enjoyable and safe through the use of *electricity*. (Credit: The Victoria Advocate)

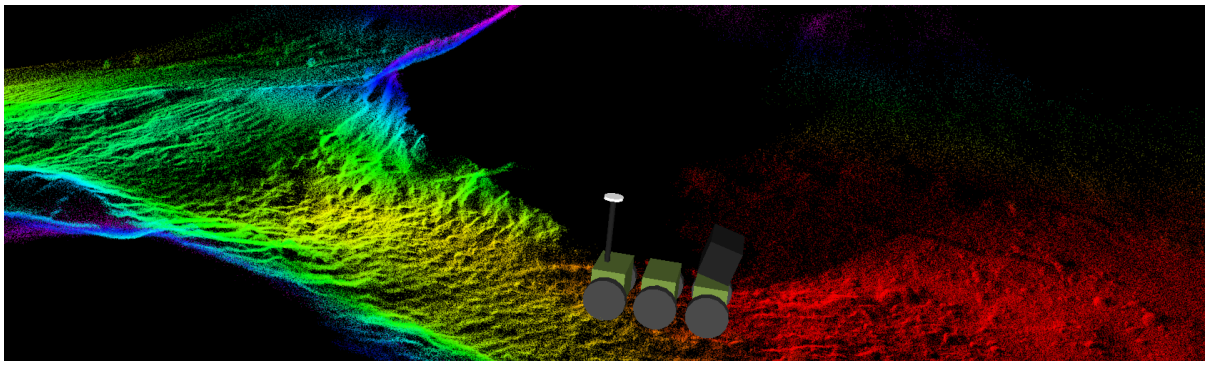
(b) A depiction of Google's autonomous vehicle prototype in 2015. The driverless vehicle uses a combination of lasers, radars, and cameras to safely navigate roads shared with other civilian drivers. (Credit: Google)

Figure 1.1: Self-driving cars.

The focus (and contributions) of this thesis are in the area of mobile robotic localization. Plainly, a robot localization algorithm is responsible for answering the question: *"Where am I?"*. Depending on the intended application, the answer can either be topological (e.g., city → street → lane), or metric (e.g., translational and rotational coordinates with respect to a frame of reference). In modern day life, whether driving or walking to a destination, many citizens have become accustomed to using the Global Positioning System (GPS) on their cellular devices for real-time localization with respect to a map. Similarly, in order to navigate an autonomous robot to a destination, we must first determine the robot's location in the space that our desired path is defined. However, we note that GPS technology (although only available on Earth) is often insufficient for *terrestrial* robot operation; the reliability and availability of a GPS signal is affected by many typical operating environments (e.g., urban canyon, underground/mines, indoors/tunnels, or even forested areas). Furthermore, the installation of infrastructure-based solutions to cover all of the desired areas of robot operation is often prohibitively expensive, and thus we rely on onboard sensing to satisfy our localization needs.

Arguably, the most popular onboard sensing modality for three-dimensional motion estimation is passive camera technology, which has been a core topic in robotics research for decades (Moravec, 1980). In particular, the use of a stereo camera to perform sparse-feature-based *visual odometry* (VO) (Matthies and Shafer, 1987) has remained a leading paradigm that enables accurate pose estimation over long distances (Sibley et al., 2010); both the Mars Exploration Rovers (MERs) (Maimone et al., 2007) and the Mars Science Laboratory (MSL) (Johnson et al., 2008) have used stereo-camera VO for extraterrestrial motion estimation. Notably, without an *a priori* map of visual landmarks, exploratory traverses are forced to localize with respect to a map that is built *during the traversal*. In robotics this paradigm is best known as *simultaneous localization and mapping* (SLAM) and is mathematically posed as a *state estimation* problem; that is, given some observations of the environment, we wish to determine the *state* of the map, which is typically a set of positions associated with the landmarks, and the *full state* of the robot, which is a temporal set of positions, orientations, velocities, and other quantities (e.g., sensor biases or calibration parameters), that fully describe the robot and its sensors throughout the traversal. In contrast to exploratory traverses, we note that autonomous *retrotraverse* can be enabled with separate mapping and localization phases (Furgale and Barfoot, 2010). However, in many cases the most effective way to perform mapping is with SLAM, and furthermore, because environments are susceptible to change, the map is kept up to date

(a) A photograph of the ROC6 traversing typical terrain at the Ethier Sand and Gravel pit. Note the large shadows cast by the robot, and rocks, as the sun begins to set.



(b) A lidar-based reconstruction of the terrain, coloured by elevation. This result uses the appearance-based lidar pipeline described in Chapter 2, and novel algorithms described in Chapters 4 and 6.

Figure 1.2: This figure shows a photograph and rendering of the ROC6 mobile robot at the Ethier Sand and Gravel pit in Sudbury, Ontario, Canada. The large black instrument mounted on the front of the robot is an Autonosys scanning lidar that we use for scanning-while-moving motion estimation.

by performing SLAM during localization as well.

The downfall of vision-based SLAM, using passive camera technology, is that it relies on consistent ambient lighting in order to find temporally similar appearance-based features. In a real-world scenario, such as the test environment shown in Figure 1.2(a), appearance can differ drastically with changes in lighting conditions; for example, change in the orientation of shadows, or more severely, the total absence of light. Although passive cameras have limited use in these severe conditions, the estimation machinery developed to use sparse appearance-based features is principled and time-tested. Therefore, a technology we are interested in investigating is the application of these mature visual techniques to lidar (*li*ght ra*dar*) data, which is more robust to varying lighting conditions. This method was originally investigated by McManus et al. (2011), and a detailed review

of the methodology we follow is described in Chapter 2. In short, due to the scanning nature of the sensor, assumptions made in the traditional VO pipeline become invalid; in particular, the methods used for both outlier rejection and the nonlinear numerical solution require re-evaluation. We overcome these issues by developing new technologies that consider the temporal nature of the data (see Figure 1.2(b)).

In Chapter 3 we review the mathematical principles underlying the probabilistically based SLAM estimation problem; this review leads up to, and includes, the recent introduction of a *batch continuous-time trajectory estimation* framework (Furgale et al., 2012). In Chapter 4 we introduce a novel Random Sample Consensus (RANSAC) algorithm, using a constant-velocity model, to perform outlier rejection on our motion-distorted 3D lidar data (Anderson and Barfoot, 2013a; Anderson et al., 2015b). By modelling the robot trajectory in continuous time, we are able to account for the temporal nature of scanning-type sensors and expose a subtle generalization of the SLAM problem that we refer to as *simultaneous trajectory estimation and mapping* (STEAM). The core contributions of this thesis adopt the STEAM approach and propose new representations for the six-degree-of-freedom robot trajectory. Specifically, in Chapter 5 we present a novel, parametric STEAM algorithm that is able to process loop closures in constant time (Anderson and Barfoot, 2013b; Anderson et al., 2015b); in essence, this work moves the *relative* SLAM formulation (Sibley et al., 2009) into the continuous-time framework by estimating the body-centric velocity profile of the robot (and the spatial pose changes associated with loop closures). In Chapter 6 we explore an alternative STEAM formulation based on Gaussian-process (GP) regression; we build upon the initial work of Tong et al. (2013) and the exactly sparse approach of Barfoot et al. (2014) by introducing: (i) the use of *nonlinear* time-varying (NTV) stochastic differential equations (SDE) to generate exactly sparse GP priors for trajectories in a *vectorspace* (Anderson et al., 2015a), and (ii) a novel, exactly sparse, singularity-free, and physically motivated GP prior for bodies translating and rotating in three-dimensional space (Anderson and Barfoot, 2015). The core contributions from Chapters 4, 5, and 6 are all validated using a 1.1 kilometre lidar dataset collected in Sudbury, Ontario, Canada (details in Appendix A). Finally, a summary of the contributions and discussion of future work are presented in Chapter 7.

Chapter 2

Appearance-Based Lidar Odometry

In this chapter, we review the techniques used in a typical *visual odometry* (VO) pipeline and the theory behind using lidar *intensity images* in lieu of passive camera imagery. Augmentation of the VO pipeline, for lidar intensity data, was originally investigated by McManus et al. (2011, 2012), and later followed up by Dong and Barfoot (2012) and Tong et al. (2014). The methodologies investigated in this thesis build upon these works and are motivated by the problems described in this chapter. In short, the standard VO pipeline is not equipped to deal with scanning-type sensors (such as lidar or a rolling-shutter camera) and requires new technologies to perform outlier rejection and generate a reasonable nonlinear numerical trajectory estimate.

2.1 Motivation

In cases where an *a priori* map of the environment does not exist, or localization to the map has been lost, it is necessary to perform incremental robot localization using sequential pose change estimates (i.e., *odometry*) from the available sensor data. In the most basic sense, odometry can be provided by something as simple as wheel encoders; however, wheel odometry is fairly undependable in real-world environments. For example, Matthies et al. (2007) note that (owing to sand) the MERs had a slip rate of 95% on a 20 degree incline. Although we would typically expect better wheel odometry from a modern automobile on an asphalt road, more reliable sensing is a necessity.

Vision-based methods have proven to provide reliable and accurate odometric estimates (in both position and orientation) over long distances (Konolige et al., 2007; Sibley et al., 2010). These methods operate by identifying and tracking a sparse set of recognizable,

static features across a sequence of images. Camera geometry can then be used to solve for both the 3D positions of the static features (i.e., *landmarks*) and the temporal poses[1] of the camera. This style of optimization problem is described as *bundle adjustment* (BA) and can trace its heritage to the stitching of aerial photography (Brown, 1958). Owing to the restrictions of computing power, early applications of BA to mobile robotics focused on a simplified version of the problem, in which stereo-triangulated points were aligned in a Euclidean frame (Moravec, 1980); this work was later refined (Matthies and Shafer, 1987; Matthies, 1989) and eventually deployed on the MERs (Maimone et al., 2006, 2007) to provide accurate and reliable odometry estimates on Mars.

As described previously, the problem with using passive cameras as the primary sensing modality for a robotic platform is that the provided appearance information is highly dependent on external lighting conditions. The most obvious failure mode of VO is during low-light conditions (or even complete darkness), when tracking features reliably becomes difficult (or impossible). Even during 'daylight' conditions, Matthies et al. (2007) specifically note a VO failure on the MERs when the dominant features were tracking the rover's own shadow. In an attempt to enable dark navigation with a passive stereo camera rig, Husmann and Pedersen (2008) used an LED spotlight to show the promise of High Dynamic Range (HDR) imaging in lunar-analogue conditions; using this method, it was noted that view range is a limiting factor due to inverse-square illumination drop off, and that very specialized camera hardware would be required to enable HDR imaging during continuous locomotion. While vision-based techniques have been widely adopted due to the low cost and availability of passive camera technology, we note that photogrammetry techniques are not restricted to the human-visible light spectrum (this foreshadows the lidar-based technique we review in Section 2.2). An interesting use of the infrared spectrum was presented by Rankin et al. (2007) in their work on negative obstacle detection using a thermal camera.

Active sensors, such as lidar, enable more robust odometry under varying lighting conditions. Furthermore, the wealth of geometric information provided by lidar range measurements have made the sensors very popular for mapping both 2D and 3D environments. In order to align scans taken from different locations, a wide range of laser-based odometry techniques have been developed. For 3D pose estimation, a 3D lidar 'scan' is traditionally constructed by concatenating *swathes* of temporal lidar data into a single *point cloud*.

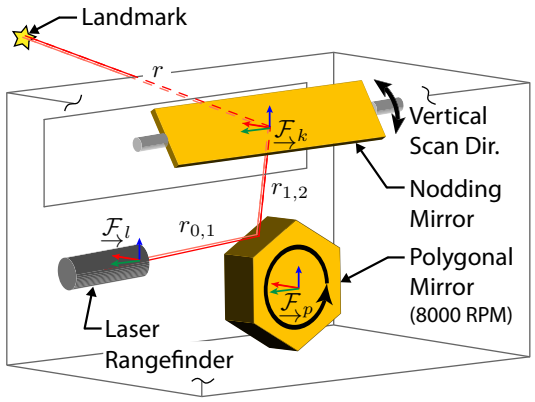
[1] In this thesis, *pose* is used to indicate both a position and orientation.

The basis of many *point cloud* alignment techniques is the Iterative Closest Point (ICP) algorithm (Besl and McKay, 1992), which iteratively minimizes the least-squared Euclidean error between nearest-neighbour points. Although the basic ICP algorithm is fairly naive, it can provide impressive odometry estimates when applied sequentially (Nüchter et al., 2007). There exist a multitude of works based on ICP that typically modify the optimization problem by using additional information extracted from the point cloud, such as surface normals or curvatures; Pomerleau et al. (2015) provide an excellent survey. In general, registration techniques that work with the *full* dense point clouds tend to be computationally intensive, and struggle to run online[2]. An alternative strand of registration techniques focus on compressing a point cloud into a much smaller number of points (or *features*) with a set of associated statistics that retain information about the original structure. The most successful 'compression' techniques in this area of point cloud registration are the Normalized Distributions Transform (NDT) (Magnusson et al., 2007) and surfel (*surf*ace *el*ement) (Zlot and Bosse, 2012) representations; notably, both techniques use a form of discretization on the points (such as voxelization), followed by an eigen decomposition of the second-order statistics within each voxel. Pathak et al. (2010) also achieve online performance with lidar by using a method in which large segments of lidar data are 'compressed' into planar features for fast, closed-form pose-graph-relaxation; notably their technique relies on stop-and-go motion to avoid motion-distortion over the large planar features.

Though the use of geometric information has provided impressive odometric results, it is interesting to note that the secondary lidar data product (i.e., intensity/reflectance information) is largely discarded by most of the robotic literature. Most applications of geometric-based lidar odometry have been in urban areas, or mines, where walls, ridges, and other rich geometric information is typically present; in contrast, a road through an open plain represents a failure mode for most of these algorithms – using intensity information, such as the return from white, dashed lane markings, could prevent this failure. Early work by Neira et al. (1999) investigated the use of intensity and range images for localization against a known map. Prominent work by Levinson (2011) has showed the use of lidar intensity information to enable localization and dark navigation of an autonomous vehicle.

[2]In robotics, *online* performance typically implies that the result of an algorithm can be computed before the next measurement to be processed arrives.

(a) The ROC6 mobile rover, equipped with an Autonosys LVC0702 lidar and a Thales DG-16 Differential GPS unit.



(b) Schematic of the Autonosys LVC0702 (Dong et al., 2013), showing the nodding and hexagonal mirrors used to achieve an image-style scan pattern.

Figure 2.1: The Autonosys LVC0702 is a high-framerate amplitude modulated continuous wave (AMCW) lidar with a pulse repetition rate (PRR) of 500,000 points per second and maximum range of ~50 meters. In our experiments, the Autonosys was configured to have a 90° horizontal and 30° vertical field of view, and produce 480 × 360 images at 2 Hz.

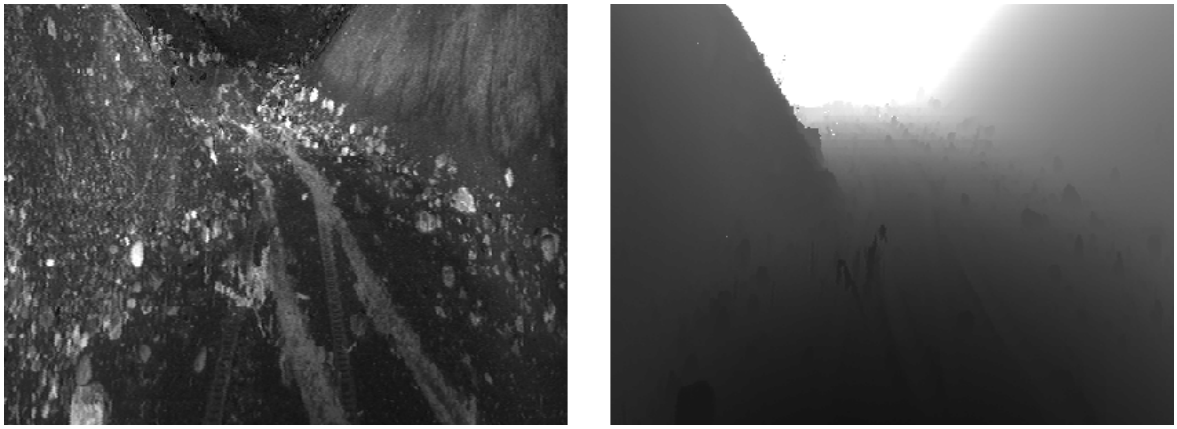


Figure 2.2: This figure depicts corresponding intensity (left) and range (right) images formed using data from the Autonosys lidar and the methodology developed by McManus et al. (2011). In this process, image 'creation' using lidar is best described as pushing *swathes* of temporally sequential lidar scans into a rectilinear image format (where pixel rows and columns roughly correspond to elevation and azimuth angles). In reality, the robot has moved up to 25 cm during the 0.5 s it takes to acquire these 'images'.

2.2 Lidar and the Visual-Odometry Pipeline

Most of the experiments in this thesis leverage a technique, initially explored by McManus et al. (2011), which fuses the intensity information provided by lidar with the mature and computationally efficient, vision-based algorithms. The keystone in merging these

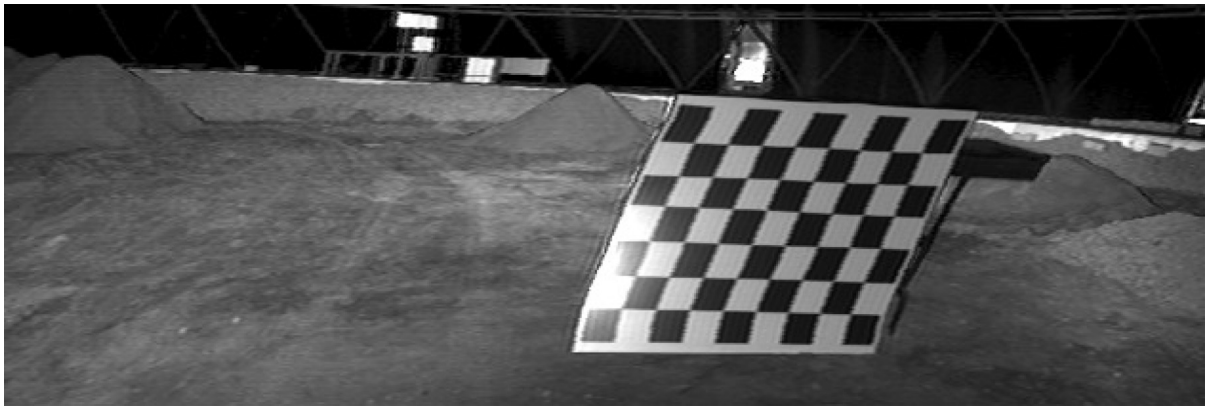


Figure 2.3: This figure shows an example intensity image captured using the Autonosys LVC0702 two-axis scanning lidar. The nonaffine image distortion is caused by a yaw-type motion during image acquisition. Note the irregular deformation of the square checkerboard pattern.

technologies is the use of a two-axis scanning lidar sensor with high-resolution intensity information and a fairly high acquisition rate; specifically, we investigate the use of an Autonosys LVC0702, as seen on our robot in Figure 2.1. We then use the method developed by McManus et al. (2011) to form the lidar intensity and range data into images, as seen in Figure 2.2. Sparse appearance-based features are then extracted from the intensity images and temporally matched. This tactic allows us to perform visual odometry even in complete darkness (Barfoot et al., 2013).

As with any meshing of technologies, there are often new machineries that need to be developed. In this case, the development of vision-based algorithms have long assumed the use of imaging sensors with a global shutter, which are well suited to discrete-time problem formulations. In contrast to a charge-coupled device (CCD) that has a global shutter, the slow vertical scan of the Autonosys causes nonaffine image distortion, as seen in Figure 2.3, based on the velocity of the robot and the capture rate of the sensor. This is similar in nature to rolling-shutter cameras that use complementary metal-oxide-semiconductor (CMOS) technology; these scanning-type sensors are often avoided in robotics due to the added complexity of rolling-shutter-type distortions. In the presence of motion-distorted imagery, there are three steps in which the traditional visual pipeline is affected (shown in Figure 2.4): (i) feature extraction, (ii) outlier rejection, and (iii) the nonlinear numerical solution. In order to adapt these vision-based technologies to use scanning sensors, this thesis proposes novel methods for both outlier rejection and a nonlinear numerical solution.

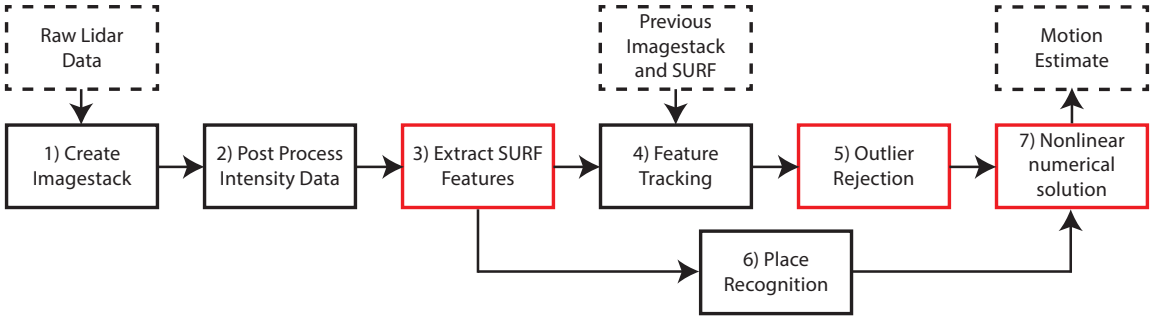


Figure 2.4: This figure depicts the typical data processing pipeline used for visual odometry with lidar intensity images (McManus et al., 2013). Beginning with new data, an imagestack is formed from the intensity, elevation, azimuth, range, and time information. The intensity image is then post processed, SURF features are extracted, and then matched against features from the previous image. Outlier rejection (typically a RANSAC algorithm) is then used to qualify a likely set of inliers from the proposed feature correspondences. In parallel, a place recognition module may be used to identify whether or not we have visited 'this' location previously. Finally, a nonlinear numerical solution is used to produce a motion/localization estimate. Take special note that this pipeline is affected by motion distortion in the three highlighted modules: feature extraction (due to nonaffine image distortion), outlier rejection (due to assumptions made by traditional RANSAC models), and the nonlinear numerical solution (due to the discrete nature of the trajectory representation used by typical pose estimation methods).

2.2.1 Keypoint Detection

The experiments in this thesis closely follow the methodology laid out by McManus et al. (2011) for lidar-based image creation and sparse-feature extraction. In our experiments, the Autonosys was configured to produce 480×360 resolution lidar scans at 2 Hz. These scans are stored as *imagestacks*, which are a collection of 2D images, containing intensity, elevation, azimuth, range, and timestamp data. Each *pixel* in an *imagestack* corresponds to a single, individually timestamped, lidar measurement. Notably, the raw intensity images are post-processed to make them suitable for feature extraction; this is done by applying an adaptive histogram to equalize areas of high and low reflectance, followed by a low-pass Gaussian filter.

After processing the intensity images, keypoints are extracted. The goal of keypoint detection is to find distinctive points in an image that can be robustly tracked as long as they stay in view. Sparse appearance-based feature extraction is a popular strand of research in computer vision. Each feature extraction algorithm typically involves a keypoint detection scheme and a feature descriptor scheme. After the detector has found an interesting keypoint (typically a corner, edge, or blob), a descriptor algorithm

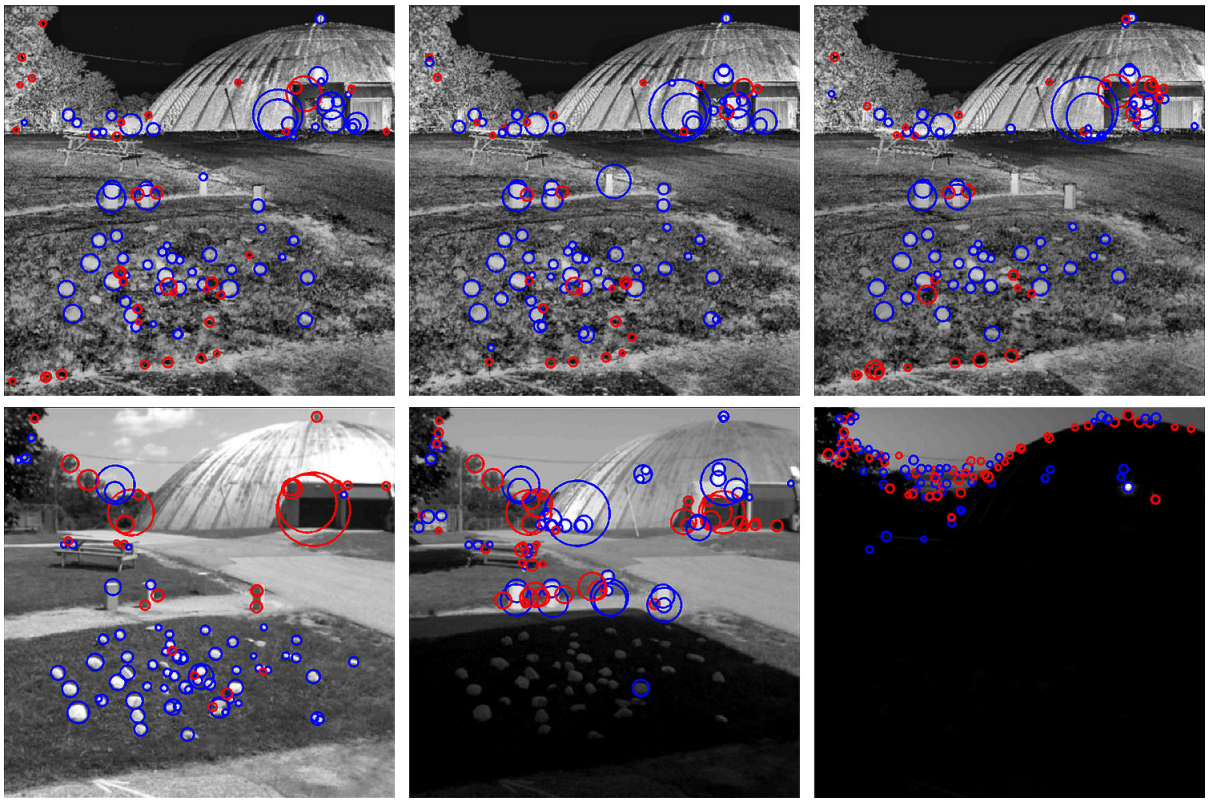


Figure 2.5: Qualitative results from the 24-hour experiment conducted by McManus et al. (2011) comparing intensity images captured from an Optec ILRIS3D survey-grade lidar (top row) against passive camera imagery (bottom row) at three different times (13:38, 18:12, and 05:43). Extracted SURF features are overlaid – blue indicating a light blob on a dark background and red indicating a dark blob on a light background. Note the robustness of SURF detection (in the top row) despite the drastic changes in lighting.

is used to compress the local image patch into a shorter sequence of values that can be quickly compared to other keypoints for similarity. In robotics, popular feature extraction algorithms for VO include Features from Accelerated Segment Test (FAST) (Rosten and Drummond, 2006), Scale Invariant Feature Transforms (SIFT) (Lowe, 2004), Speeded-Up Robust Features (SURF) (Bay et al., 2006), and more recently, several binary schemes, such as Binary Robust Independent Elementary Features (BRIEF) (Calonder et al., 2010), Binary Robust Invariant Scalable Keypoints (BRISK) (Leutenegger et al., 2011), and Oriented FAST and Rotated BRIEF (ORB) (Rublee et al., 2011).

Using a survey-grade 3D lidar, McManus et al. (2011) determined that SURF extracted from intensity imagery could be robustly matched over a 24-hour period, and were far superior to features extracted from passive camera imagery in low-light conditions (see Figure 2.5). SURF detects 'blobs' of interest at a variety of scales, and encodes

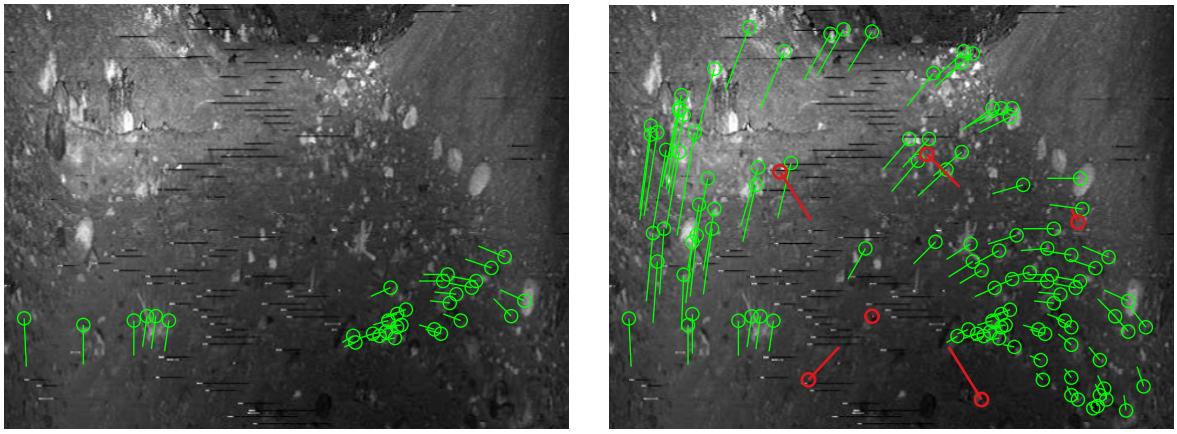
approximations to the local gradient information used by SIFT into a 64-floating-point-number descriptor; it has become a very popular choice since being implemented on the Graphics Processing Unit (GPU), as it can be calculated quickly in parallel, performs well compared to its ancestor, SIFT, and has even been used to enable quick place recognition (Cummins and Newman, 2008). Our experiments leverage a highly parallelized SURF implementation that runs on the GPU.

During continuous movement, intensity imagery is subject to nonaffine motion distortions (due to the temporal nature of lidar technology). It is expected that the performance of typical feature detectors and descriptors (including SURF) will degrade when the sensor acquisition rate is inadequate for the platform speed. However, since features are extracted locally, and the local effect of motion distortion is small, we have found that SURF performs sufficiently well for our particular configuration (a platform speed of up to 0.5 m/s, and an Autonosys frame rate of 2 Hz).

2.2.2 Keypoint Tracking and Outlier Rejection

Given a set of temporally acquired images and the extracted features from each image, we must now determine the feature correspondences (i.e., matches) between the sequential image pairs. This task is typically accomplished by proposing matches based on the keypoint's descriptor similarity, and then performing outlier rejection (usually based on a geometric model) to eliminate mismatches. In robotics, the most popular way to perform outlier rejection is RANSAC (Fischler and Bolles, 1981); although arguments can be made for the benefits of both robust M-estimation (Huber, 1981) and joint-compatibility methods, such as active matching (Chli and Davison, 2008). RANSAC has become popularized as an outlier rejection scheme for VO pipelines because it is fast and suitable for robustly estimating the parameters of a model, despite a large number of outliers.

RANSAC determines the parameters of a model by generating hypotheses and using consensus to let the data determine the most likely one (i.e., the hypothesis with which the majority of the data agrees). In order to generate a hypothesis, a set of data (in this case, matches) are sampled randomly from the proposed set and used to calculate the model; note that the number of samples that must be drawn to compute a hypothesis is dependent on the model that is being solved. The challenge in using RANSAC is that it is not a deterministic algorithm (unless run to exhaustion) and good performance depends on being able to (probabilistically) draw a set of matches that all belong to



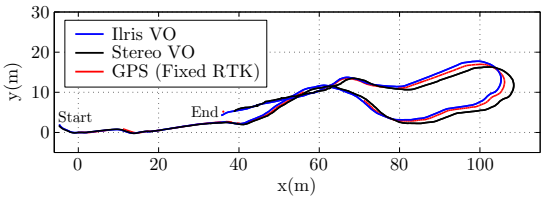
(a) This figure shows inlying feature tracks after using a moderate threshold on reprojection error. Due to fast motion and a slow vertical scan, only a small temporal band of the features are matched.

(b) This figure shows how relaxing the inlier threshold (for a rigid RANSAC model) allows for a larger number of inlying matches (green), but also introduces false positives (red outliers).

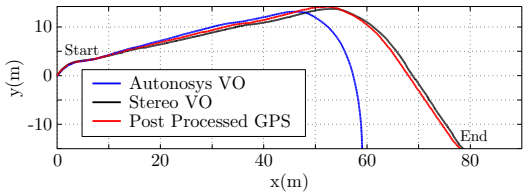
Figure 2.6: Figures showing the inlying feature tracks after applying a rigid 3-point RANSAC model on lidar data captured during fairly fast and rough motion.

the inlier set within a reasonable number of iterations. Computational improvements to RANSAC are usually based on reducing the number of samples required to compute the model, or determining a criterion that helps bias the random sampler toward selecting likely inliers. The most general (monocular) model, based on epipolar geometry and the fundamental matrix, uses 8 points (Longuet-Higgins, 1981; Hartley and Zisserman, 2000). More suitable models for real-time performance are the monocular 5-point algorithm (Nistér, 2004), and stereo-pair, 3-point algorithm (Horn, 1987).

Direct application of the traditional 3-point RANSAC algorithm to features extracted from motion-distorted lidar imagery results in poor outlier rejection, as seen in Figure 2.6; the scanning nature of lidar violates the assumed rigid rotation and translation model used by the algorithm. Early experiments conducted by McManus et al. (2013) used the rigid model and suffered from poor correspondences. Dong and Barfoot (2011) later employed robust M-estimation as an outlier rejection scheme and more recent work by Tong et al. (2014) leveraged our novel RANSAC method (contributed in Chapter 4 of this thesis). The adaptation presented in this thesis is also a 3-point RANSAC algorithm, but uses a *constant-velocity* model that can account for the individual timestamps of features.



(a) In this experiment, a survey grade lidar (Ilris) was used with a stop-scan-go motion scheme to judge the potential of lidar intensity imagery for use in the VO pipeline (McManus et al., 2011). Notably, the lidar-based odometry performs best ~100m from the start of the run.



(b) In this experiment, the Autonosys lidar (running at 2Hz) was used during continuous locomotion to determine how drastic the effect of motion distortion is without proper compensation (McManus et al., 2013). Note the estimate deteriorates quickly after a change in orientation.

Figure 2.7: These figures compare VO estimates generated by using both lidar intensity imagery (blue) and a stereo camera (black). Ground truth was provided by GPS (red).

2.2.3 Nonlinear Numerical Solution

The final step in the VO pipeline is to estimate odometry using the feature tracks that passed outlier rejection. A rudimentary implementation might simply re-solve the model used by RANSAC with all of the inliers (rather than the minimal set) in a least-squares fashion. For example, by taking advantage of depth information, the least-squares 3D-point-alignment problem (used in the 3-point RANSAC method) can also be solved for N-points in closed-form (Horn, 1987; Arun et al., 1987; Umeyama, 1991); owing to its low computational cost and fairly accurate results, this method is ideal for resource constrained systems, such as the MERs (Matthies et al., 2007).

More recently, easy access to powerful computing has begun to favour more computationally intensive solutions based on batch nonlinear optimization, such as *bundle adjustment* (mathematical preliminaries will be provided in Chapter 3). However, owing to the nature of traditional VO and discrete-time SLAM formulations, they are not well-suited to handle many challenging sensor outputs; specifically, the use of high-rate, motion-distorted, or unsynchronized sensors all require *special treatments*. In our case, scanning during continuous locomotion causes each SURF feature extracted from the intensity imagery to be individually timestamped, and in general, traditional discrete-time batch SLAM estimators require a pose at every measurement time. Placing a discrete pose at each SURF measurement time causes two problems: (i) the state size becomes computationally intractable, and (ii) with only one range/bearing measurement at each discrete pose, the *maximum likelihood* (ML) problem is ill-conditioned (i.e., unobservable).

Before developing more complex machinery that allows us to properly handle these

types of sensors, two practical experiments were conducted by McManus et al. (2011). First, in order to justify lidar intensity imagery as a potentially suitable replacement for passive camera imagery, an initial experiment compared lidar odometry (using a stop-scan-go methodology to avoid motion distortion) against stereo VO (see Figure 2.7(a)). Using the same estimation scheme, the second experiment (seen in Figure 2.7(b)) enabled continuous locomotion to demonstrate the need for motion compensation.

Outside of research focused on solving the motion-distortion problem, there are two commonly used strategies to improve odometry estimates from scanning sensors. The first is to avoid the problem entirely by using a stop-scan-go motion strategy, as employed by McManus et al. (2011), Nüchter et al. (2007), and Pathak et al. (2010) (although this strategy has a serious impact on the platform's freedom of motion). The second, is to 'pre-correct' the distortion by using an estimate of the vehicle's velocity – provided by an Inertial Measurement Unit (IMU) or other external sensor. Although this type of correction leads to satisfactory results, we note that reliance on an additional sensor is undesirable; in particular, we note that noise from the IMU measurements, failure to properly estimate IMU biases, and calibration error between the sensors, all become built into the augmented scans and will contribute to irreversible odometry error. We assert that a more general approach is to change how the robot's trajectory is modelled in the state, such that odometry can be estimated using motion-distorted lidar data alone; then, additional information about the motion, such as IMU measurements, can be exploited when, or if, it is available.

In an attempt to augment the discrete-time state formulation for scanning-type sensors, a few works have employed the use of 3D-pose interpolation (Dong and Barfoot, 2011; Hedborg et al., 2012; Bosse et al., 2012). The idea behind these techniques is to maintain a small set of discrete-time *keyframes* that are used to interpolate a smooth 3D trajectory; in practice, this is akin to having a discrete pose at each measurement time, where the chosen interpolation policy enforces a strict smoothness constraint across poses that exist between *keytimes*. The result is a (generally) well-conditioned problem, with a drastically decreased state size. Notably, all three aforementioned works used time as the interpolation variable between poses, giving rise to a reinterpretation of the formulation as a *continuous-time trajectory estimation problem*. Rather than using an ad hoc linear interpolation scheme, it is proposed that richer trajectory representations can be chosen to better capture the true motion of the robot. Early investigations of continuous-time batch estimation have been performed both parametrically, using a weighted sum of temporal

basis functions (Furgale et al., 2012), and nonparametrically, using a Gaussian Process (Tong et al., 2012). The core contributions of this thesis are based on these works; in particular, we explore alternative trajectory representations that aim to improve both the utility and efficiency of these methods. While more specific information about the implementation of these methods is left to the later chapters, we note that it is the use of these continuous-time trajectory formulations that allow us to produce accurate odometry estimates and maps using motion-distorted lidar imagery.

2.3 Place Recognition

In this thesis, we will go beyond the VO-style nonlinear numerical solutions provided by Dong and Barfoot (2011) and Tong et al. (2014) by taking advantage of large-scale loop closures. In essence, by recognizing when the robot has returned to a location that it has previously traversed, the 'loop' can be closed and parameters in the map (i.e., 3D landmark positions) from the same physical location can be associated – this helps to improve the metric accuracy of localization, as well as introduce topological linkages that can be used for navigation. Chapters 5 and 6 each contain a novel algorithm that uses a *relative-pose* formulation to process these large-scale loop closures in constant time (background on the *relative* SLAM paradigm will be discussed in Chapter 3).

Taking advantage of vision algorithms once again, we note that the use of SURF allows us to leverage existing Bag-of-Words place-recognition algorithms, such as FAB-MAP (Cummins and Newman, 2008). However, owing to motion distortion we found that the standard FAB-MAP methodology did not perform well – instead, we used the extension of MacTavish and Barfoot (2014). In contrast to the original FAB-MAP algorithm, which compares a single query image to all of the previously seen images using a Bag-of-Words descriptor, the modification proposed by MacTavish and Barfoot (2014) describes and compares groups of images (i.e., they use a bigger bag of words); this modification was key in enabling more robust place recognition despite motion distortion.

2.4 Summary

In this chapter we reviewed some of the theory and literature related to using lidar intensity imagery, rather than passive camera imagery, in the VO pipeline. The advantage of using an active sensor, such as lidar, is that it is almost completely invariant to

lighting conditions and allows us to operate even in complete darkness. Furthermore, by extracting sparse appearance-based features from the intensity imagery we are able to leverage a variety of mature techniques (e.g., feature tracking, RANSAC, Bag-of-Words place recognition, bundle-adjustment-style estimation). The issue that arises in this fusion of technologies is *motion distortion* and the fact that standard vision-based techniques are not equipped to deal with scanning sensors. Chapters 4, 5, and 6 of this thesis will propose novel solutions for pieces of the VO pipeline that require special care in order to account for the temporal nature of the lidar sensor.

Chapter 3

Batch Estimation Theory and Applications for SLAM

This chapter serves as a mathematical primer on the use of batch estimation for SLAM. The core contributions of this thesis depend on many of the techniques described in this chapter. The following sections will introduce: (i) probabilistic theory geared towards the batch SLAM formulation, (ii) nonlinear optimization using the Gauss-Newton algorithm, (iii) estimation machinery for the matrix Lie groups $SO(3)$ and $SE(3)$, (iv) variants of the batch SLAM problem aimed at achieving constant-time performance, and (v) preliminaries towards estimating a continuous-time robot trajectory.

3.1 Discrete-Time Batch SLAM

Accurately determining a robot's position with respect to a set of obstacles, landmarks, or path of interest (i.e., a map), is a precursor to autonomous navigation and control of a mobile robotic system. In contrast to pure localization (against an *a priori* map), simultaneous mapping is absolutely vital in traversing previously unvisited environments. Since state-of-the-art robotic systems have set their sights on enabling long-term autonomy in unstructured 3D environments, the importance of robust SLAM technology has only grown. Even when an *a priori* map exists, online mapping remains an integral component of long-term localization engines due to the possibility of vast scene change (e.g., weather, lighting, or even infrastructure). Recording new 'experiences' for *life-long* SLAM has enabled some of the most promising results (Churchill and Newman, 2013).

From a mathematical perspective, modern SLAM algorithms continue to leverage

a probabilistic foundation, as it provides a principled and successful model for fusing measurements from multiple sensors and incorporating *a priori* knowledge of the state. The standard state representation for this type of problem continues to be the discrete-time pose (and sometimes velocity) of the robot, in addition to a set of discrete landmark positions. The origin of this formulation in the robotics community can be traced back to the work of Smith et al. (1990), which set the stage for 2D probabilistic SLAM algorithms to use filtering-based estimation theory (Kalman, 1960). Only much later did Lu and Milios (1997) derive the full *batch* problem formulation, using odometry measurements to smooth the trajectory between landmark observations. Notably, the discrete-time batch SLAM problem is closely related to that of the (much earlier) *bundle adjustment* problem (Brown, 1958); the unique nature of the SLAM problem is that we are estimating the temporal pose of a single rigid body (in contrast to the individual poses of an unordered set of aerial cameras). Exploiting the temporal nature of the measurements, a *specialized* problem is formed by including an *a priori* motion model and measurements from other sensors, such as wheel encoders, compass, IMU, or even lidar.

Despite the batch formulation offering a more accurate solution, filter-based algorithms, such as the Extended Kalman Filter (EKF) (Kalman, 1960), Sigma-Point Kalman Filter (Julier and Uhlmann, 1997), and the Particle Filter (Thrun et al., 2001), were widely used due to their lower computational requirements and ability to run *online*. However, the advancement of modern computing power has slowly favoured batch estimation techniques that were previously too inefficient for practical use. Today, the most successful SLAM solutions have taken advantage of the batch problem formulation (Thrun and Montemerlo, 2006; Dellaert and Kaess, 2006; Kaess et al., 2008; Sibley et al., 2009; Konolige et al., 2010), and modern implementations have proven to provide higher accuracy solutions (per unit of computation) over the filtering-based competitors (Strasdat et al., 2010).

3.1.1 Probabilistic Formulation

In the standard discrete-time batch SLAM formulation, there are two sets of quantities we are interested in estimating: (i) the pose of the robot at all measurement times, $\mathbf{x}_{0:K}$, and (ii) a set of static landmarks parameters, $\boldsymbol{\ell}_{0:L}$. In order to determine the values of $\mathbf{x}_{0:K}$ and $\boldsymbol{\ell}_{0:L}$, we typically use two sources of information. The first is *a priori* information, based on our initial knowledge of the robot's position, $\mathbf{x}_0$, and the (known) control inputs,

$\mathbf{u}_{0:K-1}$, combined with a motion model:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k, \tag{3.1}$$

where $\mathbf{x}_k$ is the pose of the robot at time t_k, $\mathbf{f}(\cdot)$ is a nonlinear function, $\mathbf{u}_k$ is the discretized control input, and $\mathbf{w}_k$ is the process noise. In order to keep this derivation relatively straightforward we have assumed that our motion model has an additive process noise. In general, $\mathbf{w}_k$ could also be an input parameter of the nonlinear function $\mathbf{f}(\cdot)$, as we will later explore in Chapter 6 (for continuous-time models).

The second piece of information (used to refine the estimate), is an observation model, which correlates our series of poses, $\mathbf{x}_{0:K}$, through measurements of common static landmark parameters, $\boldsymbol{\ell}_j$,

$$\mathbf{y}_{kj} = \mathbf{g}(\mathbf{x}_k, \boldsymbol{\ell}_j) + \mathbf{n}_{kj}, \tag{3.2}$$

where $\mathbf{y}_{kj}$ is a sensor measurement, $\mathbf{g}(\cdot)$ is a nonlinear measurement model, $\boldsymbol{\ell}_j$ is an observed point-landmark, and $\mathbf{n}_{kj}$ is the sensor noise. In this model, we have again assumed that the noise, $\mathbf{n}_{kj}$, is additive; however, we note that for observation models this is a much more common/fair assumption.

Taking the probabilistic approach to discrete-time batch SLAM, the *maximum a posteriori* (MAP) problem we wish to solve is

$$\{\hat{\mathbf{x}}, \hat{\boldsymbol{\ell}}\} = \underset{\mathbf{x},\boldsymbol{\ell}}{\operatorname{argmax}} \; p(\mathbf{x}, \boldsymbol{\ell} \,|\, \mathbf{u}, \mathbf{y}), \tag{3.3}$$

where (for convenience) we have defined

$$\mathbf{x} := (\mathbf{x}_0, \ldots, \mathbf{x}_K), \quad \boldsymbol{\ell} := (\boldsymbol{\ell}_0, \ldots, \boldsymbol{\ell}_L), \quad \mathbf{u} := (\check{\mathbf{x}}_0, \mathbf{u}_0, \ldots, \mathbf{u}_{K-1}), \quad \mathbf{y} := (\mathbf{y}_{00}, \ldots, \mathbf{y}_{KL}),$$

and $\{\hat{\mathbf{x}}, \hat{\boldsymbol{\ell}}\}$ is the posterior value of $\{\mathbf{x}, \boldsymbol{\ell}\}$. An equivalent solution to (3.3) can be found by minimizing the negative log likelihood:

$$\{\hat{\mathbf{x}}, \hat{\boldsymbol{\ell}}\} = \underset{\mathbf{x},\boldsymbol{\ell}}{\operatorname{argmin}} \; (-\ln p(\mathbf{x}, \boldsymbol{\ell} \,|\, \mathbf{u}, \mathbf{y})). \tag{3.4}$$

By assuming that the noise variables, $\mathbf{w}_k$ and $\mathbf{n}_{kj}$, for $k = 0 \ldots K$, are uncorrelated, we follow the standard Bayes' rule derivation[1] to rewrite the posterior probability density as,

$$\ln p(\mathbf{x}, \boldsymbol{\ell} \,|\, \mathbf{u}, \mathbf{y}) = \ln p(\mathbf{x}_0 \,|\, \check{\mathbf{x}}_0) + \sum_k \ln p(\mathbf{x}_{k+1} \,|\, \mathbf{x}_k, \mathbf{u}_k) + \sum_{kj} \ln p(\mathbf{y}_{kj} \,|\, \mathbf{x}_k, \boldsymbol{\ell}_j). \tag{3.5}$$

[1]The details of the probabilistic SLAM derivation are fairly common knowledge among roboticists. However, for more details, as well as interesting discussions and demonstrations, I highly recommend both Barfoot (2016) and Thrun et al. (2005).

Next, we assume that the above probability densities are Gaussian by setting

$$\mathbf{x}_0 \sim \mathcal{N}(\check{\mathbf{x}}_0, \check{\mathbf{P}}_0), \qquad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k), \qquad \mathbf{n}_{kj} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{kj}), \tag{3.6}$$

where $\check{\mathbf{x}}_0$ and $\check{\mathbf{P}}_0$ are the prior mean and covariance of the initial position, and the process noise, $\mathbf{w}_k$, and measurement noise, $\mathbf{n}_{kj}$, are normally distributed with covariances $\mathbf{Q}_k$ and $\mathbf{R}_{kj}$, respectively. Finally, by using the models in (3.1) and (3.2), and substituting the Gaussian distributions into (3.5), and subsequently (3.4), we arrive at the typical least-squares batch optimization problem:

$$\{\hat{\mathbf{x}}, \hat{\boldsymbol{\ell}}\} = \underset{\mathbf{x},\boldsymbol{\ell}}{\operatorname{argmin}} \big(J_p(\mathbf{x}) + J_m(\mathbf{x}, \boldsymbol{\ell}) + \text{const.} \big), \tag{3.7}$$

where $J_p(\mathbf{x})$ is a sum of Mahalanobis distances (i.e., squared-error terms) related to the *a priori* data,

$$J_p(\mathbf{x}) := \frac{1}{2} \mathbf{e}_{p_0}^T \check{\mathbf{P}}_0^{-1} \mathbf{e}_{p_0} + \frac{1}{2} \sum_k \mathbf{e}_{u_k}^T \mathbf{Q}_k^{-1} \mathbf{e}_{u_k}, \tag{3.8}$$

and similarly, $J_m(\mathbf{x}, \boldsymbol{\ell})$ is a sum of Mahalanobis distances related to the observations,

$$J_m(\mathbf{x}, \boldsymbol{\ell}) := \frac{1}{2} \sum_{kj} \mathbf{e}_{m_{kj}}^T \mathbf{R}_{kj}^{-1} \mathbf{e}_{m_{kj}}, \tag{3.9}$$

where the *error terms* are:

$$\mathbf{e}_{p_0} := \mathbf{x}_0 - \check{\mathbf{x}}_0, \tag{3.10a}$$

$$\mathbf{e}_{u_k} := \mathbf{x}_{k+1} - \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \tag{3.10b}$$

$$\mathbf{e}_{m_{kj}} := \mathbf{y}_{kj} - \mathbf{g}(\mathbf{x}_k, \boldsymbol{\ell}_j). \tag{3.10c}$$

Defining the *objective function*,

$$J(\mathbf{x}, \boldsymbol{\ell}) := J_p(\mathbf{x}) + J_m(\mathbf{x}, \boldsymbol{\ell}), \tag{3.11}$$

the final MAP estimator is simply

$$\{\hat{\mathbf{x}}, \hat{\boldsymbol{\ell}}\} = \underset{\mathbf{x},\boldsymbol{\ell}}{\operatorname{argmin}} \; J(\mathbf{x}, \boldsymbol{\ell}) \; . \tag{3.12}$$

By solving this nonlinear, least-squares problem, we obtain values for $\hat{\mathbf{x}}$ and $\hat{\boldsymbol{\ell}}$ that maximize the joint-likelihood of our data and *a priori* information. In the absence of a *prior* (i.e., the prior is a uniform distribution over all possible robot states), this

formulation is equivalent to the well understood *maximum likelihood* (ML) problem. Over the past decade, an exploding number of works have built upon this probabilistic batch estimation scheme to solve a variety of problems. Typical extensions include (but are not limited to): (i) penalty, constraint, and conditioning terms that directly affect the objective function, (ii) optimization strategies to more efficiently, or robustly, find the optimal state, and (iii) new parameterizations for the robot or landmark states to improve utility or performance. The remainder of this chapter will review existing tools and extensions of the batch framework used by the contributions of this thesis.

3.1.2 Gauss-Newton Algorithm

In this thesis, we take the Gauss-Newton approach to solving the *unconstrained* nonlinear optimization problem presented in (3.12). Gauss-Newton is an iterative scheme that approximates Newton's method by ignoring the second-order-derivative term in the Hessian matrix. Defining the joint state vector,

$$\mathbf{z} := \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\ell} \end{bmatrix}, \tag{3.13}$$

and the following block quantities,

$$\mathbf{e}_p = \begin{bmatrix} \mathbf{e}_{p_0} \\ \mathbf{e}_{u_0} \\ \vdots \\ \mathbf{e}_{u_{K-1}} \end{bmatrix}, \quad \mathbf{e}_m = \begin{bmatrix} \mathbf{e}_{m_{00}} \\ \vdots \\ \mathbf{e}_{m_{KL}} \end{bmatrix}, \tag{3.14}$$

$$\mathbf{Q} = \text{diag}(\check{\mathbf{P}}_0, \mathbf{Q}_0, \ldots, \mathbf{Q}_{K-1}), \quad \mathbf{R} = \text{diag}(\mathbf{R}_{00}, \ldots, \mathbf{R}_{KL}),$$

we can rewrite the objective function, $J(\mathbf{x}, \boldsymbol{\ell})$, in matrix form as

$$J(\mathbf{z}) = \underbrace{\frac{1}{2}\mathbf{e}_p^T \mathbf{Q}^{-1} \mathbf{e}_p}_{\text{prior}} + \underbrace{\frac{1}{2}\mathbf{e}_m^T \mathbf{R}^{-1} \mathbf{e}_m}_{\text{measurements}}, \tag{3.15}$$

which is quadratic, but not in our state vector, $\mathbf{z}$. In order to linearize the error terms, we decompose the (desired) posterior state, $\hat{\mathbf{z}}$, into two components: (i) an operating point (i.e., best guess of the posterior state), $\bar{\mathbf{z}}$, and (ii) an unknown state perturbation, $\delta\mathbf{z}$. Since the state belongs to a *vectorspace*, we assume that the perturbation is additive:

$$\hat{\mathbf{z}} = \bar{\mathbf{z}} + \delta\mathbf{z}. \tag{3.16}$$

Therefore, by solving the intermediate problem,

$$\delta\mathbf{z}^\star = \underset{\delta\mathbf{z}}{\operatorname{argmin}}\ J(\bar{\mathbf{z}} + \delta\mathbf{z}) \,, \tag{3.17}$$

the optimal state perturbation, $\delta\mathbf{z}^\star$, can be used to bring our best guess, $\bar{\mathbf{z}}$, closer to the optimal posterior value, $\hat{\mathbf{z}}$. In Newton's method, $J(\mathbf{z})$ is approximated as quadratic by using a three-term Taylor-series expansion:

$$J(\bar{\mathbf{z}} + \delta\mathbf{z}) \approx J(\bar{\mathbf{z}}) + \Bigg(\underbrace{\frac{\partial J(\mathbf{z})}{\partial \mathbf{z}}\bigg|_{\bar{\mathbf{z}}}}_{\text{Jacobian}}\Bigg)\delta\mathbf{z} + \frac{1}{2}\delta\mathbf{z}^T\Bigg(\underbrace{\frac{\partial^2 J(\mathbf{z})}{\partial \mathbf{z}\partial \mathbf{z}^T}\bigg|_{\bar{\mathbf{z}}}}_{\text{Hessian}}\Bigg)\delta\mathbf{z}. \tag{3.18}$$

In practice, Newton's method is seldom used for multivariate problems since the second-order-derivative terms in the Hessian can be difficult to compute. The Gauss-Newton method approximates Newton's method by simply ignoring the second-order derivatives in the Hessian. Notably, an equivalent approximation is to simply start with a first-order Taylor-series expansion of the error terms. Following this alternative derivation, the nonlinear error terms in (3.10) are linearized by substituting the assumption from (3.16),

$$\mathbf{e}_{p_0}(\bar{\mathbf{z}} + \delta\mathbf{z}) \approx \mathbf{e}_{p_0}(\bar{\mathbf{z}}) + \delta\mathbf{x}_0, \qquad \mathbf{e}_{p_0}(\bar{\mathbf{z}}) = \bar{\mathbf{x}}_0 - \check{\mathbf{x}}_0, \tag{3.19a}$$

$$\mathbf{e}_{u_k}(\bar{\mathbf{z}} + \delta\mathbf{z}) \approx \mathbf{e}_{u_k}(\bar{\mathbf{z}}) + \delta\mathbf{x}_{k+1} - \mathbf{F}_k\,\delta\mathbf{x}_k, \qquad \mathbf{e}_{u_k}(\bar{\mathbf{z}}) = \bar{\mathbf{x}}_{k+1} - \mathbf{f}(\bar{\mathbf{x}}_k, \mathbf{u}_k), \tag{3.19b}$$

$$\mathbf{e}_{m_{kj}}(\bar{\mathbf{z}} + \delta\mathbf{z}) \approx \mathbf{e}_{m_{kj}}(\bar{\mathbf{z}}) - \mathbf{G}_{\mathbf{x},kj}\,\delta\mathbf{x}_k - \mathbf{G}_{\boldsymbol{\ell},kj}\,\delta\boldsymbol{\ell}_j, \qquad \mathbf{e}_{m_{kj}}(\bar{\mathbf{z}}) = \mathbf{y}_{kj} - \mathbf{g}(\bar{\mathbf{x}}_k, \bar{\boldsymbol{\ell}}_j), \tag{3.19c}$$

where $\mathbf{F}_k$, $\mathbf{G}_{\mathbf{x},kj}$, and $\mathbf{G}_{\boldsymbol{\ell},kj}$ are the *Jacobian* matrices,

$$\mathbf{F}_k := \left.\frac{\partial \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{x}_k}\right|_{\bar{\mathbf{x}}_k, \mathbf{u}_k}, \quad \mathbf{G}_{\mathbf{x},kj} := \left.\frac{\partial \mathbf{g}(\mathbf{x}_k, \boldsymbol{\ell}_j)}{\partial \mathbf{x}_k}\right|_{\bar{\mathbf{x}}_k, \bar{\boldsymbol{\ell}}_j}, \quad \mathbf{G}_{\boldsymbol{\ell},kj} := \left.\frac{\partial \mathbf{g}(\mathbf{x}_k, \boldsymbol{\ell}_j)}{\partial \boldsymbol{\ell}_j}\right|_{\bar{\mathbf{x}}_k, \bar{\boldsymbol{\ell}}_j}. \tag{3.20}$$

Rearranging these quantities into matrix form, we write

$$\mathbf{e}_p \approx \bar{\mathbf{e}}_p - \mathbf{F}\,\delta\mathbf{z}, \quad \mathbf{e}_m \approx \bar{\mathbf{e}}_m - \mathbf{G}\,\delta\mathbf{z}, \tag{3.21}$$

where

$$\bar{\mathbf{e}}_p = \mathbf{e}_p|_{\bar{\mathbf{z}}}\,, \quad \bar{\mathbf{e}}_m = \mathbf{e}_m|_{\bar{\mathbf{z}}}\,, \quad \mathbf{F} = \left.\frac{\partial \mathbf{e}_p}{\partial \mathbf{z}}\right|_{\bar{\mathbf{z}}}, \quad \mathbf{G} = \left.\frac{\partial \mathbf{e}_m}{\partial \mathbf{z}}\right|_{\bar{\mathbf{z}}}, \tag{3.22}$$

and the objective function becomes,

$$J(\mathbf{z} + \delta\mathbf{z}) = \frac{1}{2}(\bar{\mathbf{e}}_p - \mathbf{F}\delta\mathbf{z})^T\mathbf{Q}^{-1}(\bar{\mathbf{e}}_p - \mathbf{F}\delta\mathbf{z}) + \frac{1}{2}(\bar{\mathbf{e}}_m - \mathbf{G}\delta\mathbf{z})^T\mathbf{R}^{-1}(\bar{\mathbf{e}}_m - \mathbf{G}\delta\mathbf{z}). \tag{3.23}$$

Taking the derivative of $J(\mathbf{z})$, with respect to $\delta\mathbf{z}$, and setting it to zero,

$$\frac{\partial^T J(\mathbf{z})}{\partial \delta\mathbf{z}} = -\mathbf{F}^T\mathbf{Q}^{-1}(\bar{\mathbf{e}}_p - \mathbf{F}\delta\mathbf{z}) - \mathbf{G}^T\mathbf{R}^{-1}(\bar{\mathbf{e}}_m - \mathbf{G}\delta\mathbf{z}) = \mathbf{0}, \tag{3.24}$$

this expression can be rearranged to find the system of linear equations

$$\underbrace{\left(\overbrace{\mathbf{F}^T\mathbf{Q}^{-1}\mathbf{F}}^{\mathbf{A}_{\text{pri}}} + \overbrace{\mathbf{G}^T\mathbf{R}^{-1}\mathbf{G}}^{\mathbf{A}_{\text{meas}}}\right)}_{\mathbf{A}} \delta\mathbf{z}^\star = \underbrace{\mathbf{F}^T\mathbf{Q}^{-1}\bar{\mathbf{e}}_p + \mathbf{G}^T\mathbf{R}^{-1}\bar{\mathbf{e}}_m}_{\mathbf{b}}, \tag{3.25}$$

which we can solve for the optimal perturbation, $\delta\mathbf{z}^\star = \mathbf{A}^{-1}\mathbf{b}$. In an iterative fashion, we then update our best guess, $\bar{\mathbf{z}} \leftarrow \bar{\mathbf{z}} + \delta\mathbf{z}^\star$, to convergence, and set the final posterior estimate, $\hat{\mathbf{z}} = \bar{\mathbf{z}}$. Furthermore, we note that the covariance is simply

$$\text{cov}(\delta\mathbf{z}^\star, \delta\mathbf{z}^\star) = \mathbf{A}^{-1}. \tag{3.26}$$

In order to improve the estimator's convergence and avoid local minima, several strategies exist to increase the *robustness* of this nonlinear least-squares solution. In particular, we make use of robust *M-estimation* (Huber, 1981) and trust-region solvers, such as Levenberg-Marquardt (Levenberg, 1944; Marquardt, 1963) and Powell's Dogleg (Powell, 1970). Increasing robustness remains an active interest in the robotics community due its importance in online systems; more recent works, such as Sünderhauf and Protzel (2012) have focused on generalizing outlier rejection within the batch optimization framework. To aid the following discussion on sparsity, the left- and right-hand side terms of (3.25) have been labelled; we note that the $\mathbf{A}_{\text{pri}}$ term comes from the *a priori* information, and $\mathbf{A}_{\text{meas}}$ is associated with the landmark measurements.

3.1.3 Exploiting Sparsity

In general, the naive complexity of solving the system $\delta\mathbf{z}^\star = \mathbf{A}^{-1}\mathbf{b}$, with K measurement times and L landmarks, is $O((K+L)^3)$. Rewriting the linear system of equations, $\mathbf{A}\,\delta\mathbf{z}^\star = \mathbf{b}$, in the 2×2 block-form,

$$\begin{bmatrix} \mathbf{A}_{xx} & \mathbf{A}_{x\ell} \\ \mathbf{A}_{x\ell}^T & \mathbf{A}_{\ell\ell} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}^\star \\ \delta\boldsymbol{\ell}^\star \end{bmatrix} = \begin{bmatrix} \mathbf{b}_x \\ \mathbf{b}_\ell \end{bmatrix}, \tag{3.27}$$

we note that the left-hand side has the form,

$$\begin{bmatrix} \mathbf{A}_{xx} & \mathbf{A}_{x\ell} \\ \mathbf{A}_{x\ell}^T & \mathbf{A}_{\ell\ell} \end{bmatrix} = \overbrace{\begin{bmatrix} \mathbf{Y} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}^{\mathbf{A}_{\text{pri}}} + \overbrace{\begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V} \end{bmatrix}}^{\mathbf{A}_{\text{meas}}}, \tag{3.28}$$

where $\mathbf{Y}$ is a block-tridiagonal matrix related to the prior and $\mathbf{A}_{\text{meas}}$ is the usual *arrowhead* matrix (Brown, 1958), where $\mathbf{U}$ and $\mathbf{V}$ are block-diagonal, and $\mathbf{W}$ is (potentially) dense.

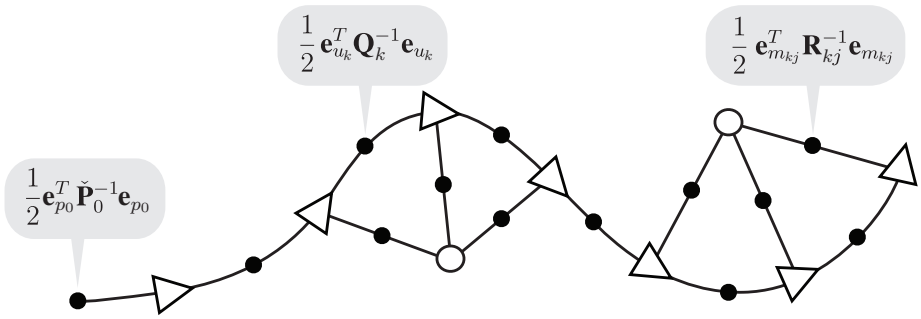


Figure 3.1: The typical SLAM problem we consider, illustrated as a factor graph. The trajectory states, $\mathbf{x}_k$, are represented as triangles and the landmark parameters, $\boldsymbol{\ell}_j$, are represented by hollow circles. The factors (a.k.a. squared-error cost terms) are represented as black dots.

A number of performance-improving techniques have been developed by explicitly exploiting the sparsity of *bundle adjustment* problem structure; the estimators presented in this thesis will primarily leverage the Schur complement and Cholesky decomposition techniques. Alternatively, we note that factor-graph-based solutions exist (Kaess et al., 2012); however, for the scope of this thesis, factor graphs are used simply as a tool to visually represent the structure of a problem, as seen in Figure 3.1.

Schur Complement

The standard *sparse bundle adjustment* (SBA) algorithm uses the Schur complement to exploit the sparse structure of $\mathbf{A}_{\ell\ell}$ by pre-multiplying both sides of (3.27) by

$$\begin{bmatrix} \mathbf{1} & -\mathbf{A}_{x\ell}\mathbf{A}_{\ell\ell}^{-1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \tag{3.29}$$

which results in,

$$\begin{bmatrix} \mathbf{A}_{xx} - \mathbf{A}_{x\ell}\mathbf{A}_{\ell\ell}^{-1}\mathbf{A}_{x\ell}^T & \mathbf{0} \\ \mathbf{A}_{x\ell}^T & \mathbf{A}_{\ell\ell} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}^\star \\ \delta\boldsymbol{\ell}^\star \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 - \mathbf{A}_{x\ell}\mathbf{A}_{\ell\ell}^{-1}\mathbf{b}_2 \\ \mathbf{b}_2 \end{bmatrix}. \tag{3.30}$$

Using the system above, it is straightforward to solve for $\delta\mathbf{x}^\star$ while exploiting the block-diagonal sparsity of $\mathbf{A}_{\ell\ell}$ to find $\mathbf{A}_{\ell\ell}^{-1}$ efficiently. Back-substituting the solution for $\delta\mathbf{x}^\star$, we can then quickly calculate the landmark updates, $\delta\boldsymbol{\ell}^\star$. Assuming that the top-left corner, $\mathbf{A}_{xx} - \mathbf{A}_{x\ell}\mathbf{A}_{\ell\ell}^{-1}\mathbf{A}_{x\ell}^T$, is dense, the complexity is reduced from $O((K+L)^3)$ to $O(K^3 + K^2L)$. This complexity can be improved further by using a sparse-matrix solver to exploit any 'secondary sparsity' that remains in $\mathbf{A}_{xx} - \mathbf{A}_{x\ell}\mathbf{A}_{\ell\ell}^{-1}\mathbf{A}_{x\ell}^T$ (Konolige, 2010).

Sparse Cholesky Decomposition

In Chapter 6, due to the problem formulation and types of experiments, a reverse situation occurs where we potentially have many more trajectory variables than landmark variables, $L \ll K$, and so we instead wish to exploit the sparsity of $\mathbf{A}_{xx}$. This is a more complicated situation, as $\mathbf{A}_{xx}$ is block-tridiagonal, rather than block-diagonal. In this situation, sparse (lower-upper) Cholesky decomposition offers an efficient solution,

$$\underbrace{\begin{bmatrix} \mathbf{V}_{xx} & \mathbf{0} \\ \mathbf{V}_{\ell x} & \mathbf{V}_{\ell\ell} \end{bmatrix}}_{\mathbf{V}} \underbrace{\begin{bmatrix} \mathbf{V}_{xx}^T & \mathbf{V}_{\ell x}^T \\ \mathbf{0} & \mathbf{V}_{\ell\ell}^T \end{bmatrix}}_{\mathbf{V}^T} = \underbrace{\begin{bmatrix} \mathbf{A}_{xx} & \mathbf{A}_{\ell x}^T \\ \mathbf{A}_{\ell x} & \mathbf{A}_{\ell\ell} \end{bmatrix}}_{\mathbf{A}}, \tag{3.31}$$

in which we are able to decompose $\mathbf{V}_{xx}\mathbf{V}_{xx}^T = \mathbf{A}_{xx}$ in $O(K)$ time; this results in $\mathbf{V}_{xx}$ being a lower-bidiagonal matrix. Sparing the details, the *decomposition phase* is performed in $O(L^3 + L^2K)$ time. Performing the standard forward-backward passes,

$$\begin{aligned} \text{solve for } \mathbf{d}\text{:} \quad & \mathbf{V}\,\mathbf{d} = \mathbf{b}, \\ \text{solve for } \delta\mathbf{z}^\star\text{:} \quad & \mathbf{V}^T\delta\mathbf{z}^\star = \mathbf{d}, \end{aligned}$$

where $\mathbf{d}$ is an intermediate variable, the system is then solved in $O(L^2 + LK)$ time. Therefore, the total complexity is dominated by the decomposition, which is $O(L^3 + L^2K)$. Notably, this method avoids any direct matrix inversions (which can ruin sparsity and sometimes cause numerical instability).

3.2 State Estimation Using Matrix Lie Groups

Thus far we have presented the estimation theory and mathematics for solving batch nonlinear optimization problems with state variables that belong to a *vectorspace* (i.e., $\mathbf{z} \in \mathbb{R}^N$). However, in robotics it is common to have state variables that describe the orientation, or pose, of the robot in three-dimensional space. The issue with using rotations and transformations in our probabilistic derivation is that they do not belong to a *vectorspace*, but rather to the (noncommutative) matrix Lie groups: the *special orthogonal group*, $SO(3)$, and the *special Euclidean group*, $SE(3)$. Specifically, without special care, variables that exist on *manifolds* cannot be directly included in our *unconstrained* Gauss-Newton estimator because they violate two of the tools used in our MAP derivation: (i) the use of an *unconstrained* additive state perturbation, $\mathbf{z} = \bar{\mathbf{z}} + \delta\mathbf{z}$, $\delta\mathbf{z} \in \mathbb{R}^N$, and

(ii) the use of simple *Jacobian* matrices, $\mathbf{F}$ and $\mathbf{G}$, relating the output of our motion and observation models to changes in the state vector, $\mathbf{z} \in \mathbb{R}^N$.

Notably, several representations exist for three-dimensional rotations; however, naively choosing a rotation representation can yield undesirable consequences. For example, a minimal parameterization, such as Euler angles ($\in \mathbb{R}^3$), suffers from singularities, while over-parameterized representations, such as quaternions ($\in \mathbb{R}^4$), required additional constraints. In this thesis, we favour using the constraint-sensitive perturbation schemes detailed in Barfoot and Furgale (2014)[2]; by using an over-parameterized (but singularity-free) 4×4 homogeneous transformation matrix with a 6×1 (constraint-sensitive) perturbation scheme, the typical downfalls of rotation representations (in the Gauss-Newton context) are avoided. Note that we do not require a *vectorspace* state, $\mathbf{z} \in \mathbb{R}^N$, in order to leverage the iterative Gauss-Newton updates in (3.25), but only a *vectorspace* perturbation, $\delta\mathbf{z} \in \mathbb{R}^N$ (assuming that the Gaussian probability densities are properly handled). The remainder of this chapter section is used to review the mathematical machinery that we leverage for unconstrained nonlinear optimizations involving rotations, transformations, and homogeneous points.

3.2.1 Rotations, Transformations, and the Exponential Map

We begin by defining the three-dimensional reference frames, $\underrightarrow{\mathcal{F}}_a$ and $\underrightarrow{\mathcal{F}}_b$, where a vector from $\underrightarrow{\mathcal{F}}_a$ to $\underrightarrow{\mathcal{F}}_b$ (superscript), expressed in $\underrightarrow{\mathcal{F}}_a$ (subscript), is written $\mathbf{v}_a^{b,a}$, and $\mathbf{C}_{b,a}$ is the 3×3 rotation matrix that transforms vectors from $\underrightarrow{\mathcal{F}}_a$ to $\underrightarrow{\mathcal{F}}_b$:

$$\mathbf{v}_b^{b,a} = \mathbf{C}_{b,a}\mathbf{v}_a^{b,a}, \tag{3.32}$$

where $\mathbf{C}_{b,a} \in SO(3)$ and is subject to the constraints,

$$\mathbf{C}_{b,a}\mathbf{C}_{b,a}^T = \mathbf{1}, \quad \det \mathbf{C}_{b,a} = 1. \tag{3.33}$$

Using the exponential map, we note the closed-form expression

$$\mathbf{C}(\boldsymbol{\phi}) := \exp(\boldsymbol{\phi}^\wedge) = \cos\phi\mathbf{1} + (1 - \cos\phi)\mathbf{a}\mathbf{a}^T + \sin\phi\mathbf{a}^\wedge, \tag{3.34}$$

[2]While this thesis provides an overview of the mathematical machinery and identities necessary to perform Gauss-Newton with transformation matrix state variables, a more detailed understanding of how the Gaussian uncertainties are handled on $SE(3)$ can be gained by reading Barfoot and Furgale (2014). Pertaining to optimization, Absil et al. (2009) provides an in-depth discussion of first-order, second-order and trust-region-based techniques for problems involving matrix manifolds.

where $\phi = ||\boldsymbol{\phi}||$ is the angle of rotation, $\mathbf{a} = \boldsymbol{\phi}/\phi$ is the axis of rotation, and $\wedge$ turns $\boldsymbol{\phi} \in \mathbb{R}^3$ into a 3×3 member of the *Lie algebra*, $\mathfrak{so}(3)$(Murray et al., 1994),

$$\boldsymbol{\phi}^\wedge := \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\wedge := \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}. \tag{3.35}$$

Introducing our 4×4 homogeneous transformation matrix definition,

$$\mathbf{T}_{b,a} = \begin{bmatrix} \mathbf{C}_{b,a} & \mathbf{r}_b^{a,b} \\ \mathbf{0}^T & 1 \end{bmatrix} \equiv \begin{bmatrix} \mathbf{C}_{b,a} & -\mathbf{C}_{b,a}\mathbf{r}_a^{b,a} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{3.36}$$

we note that a similar closed-form expression exists by using the exponential map,

$$\mathbf{T}(\boldsymbol{\xi}) := \exp(\boldsymbol{\xi}^\wedge) = \begin{bmatrix} \mathbf{C} & \mathbf{J}\boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3), \tag{3.37}$$

where $\boldsymbol{\xi} \in \mathbb{R}^6$, the overloaded operator, $\wedge$, turns $\boldsymbol{\xi}$ into a 4×4 member of the *Lie algebra* $\mathfrak{se}(3)$(Murray et al., 1994; Barfoot and Furgale, 2014),

$$\boldsymbol{\xi}^\wedge := \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^T & 0 \end{bmatrix}, \quad \boldsymbol{\rho}, \boldsymbol{\phi} \in \mathbb{R}^3, \tag{3.38}$$

the rotation matrix, $\mathbf{C}$, can be computed using (3.34), or the identity

$$\mathbf{C} \equiv \mathbf{1} + \boldsymbol{\phi}^\wedge \mathbf{J}, \tag{3.39}$$

and $\mathbf{J}$ is the (left) Jacobian of $SO(3)$, with the closed-form expression

$$\mathbf{J}(\boldsymbol{\phi}) := \int_0^1 \mathbf{C}^\alpha d\alpha \equiv \frac{\sin\phi}{\phi}\mathbf{1} + \left(1 - \frac{\sin\phi}{\phi}\right)\mathbf{a}\mathbf{a}^T + \frac{1-\cos\phi}{\phi}\mathbf{a}^\wedge. \tag{3.40}$$

To gain some intuition about $\mathbf{J}$, we note that when $\delta\boldsymbol{\varphi}$ is small

$$\exp(\delta\boldsymbol{\phi}^\wedge)\exp(\boldsymbol{\phi}^\wedge) \approx \exp((\boldsymbol{\phi} + \delta\boldsymbol{\varphi})^\wedge), \qquad \delta\boldsymbol{\phi} = \mathbf{J}(\boldsymbol{\phi})\delta\boldsymbol{\varphi}, \tag{3.41}$$

where $\exp(\delta\boldsymbol{\phi}^\wedge) \in SO(3)$ is a small rotation matrix perturbing $\exp(\boldsymbol{\phi}^\wedge) \in SO(3)$. Similarly, for $SE(3)$, when $\delta\boldsymbol{\epsilon}$ is small (Barfoot and Furgale, 2014, Appendix A),

$$\exp(\delta\boldsymbol{\xi}^\wedge)\exp(\boldsymbol{\xi}^\wedge) \approx \exp((\boldsymbol{\xi} + \delta\boldsymbol{\epsilon})^\wedge), \qquad \delta\boldsymbol{\xi} = \boldsymbol{\mathcal{J}}(\boldsymbol{\xi})\delta\boldsymbol{\epsilon}, \tag{3.42}$$

where $\boldsymbol{\mathcal{J}}(\boldsymbol{\xi})$ is the (left) Jacobian of $SE(3)$,

$$\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}) := \int_0^1 \boldsymbol{\mathcal{T}}^\alpha d\alpha \equiv \begin{bmatrix} \mathbf{J}(\boldsymbol{\phi}) & \mathbf{Q}(\boldsymbol{\xi}) \\ \mathbf{0} & \mathbf{J}(\boldsymbol{\phi}) \end{bmatrix} \in \mathbb{R}^{6\times 6}, \tag{3.43}$$

$\boldsymbol{\mathcal{T}}$ is the *adjoint* transformation matrix,

$$\boldsymbol{\mathcal{T}} := \mathrm{Ad}(\mathbf{T}) = \mathrm{Ad}\left(\begin{bmatrix}\mathbf{C} & \mathbf{r}\\ \mathbf{0}^T & 1\end{bmatrix}\right) = \begin{bmatrix}\mathbf{C} & \mathbf{r}^\wedge\mathbf{C}\\ \mathbf{0} & \mathbf{C}\end{bmatrix} = \exp\left(\boldsymbol{\xi}^\curlywedge\right) \in \mathbb{R}^{6\times 6}, \tag{3.44}$$

$(\cdot)^\curlywedge$ is the $SE(3)$ operator,

$$\boldsymbol{\xi}^\curlywedge = \begin{bmatrix}\boldsymbol{\rho}\\ \boldsymbol{\phi}\end{bmatrix}^\curlywedge := \begin{bmatrix}\boldsymbol{\phi}^\wedge & \boldsymbol{\rho}^\wedge\\ \mathbf{0} & \boldsymbol{\phi}^\wedge\end{bmatrix}, \tag{3.45}$$

and $\mathbf{Q}(\boldsymbol{\xi})$ has the closed-form expression

$$\begin{aligned}\mathbf{Q}(\boldsymbol{\xi}) := \frac{1}{2}\boldsymbol{\rho}^\wedge + \tfrac{\phi-\sin\phi}{\phi^3}\left(\boldsymbol{\phi}^\wedge\boldsymbol{\rho}^\wedge + \boldsymbol{\rho}^\wedge\boldsymbol{\phi}^\wedge + \boldsymbol{\phi}^\wedge\boldsymbol{\rho}^\wedge\boldsymbol{\phi}^\wedge\right) - \tfrac{1-\frac{\phi^2}{2}-\cos\phi}{\phi^4}\big(\boldsymbol{\phi}^\wedge\boldsymbol{\phi}^\wedge\boldsymbol{\rho}^\wedge + \boldsymbol{\rho}^\wedge\boldsymbol{\phi}^\wedge\boldsymbol{\phi}^\wedge\\ - 3\boldsymbol{\phi}^\wedge\boldsymbol{\rho}^\wedge\boldsymbol{\phi}^\wedge\big) - \frac{1}{2}\Big(\tfrac{1-\frac{\phi^2}{2}-\cos\phi}{\phi^4} - 3\tfrac{\phi-\sin\phi-\frac{\phi^3}{6}}{\phi^5}\Big)\left(\boldsymbol{\phi}^\wedge\boldsymbol{\rho}^\wedge\boldsymbol{\phi}^\wedge\boldsymbol{\phi}^\wedge + \boldsymbol{\phi}^\wedge\boldsymbol{\phi}^\wedge\boldsymbol{\rho}^\wedge\boldsymbol{\phi}^\wedge\right).\end{aligned} \tag{3.46}$$

Lastly, we introduce the logarithmic map,

$$\boldsymbol{\phi} = \ln(\mathbf{C})^\vee \in \mathbb{R}^3, \quad \boldsymbol{\xi} = \ln(\mathbf{T})^\vee \in \mathbb{R}^6, \tag{3.47}$$

where $\vee$ is the inverse of the overloaded operator $\wedge$ in (3.35) and (3.38).

3.2.2 Homogeneous Points

Using *homogeneous coordinates* we rewrite a point vector, $\mathbf{p}_a^{b,a} \in \mathbb{R}^3$ as,

$$\boldsymbol{p}_a^{b,a} := \eta\begin{bmatrix}\mathbf{p}_a^{b,a}\\ 1\end{bmatrix} = \begin{bmatrix}\boldsymbol{\varepsilon}\\ \eta\end{bmatrix} \in \mathbb{R}^4, \tag{3.48}$$

where η is a non-negative scalar that is used to improve Gauss-Newton conditioning issues when points are infinitely far away (Triggs et al., 2000). For simplicity, the derivations in this thesis will typically assume that $\eta = 1$. One of the key strengths in using 4×4 homogeneous transformation matrices is how easily we can transform a homogeneous point expressed in one frame, $\underrightarrow{\mathcal{F}}_a$, to another, $\underrightarrow{\mathcal{F}}_b$, in a single multiplication:

$$\boldsymbol{p}_b^{c,b} = \mathbf{T}_{b,a}\,\boldsymbol{p}_a^{c,a}. \tag{3.49}$$

We also make use of the following identities in our later derivations:

$$\boldsymbol{\xi}^\wedge\boldsymbol{p} \equiv \boldsymbol{p}^\odot\boldsymbol{\xi}, \tag{3.50a}$$

$$(\mathbf{T}\boldsymbol{p})^\odot \equiv \mathbf{T}\boldsymbol{p}^\odot\boldsymbol{\mathcal{T}}^{-1}, \tag{3.50b}$$

where $(\cdot)^\odot$ is the homogeneous coordinate operator (Barfoot and Furgale, 2014),

$$\boldsymbol{p}^\odot := \begin{bmatrix}\boldsymbol{\varepsilon}\\ \eta\end{bmatrix}^\odot = \begin{bmatrix}\eta\mathbf{1} & -\boldsymbol{\varepsilon}^\wedge\\ \mathbf{0}^T & \mathbf{0}^T\end{bmatrix} \in \mathbb{R}^{4\times 6}. \tag{3.51}$$

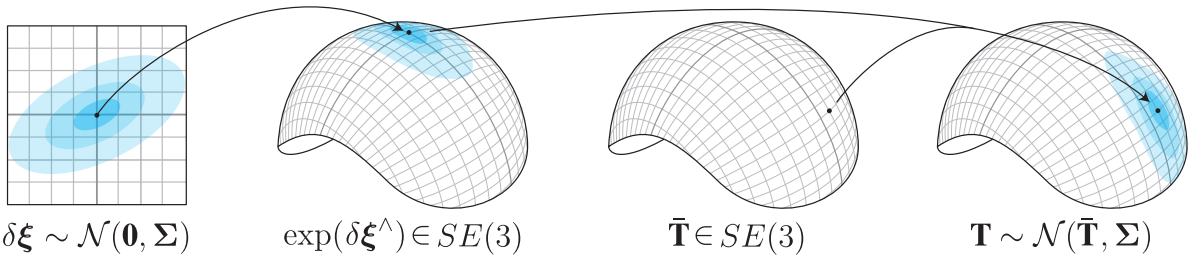


Figure 3.2: This figure depicts the injection of noise directly onto the $SE(3)$ manifold for the random variable $\mathbf{T}$, where $\mathbf{T} = \exp(\delta\boldsymbol{\xi}^{\wedge})\,\bar{\mathbf{T}}$. From left to right: (i) the uncertainty of $\delta\boldsymbol{\xi} \in \mathbb{R}^6$, (ii) a depiction of the uncertainty passed through the exponential map, (iii) the operating point, or *mean*, of $\mathbf{T}$, and (iv) a depiction of the random variable $\mathbf{T}$ on $SE(3)$.

3.2.3 Perturbations

At this point, we have described the necessary mathematical machinery for using transformations and homogeneous points to represent the state of our three-dimensional pose estimation problem. Recalling the problem formulation set up in Section 3.1.2, we now write our discrete set of three-dimensional robot poses as

$$\mathbf{x} = \{\mathbf{T}_{0,i}, \mathbf{T}_{1,i}, \ldots, \mathbf{T}_{k,i}, \ldots, \mathbf{T}_{K,i}\}, \quad \underrightarrow{\mathcal{F}}_k := \underrightarrow{\mathcal{F}}_{\text{rob}}(t_k), \tag{3.52}$$

where $\underrightarrow{\mathcal{F}}_{\text{rob}}(t_k)$ is the robot frame at the time of measurement k, and the pose state variables, $\mathbf{T}_{k,i}$, represent the robot pose with respect to an inertial frame, $\underrightarrow{\mathcal{F}}_i$. Note that we abuse our notation slightly, as $\mathbf{x}$ is not a vector; similarly, we also write $\mathbf{z} = \{\mathbf{x}, \boldsymbol{\ell}\}$.

Next, we must create the connections that allow us to use this state in the probabilistic batch estimator we derived in Section 3.1.1. As mentioned previously, the key to taking advantage of this formulation, despite having non-vectorspace state variables, is being able to formulate a perturbation vector, $\delta\mathbf{z} \in \mathbb{R}^N$. Similar to the vectorspace case, we want to decompose our (desired) posterior state, $\hat{\mathbf{z}}$, into our best guess, $\bar{\mathbf{z}}$, and an unknown perturbation, $\delta\mathbf{z}$. For transformation matrices, this is accomplished by leveraging the exponential map from $\mathbb{R}^6$ to the $SE(3)$ manifold:

$$\hat{\mathbf{T}}_{k,i} = \exp(\delta\boldsymbol{\xi}_{k,i}^{\wedge})\,\bar{\mathbf{T}}_{k,i}, \tag{3.53}$$

where $\bar{\mathbf{T}}_{k,i}$ is our best guess of $\hat{\mathbf{T}}_{k,i}$, and $\exp(\delta\boldsymbol{\xi}_{k,i}^{\wedge})$ is a small transformation matrix that uses the perturbation vector $\delta\boldsymbol{\xi}_{k,i} \in \mathbb{R}^6$ (Barfoot and Furgale, 2014). Within our estimation framework, this style of constraint-sensitive perturbation serves two distinct purposes: (i) to allow the use of non-vectorspace state variables in an unconstrained optimization problem, and (ii) to maintain Gaussian probability distributions associated

with the non-vectorspace state variables. With respect to optimization, we use the implied (constraint-sensitive) update equation,

$$\bar{\mathbf{T}}_{k,i} \leftarrow \exp(\delta\boldsymbol{\xi}_{k,i}^{\wedge})\,\bar{\mathbf{T}}_{k,i}, \tag{3.54}$$

to ensure that $\bar{\mathbf{T}}_{k,i}$ remains on the $SE(3)$ manifold, despite performing an *unconstrained* optimization for the update parameter $\delta\boldsymbol{\xi}_{k,i}$. In order to handle uncertainties, we use the approach of Barfoot and Furgale (2014), which is to store uncertainty using the *probability density function*: $p(\delta\boldsymbol{\xi}_{k,i}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{k,k})$, where $\boldsymbol{\Sigma}_{k,k}$ is a 6×6 covariance matrix (see Figure 3.2); similar approaches to storing uncertainty on $SE(3)$ are explored by Wang and Chirikjian (2006, 2008), Chirikjian (2009), Chirikjian and Kyatkin (2016) and Wolfe et al. (2011). In particular, Chirikjian and Kyatkin (2016) also provides an excellent background on stochastic processes and extend the conversation to Brownian motion of rigid bodies in $SO(3) \times \mathbb{R}^3$.

To handle the landmark parameters, $\boldsymbol{\ell}$, we define the homogeneous points

$$\boldsymbol{p}_i^{\ell_j,i} := \begin{bmatrix} \boldsymbol{\ell}_j \\ 1 \end{bmatrix} \in \mathbb{R}^4, \tag{3.55}$$

and employ the straightforward perturbation scheme,

$$\hat{\boldsymbol{p}}_i^{\ell_j,i} = \bar{\boldsymbol{p}}_i^{\ell_j,i} + \mathbf{D}\,\delta\boldsymbol{\ell}_j, \qquad \mathbf{D} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^T. \tag{3.56}$$

Finally, we are able to write the full state perturbation for a three-dimensional problem:

$$\delta\mathbf{z} := \left[\delta\boldsymbol{\xi}_{0,i}^T \ \ldots\ \delta\boldsymbol{\xi}_{k,i}^T \ \ldots\ \delta\boldsymbol{\xi}_{K,i}^T\, \delta\boldsymbol{\ell}_0^T \ \ldots\ \delta\boldsymbol{\ell}_j^T \ \ldots\ \delta\boldsymbol{\ell}_L^T\right]^T. \tag{3.57}$$

To firmly establish how these perturbations are connected with the Gauss-Newton estimator in (3.25), we demonstrate the linearization of a typical vision-based observation model. Choosing to align the robot frame with the sensor frame, $\underrightarrow{\boldsymbol{\mathcal{F}}}_{\text{rob}}(t) = \underrightarrow{\boldsymbol{\mathcal{F}}}_{s}(t)$, and recalling the nonlinear observation error from (3.10), we now write

$$\mathbf{e}_{m_{kj}}(\mathbf{z}) = \mathbf{y}_{kj} - \mathbf{g}_{kj}(\mathbf{z}), \quad \mathbf{g}_{kj}(\mathbf{z}) := \mathbf{g}(\mathbf{x}_k, \boldsymbol{\ell}_j) = \mathbf{k}(\mathbf{T}_{k,i}\,\boldsymbol{p}_i^{\ell_j,i}), \tag{3.58}$$

where $\mathbf{T}_{k,i}$ transforms the homogeneous landmark point, $\boldsymbol{p}_i^{\ell_j,i}$, into the sensor frame, and $\mathbf{k}(\cdot)$ is the nonlinear *camera* model that projects a homogeneous coordinate into sensor

coordinates (e.g., pixels coordinates). Noting the composition of two nonlinearities, one for the sensor model and one for the point transformation, we begin by defining

$$\mathbf{h}_{kj}(\mathbf{z}) := \mathbf{T}_{k,i}\, \boldsymbol{p}_i^{\ell_j,i}. \tag{3.59}$$

Noting the first-order approximation of the exponential map,

$$\exp(\delta\boldsymbol{\xi}_{k,i}^{\wedge}) \approx \mathbf{1} + \delta\boldsymbol{\xi}_{k,i}^{\wedge}, \tag{3.60}$$

we substitute our perturbation models to find that

$$\begin{aligned}
\mathbf{h}_{kj}(\bar{\mathbf{z}}, \delta\mathbf{z}) &\approx \left(\mathbf{1} + \delta\boldsymbol{\xi}_{k,i}^{\wedge}\right) \bar{\mathbf{T}}_{k,i}\, (\bar{\boldsymbol{p}}_i^{\ell_j,i} + \mathbf{D}\, \delta\boldsymbol{\ell}_j) \\
&\approx \bar{\mathbf{T}}_{k,i}\, \bar{\boldsymbol{p}}_i^{\ell_j,i} + \delta\boldsymbol{\xi}_{k,i}^{\wedge}\, \bar{\mathbf{T}}_{k,i}\, \bar{\boldsymbol{p}}_i^{\ell_j,i} + \bar{\mathbf{T}}_{k,i}\mathbf{D}\, \delta\boldsymbol{\ell}_j \\
&= \bar{\mathbf{T}}_{k,i}\, \bar{\boldsymbol{p}}_i^{\ell_j,i} + \left(\bar{\mathbf{T}}_{k,i}\, \bar{\boldsymbol{p}}_i^{\ell_j,i}\right)^{\odot} \delta\boldsymbol{\xi}_{k,i} + \bar{\mathbf{T}}_{k,i}\mathbf{D}\, \delta\boldsymbol{\ell}_j \\
&= \mathbf{h}_{kj}(\bar{\mathbf{z}}) + \mathbf{H}_{x,kj}\, \delta\boldsymbol{\xi}_{k,i} + \mathbf{H}_{\ell,kj}\, \delta\boldsymbol{\ell}_j,
\end{aligned} \tag{3.61}$$

correct to first order, where

$$\mathbf{h}_{kj}(\bar{\mathbf{z}}) := \bar{\mathbf{T}}_{k,i}\, \bar{\boldsymbol{p}}_i^{\ell_j,i}, \qquad \mathbf{H}_{\mathbf{x},kj} := \left(\bar{\mathbf{T}}_{k,i}\, \bar{\boldsymbol{p}}_i^{\ell_j,i}\right)^{\odot}, \qquad \mathbf{H}_{\boldsymbol{\ell},kj} := \bar{\mathbf{T}}_{k,i}\mathbf{D}. \tag{3.62}$$

Returning to our measurement error term,

$$\begin{aligned}
\mathbf{e}_{m_{kj}}(\bar{\mathbf{z}}, \delta\mathbf{z}) &\approx \mathbf{y}_{kj} - \mathbf{k}(\mathbf{h}_{kj}(\bar{\mathbf{z}}) + \mathbf{H}_{\mathbf{x},kj}\, \delta\boldsymbol{\xi}_{k,i} + \mathbf{H}_{\boldsymbol{\ell},kj}\, \delta\boldsymbol{\ell}_j) \\
&\approx \mathbf{y}_{kj} - \mathbf{k}(\mathbf{h}_{kj}(\bar{\mathbf{z}})) - \mathbf{K}_{kj}\mathbf{H}_{\mathbf{x},kj}\, \delta\boldsymbol{\xi}_{k,i} - \mathbf{K}_{kj}\mathbf{H}_{\boldsymbol{\ell},kj}\, \delta\boldsymbol{\ell}_j \\
&= \mathbf{e}_{m_{kj}}(\bar{\mathbf{z}}) - \mathbf{G}_{\mathbf{x},kj}\delta\boldsymbol{\xi}_{k,i} - \mathbf{G}_{\boldsymbol{\ell},kj}\delta\boldsymbol{\ell}_j,
\end{aligned} \tag{3.63}$$

correct to first order, where

$$\mathbf{e}_{m_{kj}}(\bar{\mathbf{z}}) := \mathbf{y}_{kj} - \mathbf{k}(\mathbf{h}_{kj}(\bar{\mathbf{z}})),\ \mathbf{G}_{\mathbf{x},kj} := \mathbf{K}_{kj}\mathbf{H}_{\mathbf{x},kj},\ \mathbf{G}_{\boldsymbol{\ell},kj} := \mathbf{K}_{kj}\mathbf{H}_{\boldsymbol{\ell},kj},\ \mathbf{K}_{kj} := \left.\frac{\partial \mathbf{k}}{\partial \mathbf{h}_{kj}}\right|_{\bar{\mathbf{z}}}. \tag{3.64}$$

Given that (3.63) has the same form as (3.19b), it is straightforward to see how the Gauss-Newton estimator from (3.25) can be used to solve for an optimal perturbation, $\delta\mathbf{z}^{\star}$, which includes $SE(3)$ perturbations, $\delta\boldsymbol{\xi}_{k,i} \in \mathbb{R}^6$. Our best guess of the poses, $\bar{\mathbf{T}}_{k,i}$, are then updated using the (constraint-sensitive) perturbation scheme in (3.54).

3.3 Constant-Time Algorithms

Recalling the VO paradigm discussed in Chapter 2, we note that the addition of a new frame, $\mathbf{x}_{K+1}$, to a batch SLAM problem that has already been solved, will have

little effect on the majority of the optimal state values (especially older poses that are spatially distant); the exception to this case is loop closure, which provides spatial data associations that can drastically change the structure of our pose-chain estimate. Despite information gain being incremental, the general computational complexity of the batch SLAM formulation is $O(K^3 + K^2L)$, or $O(L^3 + L^2K)$, and becomes intractable to solve over long traversals. In contrast, filtering approaches inherently exploit this incremental nature by only estimating the probability distribution over the most recent pose (and map); however, even filtering approaches will quickly become intractable without proper map management, and typically provide less accuracy due to linearization choices that cannot be undone.

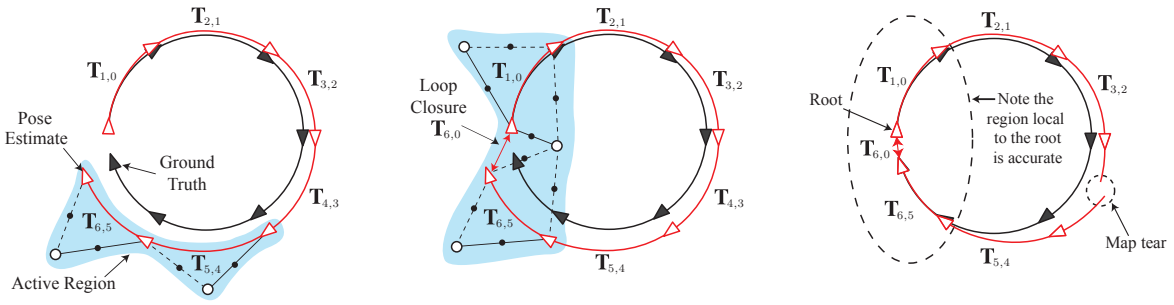
3.3.1 Incremental SLAM

In order to exploit the incremental property of SLAM in a batch context, two paradigms have arisen: (i) *incremental smoothing and mapping* (iSAM), which relies on algebraic manipulation (Kaess et al., 2008), and (ii) the *sliding window filter* (SWF), which finds a middle ground between the batch and filter-based solutions (Sibley et al., 2008).

The first method, iSAM, is based on exploiting the structure of the *square root information matrix* using QR factorization and *column approximate minimal degree* (COLAMD) variable reordering (Dellaert and Kaess, 2006). In essence, this technique directly updates the linear system of equations in light of new information, while only occasionally relinearizing Jacobians whose operating point has changed significantly. Notably, the approach incrementally solves the full nonlinear batch optimization problem without using marginalization.

In contrast, the SWF uses a sliding time window of the most recent measurements to incrementally approximate the all-time maximum likelihood solution. This is accomplished by marginalizing old poses (and landmarks no longer seen by an *active* pose) into a prior distribution over the oldest pose in the window and the *active* landmarks which were seen by marginalized poses; unlike the typical sparse *arrowhead* structure, this prior over the map causes $\mathbf{A}_{\ell\ell}$ to become dense. Drawing parallels to existing solutions, we note that by varying the size of the *window* (in number of poses), the SWF technique scales from the EKF solution to the full batch nonlinear optimization problem.

Notably, both incremental approaches are challenged at loop closure, as spatially distant data associations can require relinearization of the entire trajectory (spanning the



(a) Window-Style Optimization (b) Loop-Closure Optimization (c) Map Consistency

Figure 3.3: In the relative SLAM formulation, the full solution is found by incrementally performing many small batch optimizations. Each of these batch optimization problems estimate a subset of the full state, defined as the *active state*, consisting of a connected set of poses (and the landmarks upon which they depend). (a) In the open-loop case, the *active state* is similar to that of a sliding-window-style filter, and usually spans a few of the most recent poses. (b) At loop closure, the *active state* contains a few poses from each side; when a landmark is observed from poses on both sides of the loop closure, it contributes information to not only the loop-closure estimate, but also the trajectory estimates on each side of the loop closure. (c) The relative formulation leads to a map that is *locally* accurate and can be computed in a fast and incremental fashion; however, the map is globally inconsistent, as resolving it into a single privileged coordinate frame, or *root* frame, causes *map tears* at the far edges.

size of the loop); this is due to the poses (and landmarks) being expressed in a single, privileged, Euclidean coordinate frame. Therefore, the pitfall of the standard batch SLAM problem formulation is that the associated cubic complexity cannot be avoided when a *globally consistent* map (including large-scale loop closures) is required.

3.3.2 Relative Formulation

The completely relative formulation of the bundle adjustment problem, by Sibley et al. (2009), is of particular interest to this thesis. The methodology advocates that a privileged coordinate frame is not required to accomplish many common robotic tasks[3]. By reformulating the batch optimization problem to use a graph of relative transformations, an incremental update strategy can be used to find a close approximation of the full maximum-likelihood solution in constant time – even at loop closure. In implementation,

[3]The notion of abandoning *global* metric accuracy in favour of more efficient *local* topometric accuracy (Sibley et al., 2009) aligns with the principles of core technologies used in the ASRL. Specifically, our core autonomy technology, Visual Teach and Repeat (VT&R), operates on a spatio-temporal pose graph, with no need for a global frame (Furgale and Barfoot, 2010; Stenning et al., 2013).

this works by incrementally updating an adaptive region of state variables that are anticipated to change in light of new information. At loop closure, rather than adjust the entire pose graph, new *topometric* information is added to the graph, in constant time. This technique can be viewed as both a *continuous* submapping approach and a *manifold* representation. Unlike a *global* coordinate representation, which requires *global* consistency, the *spatio-temporal* linkages in a manifold representation allow for graph nodes to have many-to-one world correspondences that are *locally consistent* and Euclidean.

In contrast to the incremental SLAM strategies discussed in the previous section, we note a few key differences with this approach: (i) the robot localization is represented by *relative* transformation matrices across the edges of a graph, (ii) landmark parameters are estimated with respect to local poses (i.e., graph nodes), rather than a privileged frame, and (iii) marginalization is avoided by incrementally solving subsets of the full batch problem. Although leveraging a sliding-window style, this scheme is distinctly different from the sliding-window *filter*, as it does not propagate covariances through marginalization. In short, by holding older poses 'locked', while optimizing for newer poses and common landmarks, the *mean* pose estimate exhibits good accuracy, but the *covariance* estimates are bound to be overconfident. Figure 3.3(a) illustrates the typical sliding-window-style estimation for a relative bundle-adjustment problem. Each time a new frame is added to the system, an optimization problem is run using a subset of the most recent data to estimate the local pose and landmark parameters in the *active state* (shaded region). Although the problem has a cubic solve time in the number of poses being optimized, estimating only a local subset of recent poses keeps this computation time constant as the estimator progresses. The advantage of using a relative system is that when a loop closure is observed, the estimator does not need to optimize over all the poses in the trajectory in order to optimize the new spatial link; as illustrated in Figure 3.3(b), a window-style optimization can be performed using temporally linked poses from both places, and the spatial transformation that relates them.

Compared to a traditional SLAM solution, we note that the relative formulation does not generate a globally consistent map, as seen in Figure 3.3(c). Starting with a root node, the map can be put into a single coordinate frame by compounding poses in a breadth first search (BFS) fashion; this causes *map tears* at the far edges of loops, as the solution is over-parameterized and therefore multiple pose chains exist between any one pose and another. With the exception of a few select tasks, such as *globally optimal* path planning, the majority of mobile robotic tasks, such as graph-based planning, navigation, or object

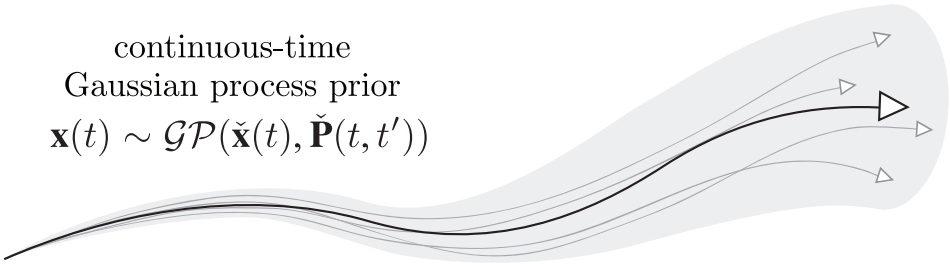


Figure 3.4: This figure depicts four trajectory samples (grey lines) drawn from a continuous-time prior distribution. Using the common Gaussian-process notation (Rasmussen and Williams, 2006), the prior over $\mathbf{x}(t)$ is written as $\mathbf{x}(t) \sim \mathcal{GP}(\check{\mathbf{x}}(t), \check{\mathbf{P}}(t,t'))$, where $\check{\mathbf{x}}(t)$ is the *mean function* (black line) and $\check{\mathbf{P}}(t,t')$ is the *covariance function* (shaded region). More intuitively, this prior can be thought of as a time-evolving Gaussian distribution, where the covariance function has two time parameters to account for cross-temporal correlations.

avoidance, require only locally accurate information and therefore are well-suited to run with a constant-time relative SLAM formulation.

3.4 Continuous-Time Trajectory Estimation

Recalling the discussion of asynchronous (motion-distorted) SURF measurements for VO, in Section 2.2.3, we reiterate that discrete-time representations of robot trajectories do not perform well when estimating motion from certain types of sensors (e.g., rolling-shutter cameras and scanning laser-range finders) and sensor combinations (e.g., high data rate, asynchronous). Specifically, the standard discrete-time batch estimation technique requires a pose variable for each measurement with a unique timestamp and is therefore subject to a few issues: (i) sensors that provide high-rate measurements cause the state size to become unmanageable, (ii) asynchronous visual measurements inadequately constrain the pose variables (without a motion model or odometry-type measurements to correlate the poses), and (iii) unsynchronized measurements from multiple sensors are not easily fused to a common set of state variables. Many works have remedied these types of situations (for discrete-time batch SLAM) by making ad hoc assumptions about the smoothness of the trajectory; for example, by using a local, piece-wise, constant-velocity model (Ait-Aider et al., 2006; Davison et al., 2007; Hedborg et al., 2012; Dong and Barfoot, 2012; Anderson and Barfoot, 2013a), or even more general models based on splines (Bosse and Zlot, 2009; Bibby and Reid, 2010; Bosse et al., 2012; Zlot and Bosse, 2012, 2014).

Given the nature of the trajectories we wish to estimate and inspired by the power of

interpolation as a smoothness constraint in the discrete-time batch SLAM problem, we note that *batch continuous-time trajectory estimation* is the logical next step. However, rather than choosing a smoothness constraint ad hoc, it is proposed that the smoothness assumption should be built directly into the estimator by using a prior distribution over continuous-time trajectories (see Figure 3.4). Furthermore, we bring to attention that continuous-time estimation is not a new subject, but has been available since the 1960s (Kalman and Bucy, 1961; Jazwinski, 1970) and has been predominantly ignored in the (roughly) two decades of SLAM research.

Building on the foundations of discrete-time batch SLAM and continuous-time filtering techniques, an explicit probabilistic formulation for *batch* continuous-time trajectory estimation was introduced for robotics (Furgale et al., 2012, 2015). Recalling (3.3), Furgale et al. (2012) propose solving the similar probabilistic problem,

$$\{\hat{\mathbf{x}}(t), \hat{\boldsymbol{\ell}}\} = \underset{\mathbf{x}(t), \boldsymbol{\ell}}{\operatorname{argmax}} \; p(\mathbf{x}(t), \boldsymbol{\ell} \,|\, \mathbf{u}(t), \mathbf{y}), \tag{3.65}$$

where $\mathbf{x}(t)$ is the robot pose at time t, over the interval $[t_0, t_K]$, and $\mathbf{u}(t)$ is the continuous-time control signal. As we are now interested in the full continuous-time trajectory, $\mathbf{x}(t)$, we refer to this subtle SLAM generalization as *simultaneous trajectory estimation and mapping* (STEAM). The primary deviation from the discrete-time SLAM derivation is with respect to the motion model. Similar to (3.1), but in continuous-time, the density $p(\mathbf{x}(t) \,|\, \mathbf{u}(t))$ can be described using a *stochastic differential equation* (SDE),

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{w}(t), \quad \mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_C \delta(t - t')), \tag{3.66}$$

where $\mathbf{f}(\cdot)$ is a nonlinear function, and $\mathbf{w}(t)$ is a zero-mean, white Gaussian process, with covariance function $\mathbf{Q}_C \delta(t - t')$ (where $\mathbf{Q}_C$ is a power-spectral-density matrix and $\delta(\cdot)$ is Dirac's delta function). The objective function cost related to the prior is then

$$J_p(\mathbf{x}(t)) := \frac{1}{2} \mathbf{e}_{p_0}^T \check{\mathbf{P}}_0^{-1} \mathbf{e}_{p_0} + \frac{1}{2} \int_{t_0}^{t_K} \mathbf{e}_u(\tau)^T \mathbf{Q}_C^{-1} \mathbf{e}_u(\tau) d\tau, \tag{3.67}$$

where the *error terms* are:

$$\mathbf{e}_{p_0} := \mathbf{x}(t_0) - \check{\mathbf{x}}(t_0), \tag{3.68a}$$

$$\mathbf{e}_u(t) := \mathbf{e}_u(\mathbf{x}(t)) = \dot{\mathbf{x}}(t) - \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)). \tag{3.68b}$$

Alternatively, the prior distribution of trajectories resulting from the solution of (3.66) can be written as the Gaussian-process prior (see Figure 3.4):

$$\mathbf{x}(t) \sim \mathcal{GP}(\check{\mathbf{x}}(t), \check{\mathbf{P}}(t, t')), \tag{3.69}$$

where $\check{\mathbf{x}}(t)$ is the prior *mean function* (driven by the control inputs, $\mathbf{u}(t)$) and $\check{\mathbf{P}}(t,t')$ is the prior *covariance function* (which encapsulates the injection of process noise, $\mathbf{w}(t)$).

Concerning the formulation of observation measurements, we note that the discrete-time state variables, $\mathbf{x}_k$, are simply replaced with temporal samples of the continuous-time trajectory, $\mathbf{x}(t_k)$. Since we cannot perform state estimation on $\mathbf{x}(t)$ directly (as it would require an *infinite* number of states) we are forced to choose some sort of discrete representation. As we will discover, this is not a trivial decision; however, it need not be direct temporal sampling (as in the discrete-time batch formulation).

Within the STEAM field of research, two paradigms have emerged for continuous-time trajectory representation: (i) a parametric approach using temporal basis functions (Furgale et al., 2012; Anderson and Barfoot, 2013b; Lovegrove et al., 2013; Oth et al., 2013; Anderson et al., 2014; Sommer et al., 2015; Furgale et al., 2015), and (ii) a nonparametric approach using Gaussian-process regression (Tong et al., 2013; Barfoot et al., 2014; Anderson et al., 2015a; Anderson and Barfoot, 2015). Additional detail and more specific references to the existing literature are given in Chapters 5 and 6, where this thesis provides novel contributions to each of the aforementioned paradigms.

3.5 Summary

In this thesis, we advocate for the use of batch estimation, rather than filtering, in order to solve the SLAM problem; using modern computing, batch methods have proven to be more accurate (per unit of computation) than filtering approaches (Strasdat et al., 2010). In Section 3.1 we reviewed the typical probabilistic formulation for the discrete-time batch SLAM problem, showed how to perform the optimization using Gauss-Newton, and described algebraic methods to exploit the general SLAM sparsity patterns. Section 3.2 then presented some relevant Lie group theory that allows us to extend the Gauss-Newton algorithm to solve six-dimensional problems (involving rotation and translation in three-dimensions). Strategies to run the batch SLAM estimator incrementally were discussed in Section 3.3. Finally, Section 3.4 introduced the notion of STEAM, a generalization of SLAM that uses a batch, continuous-time trajectory estimation scheme. The mathematics and ideas presented in this chapter form the basis for the contributions and experiments presented in this thesis. Consequently, it should be expected that the following thesis chapters will make extensive references to the equations and ideas presented herein.

Chapter 4

Outlier Rejection for Motion-Distorted 3D Sensors

This chapter describes our novel approach to outlier rejection for sparse-feature data associations extracted from sensor data that is susceptible to motion distortion. Recalling our discussion of the standard VO pipeline in Chapter 2, we note that *feature correspondence* (i.e., data association) and *outlier rejection* are a precursor to finding a trajectory estimate. Given two sequential sets of sparse features, an initial guess of the data associations between them is often generated using some type of similarity factor; for vision-based data, this is an appearance-based feature descriptor, such as the 64-float descriptor used by SURF. However, descriptors alone do not take into account the geometry of the problem and are highly prone to generating mismatches. The standard outlier-rejection scheme for filtering the initial associations is *Random Sample Consensus* (RANSAC), which is a nondeterministic algorithm for robustly finding the parameters of a mathematical model that best describe a likely set of inliers (Fischler and Bolles, 1981). With a global-shutter camera, the model used in the standard RANSAC algorithm is a single rigid transformation (i.e., translation and rotation).

The novel contributions of this chapter are: (i) a RANSAC algorithm that uses a 6D constant-velocity model in order to account for the temporal nature of rolling-shutter-type images, (ii) approximations to increase the computational efficiency of our algorithm, (iii) a derivation of the conditions for which our method can be solved, and (iv) an experiment validation using the 1.1 km lidar intensity image dataset described in Appendix A. These contributions have appeared in two previous publications. Initial results were first presented in the proceedings of a full-paper refereed conference (Anderson and Barfoot,

2013a). Combined with the contributions from Chapter 5, this work was later published in a journal article (Anderson et al., 2015b), which extended the original paper by deriving the conditions for which our model can be estimated.

4.1 Related Work

The RANSAC algorithm (Fischler and Bolles, 1981) has been highly successful as an outlier-rejection scheme in the visual pipeline and has become popularized due to its speed and efficiency. Using global shutter cameras, both the monocular 5-point algorithm (Nistér, 2004) and stereo-pair, 3-point algorithm (Haralick et al., 1994) are widely used – most notably on the Mars Exploration Rovers (Maimone et al., 2007). Although the use of CMOS technology is desirable due to low cost and widespread availability, the addition of a temporal parameter to the standard camera model causes nontrivial and unhandled complexities in the mature visual pipeline. Much of the existing rolling-shutter literature makes use of special cases and there are only a few pieces of work that properly account for the 6D sensor motion. However, many of these methods stray from using RANSAC as an outlier-rejection scheme.

Early work by Ait-Aider et al. (2006) develops a nonlinear, least-squares estimator for the velocity of a monocular rolling-shutter camera over a single frame of data; in order to find this velocity over only a single image, a known geometric target is used, and the feature correspondence is supervised. In a separate work, Ait-Aider and Berry (2009) also develop a batch nonlinear optimization technique for a stereo rig that includes only one rolling-shutter camera, and one global-shutter camera. Again, this experiment uses a target with known geometry and easy-to-find feature correspondences. The possibility of using RANSAC is mentioned, but left as a future extension.

Jia and Evans (2012) apply the extended Kalman filter (EKF) to a monocular rolling-shutter camera and use predictive measurements from a gyroscope and accelerometer. In a similar fashion to RANSAC, this algorithm handles outlier rejection by applying the EKF correction step for multiple feature correspondence hypotheses and checking to see which update generated the most likely set of inliers. Akin to the classic RANSAC algorithm, our proposed method does not require additional sensors, such as a gyroscope, to find feature correspondences.

The primary outlier-rejection scheme for the aforementioned visual pipeline by Hedborg et al. (2012) is a *cross-check* method, which forgoes the use of camera geometry and

instead relies on the use of a feature tracker, such as Kanade-Lucas-Tomasi (KLT) (Lucas and Kanade, 1981). After an initial bundle-adjustment step, a second stage of outlier rejection is applied by checking point triangulations from multiple viewpoints. In an earlier work, Hedborg et al. (2011) noted that the use of a *cross check* with KLT is ideal for sensors with a very high framerate. In contrast to the 30Hz camera used by Hedborg et al. (2011), our method aims to find feature correspondences between rolling-shutter-type images taken at 2Hz. Furthermore, Hedborg et al. (2011) take advantage of a secondary rejection step that is tightly coupled with the estimation phase of the visual pipeline. In the frame-to-frame estimation technique proposed by Dong and Barfoot (2012), a typical 3-point RANSAC algorithm is applied with a very loose threshold that allows for some outliers. In the estimation phase, a robust M-estimation scheme is then used to try to minimize the effect of incorrectly matched features. By using a constant-velocity model in the RANSAC algorithm, the method proposed in this chapter separates outlier rejection from the estimation phase.

4.2 Problem Formulation

For the feature correspondence problem, we define two sets of corresponding measurements, $\mathbf{y}_{m,1}$ and $\mathbf{y}_{m,2}$, where $m = 1 \dots M$. Each measurement pair, m, is extracted from sequential images 1 and 2 at times, $t_{m,1}$ and $t_{m,2}$, with a temporal difference of $\Delta t_m := t_{m,2} - t_{m,1}$. The sensor models for these measurements are:

$$\mathbf{y}_{m,1} := \mathbf{k}(\mathbf{T}(t_{m,1})\boldsymbol{p}_m) + \mathbf{n}_{m,1}, \tag{4.1a}$$

$$\mathbf{y}_{m,2} := \mathbf{k}(\mathbf{T}(t_{m,2})\boldsymbol{p}_m) + \mathbf{n}_{m,2}, \tag{4.1b}$$

where $\mathbf{k}(\cdot)$ is the nonlinear camera model, $\mathbf{T}(t)$ is the 4×4 homogeneous transform matrix that specifies the pose of the sensor frame, $\underrightarrow{\mathcal{F}}_s(t)$, with respect to the inertial frame, $\underrightarrow{\mathcal{F}}_i$, at time t, and the measurement noises, $\mathbf{n}_{m,1} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{m,1})$ and $\mathbf{n}_{m,2} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{m,2})$, are assumed to be normally distributed with covariances $\mathbf{R}_{m,1}$ and $\mathbf{R}_{m,2}$. Note that each measurement pairing, m, is simply the hypothesis of a common landmark, typically based on the similarity of appearance-based feature descriptors, and may not actually be projected from the same 3D location, $\boldsymbol{p}_m := \boldsymbol{p}_i^{\ell_m,i} = \begin{bmatrix} \boldsymbol{\ell}_m^T & 1 \end{bmatrix}^T$. Therefore, the goal of outlier rejection is to determine the subset of all M measurement pairs, $\mathbf{y}_{m,1}$ and $\mathbf{y}_{m,2}$, that are *truly* projected from common 3D locations (i.e., the most likely set of inliers).

The structure of the RANSAC algorithm consists of only a few main steps. It begins

by selecting N random subsets of S measurement pairs, where S is the minimum number of pairs needed to solve the mathematical model. When using 3D point correspondences, such as in stereo camera or lidar data, S is typically 3. The number of required iterations, N, can be chosen using the equation

$$N = \frac{\ln(1 - p_{\text{succ}})}{\ln(1 - p_{\text{in}}^S)}, \tag{4.2}$$

where p_{succ} is the probability of choosing S measurement pairs that are all inliers and p_{in} is the probability of a single pair, m, being an inlier. The first step (for each of the N subsets) is to solve a chosen mathematical model[1], which in turn can be used to find the unknown pose change of the sensor frame, $\mathbf{T}_m$, between times $t_{m,1}$ and $t_{m,2}$,

$$\mathbf{T}_m := \mathbf{T}(t_{m,2})\mathbf{T}(t_{m,1})^{-1}. \tag{4.3}$$

An estimate of the transform, generated from the chosen model and minimal number of measurement pairs, is denoted as $\bar{\mathbf{T}}_m$. The second step is to apply the transform hypothesis, $\bar{\mathbf{T}}_m$, to the measurements $\mathbf{y}_{m,1}$, using

$$\bar{\boldsymbol{p}}_{m,2} := \bar{\mathbf{T}}_m \boldsymbol{p}_{m,1}, \quad \boldsymbol{p}_{m,1} := \mathbf{k}^{-1}(\mathbf{y}_{m,1}), \tag{4.4}$$

where the calculation of $\boldsymbol{p}_{m,1}$ is independent of the mathematical model and is performed only once for all M pairs. Note we take advantage of the invertible camera model available to 3D sensors. This assumption prevents us from directly applying our method to a monocular rolling-shutter camera; however, with some special care we believe it is possible to extend our technique for the monocular case. The transformed measurements, $\bar{\boldsymbol{p}}_{m,2}$, are then reprojected back into measurement space:

$$\bar{\mathbf{y}}_{m,2} := \mathbf{k}(\bar{\boldsymbol{p}}_{m,2}). \tag{4.5}$$

Finally, each model is evaluated by finding the number of measurement pairs, $\mathbf{y}_{m,1}$ and $\mathbf{y}_{m,2}$, that satisfy the criterion:

$$(\mathbf{y}_{m,2} - \bar{\mathbf{y}}_{m,2})^T \mathbf{R}_{m,2}^{-1} (\mathbf{y}_{m,2} - \bar{\mathbf{y}}_{m,2}) < \alpha, \tag{4.6}$$

where α is a threshold on normalized reprojection error. For simplicity, the model with the highest number of inliers is chosen.

[1] At this point in the derivation, we have yet to choose a model. The classic model for vision is simply a rigid transform. In this thesis, we explore the use of a constant-velocity model.

4.3 Classic Rigid RANSAC

The mathematical model used in the classic RANSAC algorithm is a single rigid pose change between two static frames. It does not take into consideration the possibility of individual measurement times, and in order to use it, we must approximate our motion-distorted measurements as occurring at nominal image times, $\bar{t}_1$ and $\bar{t}_2$. The approximation being made is that

$$\Delta t_m = t_{m,2} - t_{m,1} \approx \bar{t}_2 - \bar{t}_1. \tag{4.7}$$

The transform, $\mathbf{T}_m$, is therefore the same for all pairs, m, and can be solved with only three non-collinear point pairs, in closed form, using the algorithms presented in Horn (1987), Arun et al. (1987), or Umeyama (1991). This assumption is reasonable for rolling-shutter-type images that have little to no distortion. However, under significant motion it quickly becomes clear that the rigid algorithm is unfit for identifying a good set of inliers.

Setting the threshold on reprojection error, α, appropriately, the rigid RANSAC algorithm is limited to finding inliers that have a similar temporal difference, Δt_m. In practice, we tuned α to find as many inliers as possible (over the whole trajectory), without including outliers at standstill. Recalling Chapter 2, due to the slow vertical scan of our sensor, it is only possible to match a temporal band of features at high vehicle speeds, as seen Figure 2.6(a). Loosening α to allow for more feature matches at high vehicle speeds also allows for the inclusion of outliers, as seen in Figure 2.6(b).

4.4 Motion-Compensated RANSAC

To compensate for motion during image capture, it is proposed that we can approximate the motion of the sensor as a constant velocity, $\boldsymbol{\varpi} := \begin{bmatrix} \boldsymbol{\nu}^T & \boldsymbol{\omega}^T \end{bmatrix}^T$, where $\boldsymbol{\nu}$ and $\boldsymbol{\omega}$ are the linear and angular components, expressed in the sensor frame (Anderson and Barfoot, 2013a). In order to calculate $\bar{\mathbf{y}}_{m,2}$, we derive the relationship between the sensor velocity, $\boldsymbol{\varpi}$, and the transform $\mathbf{T}_m$. Assuming constant velocity, the desired transform is simply

$$\mathbf{T}_m = \exp(\Delta t_m \, \boldsymbol{\varpi}^\wedge). \tag{4.8}$$

The major difference in moving from a rigid model to a constant-velocity model is that there now exists a different pose change for each measurement pair, based on their temporal difference, Δt_m.

4.4.1 Nonlinear Least-Squares Estimator

For each of the N subsets, the first major RANSAC step is to solve for the optimal constant velocity, $\boldsymbol{\varpi}$, that minimizes the S paired measurements' reprojection error. To do this, we setup a typical nonlinear least-squares estimation scheme, similar to that of a *bundle adjustment* problem. The objective function that we wish to minimize is simply

$$J(\boldsymbol{\varpi}) := \frac{1}{2}\sum_{m=1}^{M} \mathbf{e}_m^T \mathbf{R}_{m,2}^{-1} \mathbf{e}_m, \tag{4.9}$$

where the error term is defined as

$$\mathbf{e}_m := \mathbf{y}_{m,2} - \bar{\mathbf{y}}_{m,2} = \mathbf{y}_{m,2} - \mathbf{k}(\mathbf{T}_m \boldsymbol{p}_{m,1}). \tag{4.10}$$

Note that we avoid estimating the landmark positions by assuming perfect knowledge of $\boldsymbol{p}_{m,1}$ (i.e., no noise on the measurement $\mathbf{y}_{m,1}$). Similar to our previous estimation schemes, we formulate a Gauss-Newton problem by linearizing our error, $\mathbf{e}_m$, with a perturbation to our state variable, $\boldsymbol{\varpi}$. Starting with the transformation nonlinearity, we define

$$\mathbf{h}_m(\boldsymbol{\varpi}) := \mathbf{T}_m \boldsymbol{p}_{m,1}. \tag{4.11}$$

Consider the perturbation to the velocity,

$$\mathbf{T}_m = \exp(\Delta t_m \, \boldsymbol{\varpi}^\wedge) = \exp(\Delta t_m \, (\bar{\boldsymbol{\varpi}} + \delta\boldsymbol{\varpi})^\wedge), \tag{4.12}$$

where $\bar{\boldsymbol{\varpi}}$ is the nominal solution and $\delta\boldsymbol{\varpi}$ is the perturbation. Recalling the relationship between an additive and multiplicative perturbation in (3.42), we find that,

$$\mathbf{T}_m \approx \exp(\Delta t_m \cdot (\bar{\boldsymbol{\mathcal{J}}}_m \delta\boldsymbol{\varpi})^\wedge) \exp(\Delta t_m \cdot \bar{\boldsymbol{\varpi}}^\wedge) \tag{4.13a}$$

$$= \exp(\Delta t_m \cdot (\bar{\boldsymbol{\mathcal{J}}}_m \delta\boldsymbol{\varpi})^\wedge) \bar{\mathbf{T}}_m, \tag{4.13b}$$

where $\bar{\boldsymbol{\mathcal{J}}}_m := \boldsymbol{\mathcal{J}}(\Delta t_m \bar{\boldsymbol{\varpi}})$. Using the small-pose approximation found in (3.60),

$$\mathbf{T}_m \approx (\mathbf{1} + \Delta t_m \cdot (\bar{\boldsymbol{\mathcal{J}}}_m \delta\boldsymbol{\varpi})^\wedge) \bar{\mathbf{T}}_m. \tag{4.14}$$

Applying this perturbation scheme to (4.11), we have

$$\mathbf{h}_m(\bar{\boldsymbol{\varpi}} + \delta\boldsymbol{\varpi}) \approx \bar{\mathbf{h}}_m + \mathbf{H}_m \delta\boldsymbol{\varpi}, \tag{4.15}$$

correct to first order, where,

$$\bar{\mathbf{h}}_m := \bar{\mathbf{T}}_m \boldsymbol{p}_{m,1}, \quad \mathbf{H}_m := \Delta t_m (\bar{\mathbf{T}}_m \boldsymbol{p}_{m,1})^\odot \bar{\boldsymbol{\mathcal{J}}}_m. \tag{4.16}$$

Returning to our measurement error term, we have

$$\mathbf{e}_m \approx \mathbf{y}_{m,2} - \mathbf{k}(\bar{\mathbf{h}}_m + \mathbf{H}_m \delta\boldsymbol{\varpi}) \approx \bar{\mathbf{e}}_m - \mathbf{G}_m \delta\boldsymbol{\varpi}, \tag{4.17}$$

correct to first order, where

$$\bar{\mathbf{e}}_m := \mathbf{y}_{m,2} - \mathbf{k}(\bar{\mathbf{h}}_m), \qquad \mathbf{G}_m := \mathbf{K}_m \mathbf{H}_m, \qquad \mathbf{K}_m := \left.\frac{\partial \mathbf{k}}{\partial \mathbf{h}}\right|_{\bar{\mathbf{h}}_m}. \tag{4.18}$$

Taking our usual Gauss-Newton approach, described in Section 3.1.2, we find the optimal state update equation for a single iteration:

$$\left(\sum_m \mathbf{G}_m^T \mathbf{R}_{m,2}^{-1} \mathbf{G}_m\right) \delta\boldsymbol{\varpi}^\star = \sum_m \mathbf{G}_m^T \mathbf{R}_{m,2}^{-1} \bar{\mathbf{e}}_m. \tag{4.19}$$

Using the normal iterative method, we solve the system of equations for $\delta\boldsymbol{\varpi}^\star$, and update the nominal solution, $\bar{\boldsymbol{\varpi}} \leftarrow \bar{\boldsymbol{\varpi}} + \delta\boldsymbol{\varpi}^\star$, until convergence. The estimator requires a minimum of three well-spaced point correspondences (shown in Section 4.6). Note that three is only the minimum number of correspondences required – the estimator can also be used to improve the $\boldsymbol{\varpi}$ estimate after finding a larger set of likely inliers.

4.4.2 Point Transformation

In order to compare each of the N constant-velocity models, we must evaluate the number of inliers. This is done by transforming each of the measurements, $\mathbf{y}_{m,1}$, into the corresponding frame, $\underrightarrow{\mathcal{F}}_s(t_{m,2})$. The required transform, $\mathbf{T}_m$, can be calculated using (4.8). In contrast to the rigid RANSAC algorithm, which calculates only one transform per model, our motion-compensated RANSAC algorithm requires that $\mathbf{T}_m$ be evaluated for each pair, m, using each of the N constant-velocity models.

4.5 Fast Motion-Compensated RANSAC

The computation of a transform, $\mathbf{T}_m$, for each measurement pair, m, using each of the N constant-velocity models, adds a significant amount of overhead to the computational cost. Furthermore, the iterative estimator has to run for each of the N randomly seeded measurement subsets. In order to improve the performance of the algorithm, we propose a Euclidean least-squares estimator and a heuristic for the point-transformation step.

4.5.1 Euclidean Least-Squares Estimator

We begin by reformulating the error in Euclidean space:

$$\mathbf{e}_m := \boldsymbol{p}_{m,2} - \bar{\boldsymbol{p}}_{m,2}, \qquad \bar{\boldsymbol{p}}_{m,2} = \mathbf{T}_m \boldsymbol{p}_{m,1}, \tag{4.20}$$

where, for notational simplicity, we have assumed that $\eta = 1$ in our homogeneous point representation (recall (3.48)). The advantage of this form is that it eliminates the need to linearize the camera model. The disadvantage is that the algorithm no longer optimizes with respect to the error that is evaluated in the inlier criterion (4.6).

Given that the two images are sequential, and relatively close in time, we propose the assumption that $\mathbf{T}_m$ is 'small', and therefore can be approximated as $\mathbf{1} + \Delta t_m \boldsymbol{\varpi}^\wedge$. Inserting this into our error term, we have

$$\mathbf{e}_m \approx \bar{\mathbf{e}}_m - \mathbf{G}_m \boldsymbol{\varpi}, \quad \bar{\mathbf{e}}_m := \boldsymbol{p}_{m,2} - \boldsymbol{p}_{m,1}, \quad \mathbf{G}_m := \Delta t_m \boldsymbol{p}_{m,1}^{\odot}. \tag{4.21}$$

Inserting $\mathbf{e}_m$ into $J(\boldsymbol{\varpi}) = \frac{1}{2}\sum_m \mathbf{e}_m^T \mathbf{e}_m$ and setting $\frac{\partial J}{\partial \boldsymbol{\varpi}^T} = \mathbf{0}$, we have

$$\left(\sum_m \mathbf{G}_m^T \mathbf{G}_m\right) \boldsymbol{\varpi} = \sum_m \mathbf{G}_m^T \mathbf{e}_m, \tag{4.22}$$

which can be solved for $\boldsymbol{\varpi}$ in one step (i.e., without iteration).

4.5.2 Discretization of Required Transforms

The motion-compensated problem formulation requires the calculation of the transform, $\mathbf{T}_m$, for each measurement pair, m, using each of the N hypothesized constant-velocity models. In order to improve computational performance, it is proposed that for each of the N models, finding only a discretized subset of the transforms is admissible. Based on the time differences, Δt_m, the measurement pairs, m, are sorted into a set of discrete bins, uniformly spaced between the minimum and maximum values of Δt_m; a transform is then calculated for each bin, using its centre time. During the evaluation of each measurement pair, m, the transform $\mathbf{T}_m$ is approximated by the appropriate bin transform.

4.6 Minimal Point Set and Sufficient Conditions

Following from the derivation of our nonlinear motion-compensated RANSAC algorithm, in Section 4.4.1, we now derive the conditions under which we can solve (4.19) for a

six-dimensional constant velocity. To simplify the derivation, we assume the use of Euclidean point measurements, defining our camera model as,

$$\mathbf{k}(\mathbf{v}) := \mathbf{v}, \tag{4.23}$$

and using an isotropic covariance,

$$\mathbf{R}_{m,2} = \sigma^2 \mathbf{1}. \tag{4.24}$$

Using these simplifying assumptions, the system of equations in (4.19), becomes,

$$\left(\sum_m \mathbf{G}_m^T \mathbf{G}_m\right) \delta\boldsymbol{\varpi}^\star = \sum_m \mathbf{G}_m^T (\boldsymbol{p}_{m,2} - \bar{\boldsymbol{p}}_{m,2}), \tag{4.25}$$

correct to first order, where

$$\bar{\boldsymbol{p}}_{m,2} := \bar{\mathbf{T}}_m \boldsymbol{p}_{m,1} = \exp(\Delta t_m \bar{\boldsymbol{\varpi}}^\wedge) \boldsymbol{p}_{m,1}, \tag{4.26a}$$

$$\mathbf{G}_m := \Delta t_m \bar{\boldsymbol{p}}_{m,2}^\odot \boldsymbol{\mathcal{J}}(\Delta t_m \bar{\boldsymbol{\varpi}}), \tag{4.26b}$$

and the points, $\boldsymbol{p}_{m,1}$ and $\boldsymbol{p}_{m,2}$, are matched point measurements, the temporal difference between measurements is Δt_m, and $\bar{\boldsymbol{\varpi}}$ is the nominal constant-velocity estimate. In an iterative fashion, the nominal solution is updated, using $\bar{\boldsymbol{\varpi}} \leftarrow \bar{\boldsymbol{\varpi}} + \delta\boldsymbol{\varpi}^\star$, and will converge to a local minimum determined by the objective function and the initial condition.

At this point in the derivation, we require a simplification in order to show the conditions under which the system is solvable. Rather than make the harsh assumption that the product $\Delta t_m \bar{\boldsymbol{\varpi}}$ is small, such that $\bar{\mathbf{T}}_m \approx \mathbf{1}$ and $\boldsymbol{\mathcal{J}}(\Delta t_m \bar{\boldsymbol{\varpi}}) \approx \mathbf{1}$, as we did in the fast and approximate implementation of our motion-compensated RANSAC algorithm, we instead make the less restrictive assumption that $\Delta t_m \bar{\boldsymbol{\varpi}}$ is small enough, such that,

$$\boldsymbol{\mathcal{J}}(\Delta t_m \bar{\boldsymbol{\varpi}}) \approx \boldsymbol{\mathcal{T}}(\tfrac{1}{2} \Delta t_m \bar{\boldsymbol{\varpi}}), \tag{4.27}$$

where the functions $\boldsymbol{\mathcal{J}}(\boldsymbol{\xi})$ and $\boldsymbol{\mathcal{T}}(\frac{1}{2}\boldsymbol{\xi})$ are (Barfoot and Furgale, 2014),

$$\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}) = \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\boldsymbol{\xi}^\curlywedge)^n = \mathbf{1} + \frac{1}{2}\boldsymbol{\xi}^\curlywedge + \frac{1}{6}\boldsymbol{\xi}^\curlywedge \boldsymbol{\xi}^\curlywedge + \frac{1}{24}\boldsymbol{\xi}^\curlywedge \boldsymbol{\xi}^\curlywedge \boldsymbol{\xi}^\curlywedge + \dots, \tag{4.28a}$$

$$\boldsymbol{\mathcal{T}}(\tfrac{1}{2}\boldsymbol{\xi}) = \sum_{n=0}^{\infty} \frac{1}{n!} (\tfrac{1}{2}\boldsymbol{\xi}^\curlywedge)^n = \mathbf{1} + \frac{1}{2}\boldsymbol{\xi}^\curlywedge + \frac{1}{8}\boldsymbol{\xi}^\curlywedge \boldsymbol{\xi}^\curlywedge + \frac{1}{48}\boldsymbol{\xi}^\curlywedge \boldsymbol{\xi}^\curlywedge \boldsymbol{\xi}^\curlywedge + \dots. \tag{4.28b}$$

The assumption is correct for terms in $\mathbf{G}_m^T \mathbf{G}_m$ up to order Δt_m^3, and a good approximation thereafter for small products of $\Delta t_m \bar{\boldsymbol{\varpi}}$. In combination with (3.50b), we find that

$$\mathbf{G}_m \approx \Delta t_m \exp(\tfrac{1}{2}\Delta t_m \bar{\boldsymbol{\varpi}}^\wedge) \left(\exp(\tfrac{1}{2}\Delta t_m \bar{\boldsymbol{\varpi}}^\wedge) \boldsymbol{p}_{m,1}\right)^\odot. \tag{4.29}$$

It follows that

$$\sum_m \mathbf{G}_m^T \mathbf{G}_m \approx \boldsymbol{\mathcal{M}}, \tag{4.30}$$

where $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{6\times 6}$ has the form of a *generalized mass matrix* (Murray et al., 1994),

$$\boldsymbol{\mathcal{M}} := \sum_m \Delta t_m^2 \mathbf{h}_m^{\odot T} \mathbf{h}_m^{\odot} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ -w\mathbf{r}^\wedge & \mathbf{1} \end{bmatrix} \begin{bmatrix} w\mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{1} & w\mathbf{r}^\wedge \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \tag{4.31}$$

where

$$\mathbf{h}_m = \begin{bmatrix} \mathbf{r}_m \\ 1 \end{bmatrix} = \exp(\tfrac{1}{2}\Delta t_m \bar{\boldsymbol{\varpi}}^\wedge) \boldsymbol{p}_{m,1}, \qquad w = \sum_m \Delta t_m^2, \tag{4.32a}$$

$$\mathbf{r} = \frac{1}{w} \sum_m \Delta t_m^2 \mathbf{r}_m, \qquad \mathbf{I} = -\sum_m \Delta t_m^2 (\mathbf{r}_m - \mathbf{r})^\wedge (\mathbf{r}_m - \mathbf{r})^\wedge. \tag{4.32b}$$

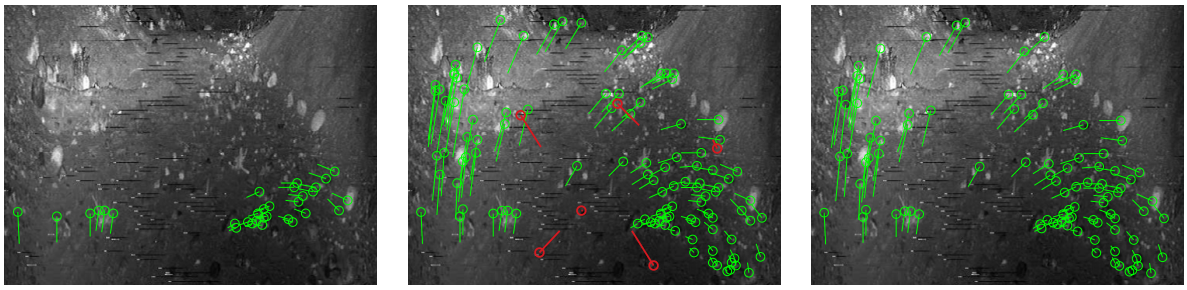
With respect to the conditions under which (4.25) has a unique solution, it follows from (4.31) that $\det \boldsymbol{\mathcal{M}} = w \det \mathbf{I}$. Given that $\Delta t_m^2 > 0$, we know $w > 0$, and therefore the unique solution of $\delta\boldsymbol{\varpi}$ requires only that $\det \mathbf{I} \neq \mathbf{0}$. For this to be true, it is sufficient to show that $\mathbf{I}$ is *positive definite*, which implies that for any $\mathbf{x} \neq \mathbf{0}$, we have that $\mathbf{x}^T \mathbf{I}\, \mathbf{x} > \mathbf{0}$. Examining the structure of $\mathbf{I}$, we find that

$$\mathbf{x}^T \mathbf{I}\, \mathbf{x} = -\mathbf{x}^T \sum_m \Delta t_m^2 (\mathbf{r}_m - \mathbf{r})^\wedge (\mathbf{r}_m - \mathbf{r})^\wedge \mathbf{x} = \sum_m \Delta t_m^2 \underbrace{((\mathbf{r}_m - \mathbf{r})^\wedge \mathbf{x})^T ((\mathbf{r}_m - \mathbf{r})^\wedge \mathbf{x})}_{\geq 0} \geq 0. \tag{4.33}$$

Given that all terms in the summation are non-negative, the total must also be non-negative. In order for the total to be zero, each term of the summation must be zero, meaning that for each m, either $\mathbf{r}_m = \mathbf{r}$, $\mathbf{x}$ is parallel to $\mathbf{r}_m - \mathbf{r}$, or $\mathbf{x} = \mathbf{0}$. Based on the first two cases (the last case is not true by assumption), the sufficient conditions to ensure a unique solution are: (i) that there is a minimum of three points, and (ii) that the three points are not collinear. In contrast to a typical rigid formulation, we note that this non-collinearity constraint happens in a *virtual* reference frame, into which the points $\boldsymbol{p}_{m,1}$ are transformed based on the current velocity estimate, $\bar{\boldsymbol{\varpi}}$, and the different time intervals, Δt_m; interestingly, this means that the system could be solvable using three points that are collinear in *world space*, given a non-zero sensor velocity.

4.7 Appearance-Based Lidar Experiment

In this section, we validate our motion-compensated RANSAC algorithm (*MC RANSAC*), and its heuristic variant (*MC-Fast RANSAC*), using the appearance-based lidar pipeline



(a) Rigid RANSAC (b) Rigid RANSAC (loose α) (c) MC RANSAC

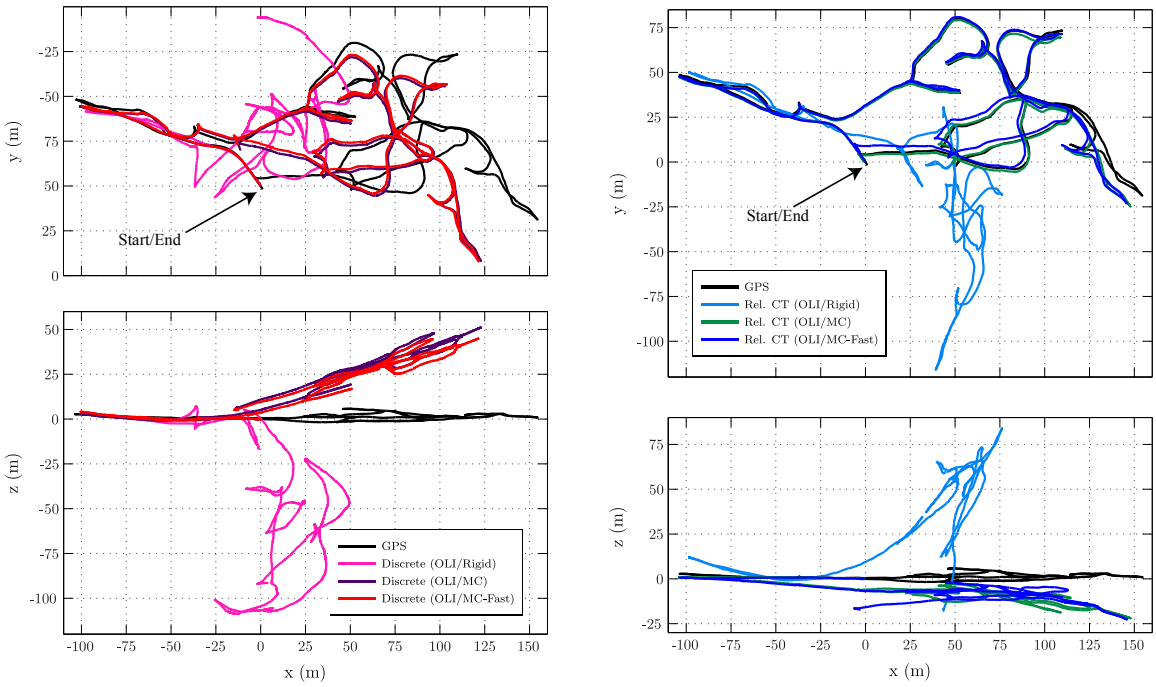
Figure 4.1: This figure shows the inlying data associations after applying various configurations of the RANSAC algorithm. Note that both (a) and (b) used a classic, rigid RANSAC model, while (c) uses our motion-compensated RANSAC algorithm. The difference between the results in (a) and (b) is due to the strictness of the inlier threshold, α. Note that (a) and (c) use the same tight inlier threshold, while (b) uses a relaxed threshold that allows for a larger number of inlying matches (green), but also introduces false positives (red outliers). By using a constant velocity to model the motion of the sensor, our motion-compensated outlier rejection scheme, seen in (c), is able to better account for the distortion of the image.

presented in Chapter 2 and the experimental setup described in Appendix A. In contrast to a typical passive camera which produces imagery at a rate of 15-30 Hz, our lidar sensor produces intensity imagery at a rate of only 2 Hz, and therefore is susceptible to large amounts of motion-distortion.

4.7.1 Quality

Here, we take three approaches to validating the quality of the proposed outlier-rejection schemes. The first is a simple visual inspection of the typical feature-track output of our motion-compensated RANSAC algorithm, as seen in Figure 4.1(c). During this sequence, the robot was translating forward while exhibiting a roll-type motion, due to the rough terrain. The quality of these feature tracks are impressive when contrasted against the results of the rigid model (shown in Figure 4.1(a) and 4.1(b)) – this is especially true when we consider that the threshold, α, is the same as the one used in Figure 4.1(a).

The second method used to evaluate the quality of outlier rejection is the output of both a discrete and a continuous-time SLAM algorithm (recalling the methods described in Chapters 3). Note that the specific implementation of continuous-time SLAM used for this result will be described in Chapter 5. The output of each RANSAC algorithm was used to initialize a set of open-loop feature correspondences. Sliding-window-style batch estimators were then run over each set of matches to produce an odometry estimate that



(a) Discrete-Time SLAM

(b) Continuous-Time SLAM (i.e., STEAM)

Figure 4.2: Top and side views of the robot trajectory over the whole 1.1km dataset. DGPS tracks are in black, and the open-loop integration (OLI) of the discrete and continuous-time relative bundle adjustment solutions are shown in different colours for various RANSAC outlier rejection schemes. Note that in order to make the discrete bundle adjustment state tractable and solvable, we assume no motion distortion in the images.

can be compared to GPS ground truth. The output of the SLAM algorithms can be seen in Figure 4.2; the important result to note is the dramatic performance improvement between solutions using rigid and motion-compensated outlier rejection schemes. Due to the 'slow' vertical scan of the lidar, the rigid RANSAC algorithm tends to find horizontal bands of temporally similar features; without a strong distribution of features in the vertical direction, it is expected that the pitch parameter of the estimation will suffer.

Lastly, we wish to evaluate the quantity of inliers that the algorithm is able to identify. Although we do not know the true number of inliers between each pair of images, we can contrast algorithms by comparing the distribution of filtered feature correspondences against the initial distribution of possible feature matches. Assuming a relatively constant ratio between inliers and outliers across images, it is expected that the distributions should have a similar shape to the initial set, but a lower mean number of feature correspondences. The results of this distribution comparison can be seen in Figure 4.3. Note that both the motion-compensated and fast motion-compensated algorithms produce a near-identical

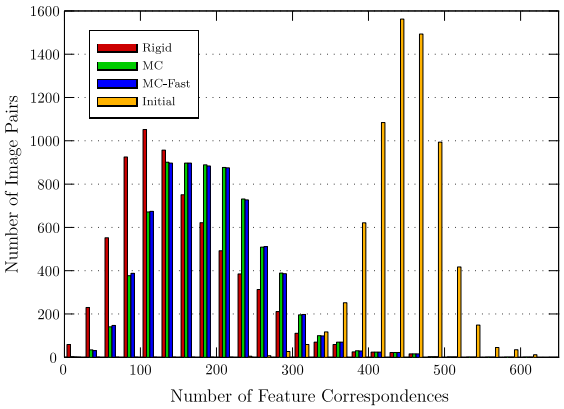


Figure 4.3: This figure shows the distribution of sequential image pairs over the number of successfully matched feature measurements. The distribution of 'initial' matches indicates the total number of hypothesized matches before filtering (including outliers). The rigid, motion-compensated, and fast motion-compensated filters are then applied to generate the plotted distribution of qualified inliers.

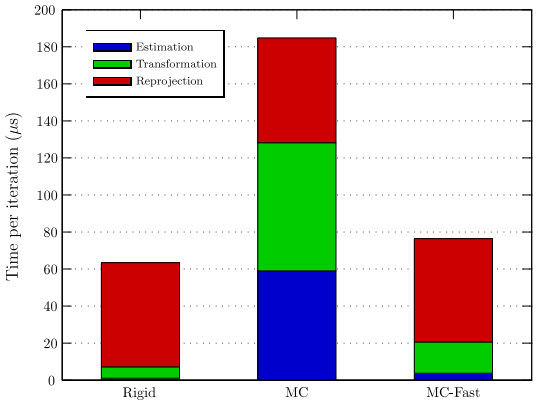


Figure 4.4: This figure shows the computational effort for each RANSAC iteration. The timing is broken down into the three major pieces of the algorithm: estimating the mathematical model (given 3 points), generating and applying the transform provided by the model, and reprojecting the Euclidean points through the spherical camera model. All computations were performed in C++ and timed with a 2.4GHz processor.

result, which, as expected, is distinctly better than the rigid RANSAC algorithm's output.

4.7.2 Computational Efficiency

In order to characterize the computational efficiency of each of these algorithms, the time spent completing each major RANSAC step was recorded over 1500 iterations, for each of the 6880 frames and averaged, as seen in Figure 4.4. The iteration time is broken down into three major sections: the time taken to estimate the mathematical model, the time it takes to generate and apply the transforms, $\mathbf{T}_m$, to each measurement, and the time it takes to reproject the Euclidean points into spherical coordinates.

The classic rigid RANSAC algorithm is the fastest, as expected; however, quantitative analysis suggests that the rigid algorithm is completely unfit for use with this sensor. Moving to the motion-compensated RANSAC algorithm, the large increase in estimation time is due to the addition of the iterative Gauss-Newton process. The increase in transformation time is due to the cost of calculating a transformation matrix for each measurement pair. The fast motion-compensated RANSAC algorithm shows a drastic reduction in estimation time, although still not as fast as the rigid algorithm. For

the transformation step, eight discrete transformations were used to approximate the constant-velocity model, providing a significant speed up, with minimal cost to quality.

4.8 Summary and Conclusions

In this chapter we have proposed a novel adaptation of the standard vision-based RANSAC algorithm to perform outlier rejection on data associations extracted from motion-distorted imagery; leveraging both the geometric and temporal measurement data, our algorithm models the sensor motion as a generalized 6D constant velocity. Using the proposed constant-velocity model, rather than a rigid transformation model, our algorithm is able to account for the temporal nature of scanning-type sensors and compensate for motion distortion in rolling-shutter-type imagery. To improve computational performance, we propose a variant of the algorithm that uses a Euclidean least-squares estimator and a heuristic for the point-transformation step. To validate the approach, each algorithm was applied to a sequence of 6880 lidar intensity/range scans acquired over a 1.1km trajectory. The methods and results presented in this chapter have previously appeared in both Anderson and Barfoot (2013a) and Anderson et al. (2015b).

In summary, the contributions of this chapter are:

1. A temporally sensitive adaptation of the vision-based RANSAC algorithm, using a constant-velocity model to account for motion distortion in 3D data.
2. A heuristic extension of the motion-compensated algorithm that improves computational performance without any large sacrifice to quality.
3. Derivation of the conditions for which our RANSAC method can be solved.
4. An experimental validation of the algorithm, using SURF features extracted from a 6880-frame sequence of lidar intensity imagery, captured over a 1.1 kilometer traversal of rough terrain.

We note that this work was vital in producing the reliable data associations that will be used for the appearance-based lidar experiments in both Chapters 5 and 6. Furthermore, our method was also employed by Tong et al. (2014) to produce feature tracks in their similar lidar-based VO pipeline. Specifically, Tong et al. (2014) used a SICK LMS511 lidar mounted on a panning unit; due to the drastically lower acquisition rate of this lidar, they experienced even more motion distortion.

Chapter 5

Basis Function Representations for Trajectories

This chapter describes a novel approach to parametric, continuous-time trajectory estimation, in which we adopt the *relative coordinate* formulation of SLAM (described in Section 3.3.2). To facilitate the creation of a continuous-time, spatio-temporal pose graph, the temporal portion of the discrete-time relative approach (which uses a kinematic chain of relative pose estimates) is transitioned into continuous-time by estimating the body-centric velocity profile of the robot; this profile can be thought of as a kinematic chain with an infinite number of relative pose changes. Similar to the discrete solution, the spatial linkages (i.e., loop closures) remain as relative transformations that describe the pose change between the robot at two (distant) discrete times. Although (arguably) not as simple as a privileged coordinate frame solution, the advantage in using relative coordinates is that a window-style filter can be used to incrementally solve the problem in constant-time, even at loop closure. Furthermore, the resulting system is *locally* accurate, singularity-free, and avoids global consistency issues.

In order to use the Gauss-Newton optimization framework described in Section 3.1.2, we introduce a discrete set of state variables that parameterize the continuous-time velocity profile. Specifically, we adopt the approach of Furgale et al. (2012), which uses a weighted sum of cubic B-spline basis functions. However, unlike Furgale's original three-dimensional state parameterization, our velocity-based representation is *singularity free*. Using basis functions for interpolation provides an effective means to include high-rate data, such as IMU measurements, or individually timestamped visual features, without negatively impacting state size. Our algorithm is validated by using a 1.1 km lidar dataset (presented

in Appendix A) for lidar-based VO and SLAM (as described in Chapter 2).

The novel contributions of this chapter are: (i) a continuous-time adaptation of the relative-coordinate formulation of SLAM, that estimates the velocity profile of the robot and relative loop closures, (ii) an explanation of sliding-window-style estimation with basis functions, and (iii) an evaluation of the algorithm using lidar-style VO and appearance-based place recognition. These contributions have appeared in two previous publications. Early (open-loop) results from the continuous-time relative coordinate formulation were first published in the proceedings of a full-paper refereed conference (Anderson and Barfoot, 2013b). Later work then used the place-recognition results in Appendix A to estimate large-scale loop closures on the spatio-temporal manifold; combined with the contributions from Chapter 4, these results were later published in a journal article (Anderson et al., 2015b).

5.1 Related Work

Owing to the practicality, effectiveness, and ease of using temporal basis functions as a continuous-time parameterization, these types of parametric representations have dominated the early implementations of continuous-time SLAM. The work of Furgale et al. (2012, 2015), which formally derives the general continuous-time SLAM problem, also demonstrates the use of temporal basis functions to find a parametric solution for a typical camera-IMU calibration problem. The use of uniformly spaced cubic B-splines to parameterize a robot trajectory can also be seen in the estimation schemes previously proposed by Bibby and Reid (2010) and Fleps et al. (2011). Lovegrove et al. (2013) formulate a methodology for visual-inertial fusion very similar to the work by Furgale et al. (2012), but parameterize rotations using the cumulative B-spline representation introduced in the computer animation field by Kim et al. (1995). Although the connection was not drawn by Lovegrove, their method shares similar properties to our previously published work (Anderson and Barfoot, 2013b); both systems are singularity-free, and perform local estimations that are related to the velocity domain. With an eye towards spacecraft attitude estimation, Sommer et al. (2015) present an estimation method for using B-spline curves on unit-length quaternions. The parametric works by Oth et al. (2013) and Anderson et al. (2014) explore the issue of *temporal resolution* by investigating how parameterizations can be chosen adaptively to prevent underfitting and overfitting the measurements. Finally, in an interesting fusion of state parameterizations, the most

recent adaptation of the mining work by Zlot and Bosse (2014) uses low-frequency cubic B-spline perturbations to update a high-frequency set of discrete pose variables.

A few notable algorithms adapt the traditional discrete-time SLAM formulation to a fundamental form of STEAM by applying pose-interpolation techniques between *keyframes*; although not explicitly shown, these works have parametric continuous-time SLAM interpretations, which solve the associated maximum likelihood problems. One such work, by Hedborg et al. (2012), demonstrates the use of inexpensive rolling-shutter cameras to perform full nonlinear bundle adjustment using SLERP (spherical linear interpolation). Using appearance-based lidar data in an identical fashion to this research, Dong and Barfoot (2012) perform frame-to-frame motion-compensated visual odometry with a piece-wise, constant-velocity model. Finally, using dense range data from a continuously spinning 2D lidar, Bosse and Zlot (2009) have developed a technique, akin to the iterative closest point (ICP) algorithm, that is capable of iteratively correcting point clouds for motion distortion. Adaptations of their work using surfels have been successful in mapping a large underground mine (Zlot and Bosse, 2012) and estimating the irregular motion of a 2D lidar attached to the end of a spring (Bosse et al., 2012).

5.2 Background – Parametric Batch Estimation

Recalling that a direct representation of the continuous-time robot trajectory, $\mathbf{x}(t)$, would require an *infinite* number of states (as discussed in the STEAM problem derivation in Section 3.4), we now introduce the parametric method to trajectory representation proposed by Furgale et al. (2012). A *finite* state vector is formed by leveraging a sum of weighted (known) temporal basis functions,

$$\mathbf{x}(t) := \boldsymbol{\Psi}(t)\mathbf{c}, \qquad \boldsymbol{\Psi}(t) := \begin{bmatrix} \boldsymbol{\psi}_1(t) & \boldsymbol{\psi}_2(t) & \cdots & \boldsymbol{\psi}_{D\times B}(t) \end{bmatrix}, \tag{5.1}$$

where we have B temporal basis functions in each of the D trajectory dimensions, and each basis function, $\boldsymbol{\psi}_b(t)$, is a $D \times 1$ column vector. Since the form of $\boldsymbol{\Psi}(t)$ is chosen, it can be queried for the value of $\mathbf{x}(t)$ at any time, using the coefficients, $\mathbf{c}$; note that the basis functions' coefficients, $\mathbf{c}$, are the free parameters that can be used to vary the trajectory estimate, $\mathbf{x}(t)$. Furthermore, depending on our choice of $\boldsymbol{\Psi}(t)$ we also have access to analytical derivatives, $\dot{\mathbf{x}}(t) = \dot{\boldsymbol{\Psi}}(t)\mathbf{c}$. An important result of the parametric representation is that the time-varying component, $\boldsymbol{\Psi}(t)$, is separated from the variables that we wish to estimate, $\mathbf{c}$; this separability is necessary in order to formulate a typical

nonlinear optimization problem with a set of discrete, time-invariant state variables.

Without even choosing a specific form for $\mathbf{\Psi}(t)$, it becomes trivial to draw connections to the Gauss-Newton algorithm described in Section 3.1.2. Defining the joint state vector, $\mathbf{z} = \begin{bmatrix} \mathbf{c}^T & \boldsymbol{\ell}^T \end{bmatrix}^T$, and noting that,

$$\hat{\mathbf{x}}(t) = \mathbf{\Psi}(t)\hat{\mathbf{c}}, \tag{5.2}$$

we are able to use the usual *additive* perturbation scheme,

$$\hat{\mathbf{z}} = \bar{\mathbf{z}} + \delta\mathbf{z}, \quad \hat{\mathbf{x}}(t) = \mathbf{\Psi}(t)(\bar{\mathbf{c}} + \delta\mathbf{c}), \tag{5.3}$$

to linearize prior and measurement errors, where $\bar{\mathbf{z}}$ is our operating point, or best guess, and $\delta\mathbf{z}$ is the unknown perturbation that we solve with Gauss-Newton. Using the motion model error presented in (3.68b) as a demonstration,

$$\begin{aligned} \mathbf{e}_u(t, \mathbf{z}) &= \dot{\mathbf{x}}(t) - \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ &= \dot{\mathbf{\Psi}}(t)\mathbf{c} - \mathbf{f}(\mathbf{\Psi}(t)\mathbf{c}, \mathbf{u}(t)), \end{aligned} \tag{5.4}$$

we substitute our perturbations to find

$$\begin{aligned} \mathbf{e}_u(t, \bar{\mathbf{z}} + \delta\mathbf{z}) &= \dot{\mathbf{\Psi}}(t)(\bar{\mathbf{c}} + \delta\mathbf{c}) - \mathbf{f}(\mathbf{\Psi}(t)(\bar{\mathbf{c}} + \delta\mathbf{c}), \mathbf{u}(t)) \\ &\approx \underbrace{\dot{\mathbf{\Psi}}(t)\bar{\mathbf{c}} - \mathbf{f}(\mathbf{\Psi}(t)(\bar{\mathbf{c}}), \mathbf{u}(t))}_{\mathbf{e}_u(t, \bar{\mathbf{z}})} - \underbrace{\begin{bmatrix} \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\big|_{\mathbf{\Psi}(t)\bar{\mathbf{c}}, \mathbf{u}(t)} \mathbf{\Psi}(t) - \dot{\mathbf{\Psi}}(t) & \mathbf{0} \end{bmatrix}}_{\mathbf{F}(t)} \delta\mathbf{z}. \end{aligned} \tag{5.5}$$

Following from the continuous-time prior cost, $J_p(\mathbf{x}(t))$, in (3.67), and excluding the term related to initial position[1], it is straight-forward to see how the *prior portion* of the Gauss-Newton update equation, in (3.25), becomes

$$\underbrace{\int_{t_0}^{t_K} \mathbf{F}(\tau)^T \mathbf{Q}_C^{-1} \mathbf{F}(\tau) d\tau}_{\mathbf{A}_{\text{pri}}} \delta\mathbf{z}^\star = \underbrace{\int_{t_0}^{t_K} \mathbf{F}(\tau)^T \mathbf{Q}_C^{-1} \mathbf{e}_u(\tau, \bar{\mathbf{z}}) d\tau}_{\mathbf{b}_{\text{pri}}}, \tag{5.6}$$

where $\mathbf{A}_{\text{pri}}$ and $\mathbf{b}_{\text{pri}}$ may have analytical solutions depending on $\mathbf{\Psi}(t)$, $\mathbf{f}(\cdot)$, and $\mathbf{u}(t)$.

The specific form of temporal basis function used by Furgale et al. (2012), and by the algorithm we present in this chapter, is the cubic B-spline basis function (see Figure 5.1).

[1] Although we ignore the prior cost on initial position in this example, we note that it is trivial to include. Furthermore, we note that using robo-centric coordinates for SLAM is quite common; in this formulation the robot pose (at all times) is estimated with respect to its own initial position (rather than an inertial frame) and prior information about the global robot pose becomes irrelevant.

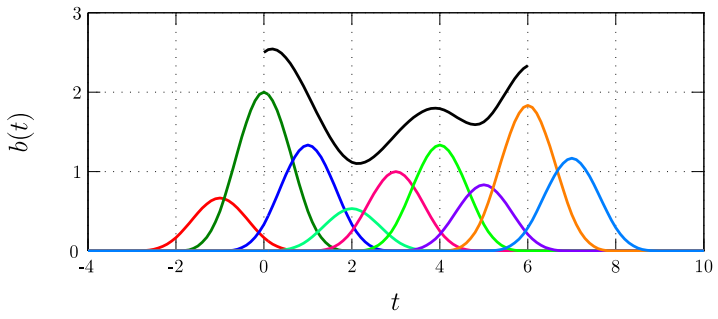


Figure 5.1: This figure illustrates our use of weighted temporal basis functions in one dimension; the example has nine uniformly spaced cubic B-spline basis functions that are used to define the weighted sum (black) over six time segments. One can imagine varying the shape of the (black) sum by scaling the weights (i.e., heights) of the underlying B-splines. A single cubic B-spline is a smooth piecewise polynomial defined over *four* time segments; therefore, only four of the basis functions have a nonzero contribution to the sum at any one *query time* – this is known as *local support*. The points in time that separate two time segments are known as *knots*.

The use of B-spline basis functions as a parametric representation is common due to their good representational power, easy analytical derivatives/integrals, and *local support* property; local support is desirable to our STEAM framework as it helps to enable incremental estimation. Using cubic B-splines, with only four basis functions non-zero at any time, we note that $\boldsymbol{\Psi}(t)$, for the b^{th} time segment, $u_b \leq t \leq u_{b+1}$, can be written

$$\boldsymbol{\Psi}(t) = \begin{bmatrix} \mathbf{u}_b(t)^T \mathbf{B} \mathbf{P}_{1,b} \\ \vdots \\ \mathbf{u}_b(t)^T \mathbf{B} \mathbf{P}_{d,b} \\ \vdots \\ \mathbf{u}_b(t)^T \mathbf{B} \mathbf{P}_{D,b} \end{bmatrix}, \quad \mathbf{u}_b(t) = \begin{bmatrix} 1 \\ (\frac{t-u_b}{u_{b+1}-u_b}) \\ (\frac{t-u_b}{u_{b+1}-u_b})^2 \\ (\frac{t-u_b}{u_{b+1}-u_b})^3 \end{bmatrix}, \quad \mathbf{B} = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}, \tag{5.7}$$

where $\mathbf{u}_b(t)$ is a cubic interpolation function based on the adjacent knot times, u_b and u_{b+1}, $\mathbf{B}$ is a fixed coefficient matrix, and $\mathbf{P}_{d,b} \in \mathbb{R}^{4\times(D\times B)}$ is a projection matrix that picks the relevant four coefficients (for dimension d and time segment b) out of state vector $\mathbf{c}$.

In order to parameterize the non-commutative transformations required for 6D pose estimation, Furgale et al. (2012) chose to directly estimate the Cayley-Gibbs-Rodrigues axis-angle parameterization, $\boldsymbol{\varphi}(t) \in \mathbb{R}^3$, and the translation vector, $\mathbf{r}(t) \in \mathbb{R}^3$, such that

$$\mathbf{T}(\mathbf{x}(t)) := \begin{bmatrix} \mathbf{C}(\boldsymbol{\varphi}(t)) & \mathbf{r}(t) \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{x}(t) := \begin{bmatrix} \mathbf{r}(t) \\ \boldsymbol{\varphi}(t) \end{bmatrix} = \boldsymbol{\Psi}(t)\mathbf{c}, \tag{5.8}$$

where $\mathbf{C}(\cdot)$ and $\mathbf{T}(\cdot)$ are mappings to $SO(3)$ and $SE(3)$, respectively. Although this representation was successful in allowing Furgale et al. (2012) to perform camera/IMU

calibration with a hugely compressed state vector, we note that it is not singularity-free and far less useful in a free range of motions.

5.3 Relative Coordinate Formulation Using Velocity

In this section, we describe and formulate the relative, continuous-time SLAM problem. To attain the desirable properties of both the relative and continuous-time SLAM formulations, it is proposed that the trajectory be represented by the robot velocity, expressed in the local robot coordinate frame. The intuition behind this decision is derived from the nature of both the continuous-time and relative formulations. Consider a discrete-time system in which we sample the robot's pose at several times and then construct the series of relative pose changes that make up that trajectory. Now, if we consider reducing that sampling period to an infinitesimal level, we note that the series of infinitesimal relative pose changes is in essence the robot-oriented velocity. Recovering a relative pose change between any two times is simply a matter of integrating the velocity information between them. Similar to the typical visual SLAM problem, our goal is to find the optimal solution for the robot trajectory and the map parameters.

5.3.1 Velocity Profile

To define the velocity profile, we begin by introducing the coordinate frames $\underrightarrow{\mathcal{F}}_i$, our inertial reference frame, and $\underrightarrow{\mathcal{F}}_s(t)$, the robot's instantaneous sensor frame. In the relative, continuous-time formulation of the problem, our algorithm aims to estimate the generalized six-dimensional (6D) velocity of the robot,

$$\boldsymbol{\varpi}(t) := \begin{bmatrix} \boldsymbol{\nu}_t^{t,i} \\ \boldsymbol{\omega}_t^{t,i} \end{bmatrix}, \tag{5.9}$$

which is expressed in the robot's sensor frame over the time interval $t = [0, T]$, where $\boldsymbol{\nu}_t^{t,i}$ and $\boldsymbol{\omega}_t^{t,i}$ are the linear and angular components. Note that for simplicity, all references to the time-varying sensor frame, $\underrightarrow{\mathcal{F}}_s(t)$, have been replaced with the timestamp that we wish to reference (e.g., time t in the velocity, $\boldsymbol{\nu}_t^{t,i}$). Notably, an advantage of the velocity profile trajectory representation (in contrast to using rotations or transformations) is that there are no constraints that need to be enforced on the parameters of the solution.

5.3.2 Kinematics

In order to include measurements that depend on the pose, or pose change, of the robot, in a batch estimation scheme, we must first derive the relationship between the robot velocity, $\boldsymbol{\varpi}(t)$, and the robot pose, $\mathbf{T}(t)$, at time t. Following the discussion of rotations and transformations in Section 3.2.1, we note that the well-known kinematic equation relating angular velocity to rotation is

$$\dot{\mathbf{C}}_{t,i} = \boldsymbol{\omega}_t^{t,i\wedge}\mathbf{C}_{t,i}, \tag{5.10}$$

where $(\cdot)^\wedge$ is the skew-symmetric operator in (3.35). Recalling the transformation matrix definition, in (3.36), and expanding the time derivative of $\mathbf{r}_t^{i,t}$, we find that

$$\dot{\mathbf{r}}_t^{i,t} = -\frac{d}{dt}(\mathbf{C}_{t,i}\mathbf{r}_i^{t,i}) = -\dot{\mathbf{C}}_{t,i}\mathbf{r}_i^{t,i} - \mathbf{C}_{t,i}\dot{\mathbf{r}}_i^{t,i} = -\boldsymbol{\omega}_t^{t,i\wedge}\mathbf{C}_{t,i}(-\mathbf{r}_i^{i,t}) - \mathbf{C}_{t,i}\dot{\mathbf{r}}_i^{t,i} = \boldsymbol{\omega}_t^{t,i\wedge}\mathbf{r}_t^{i,t} + \boldsymbol{\nu}_t^{t,i}, \tag{5.11}$$

where we define $\boldsymbol{\nu}_t^{t,i} := -\mathbf{C}_{t,i}\dot{\mathbf{r}}_i^{t,i}$. Substituting our kinematic equations, (5.10) and (5.11), into the time derivative of the transformation matrix,

$$\dot{\mathbf{T}}_{t,i} = \begin{bmatrix} \dot{\mathbf{C}}_{t,i} & \dot{\mathbf{r}}_t^{i,t} \\ \mathbf{0}^T & 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega}_t^{t,i\wedge}\mathbf{C}_{t,i} & \boldsymbol{\omega}_t^{t,i\wedge}\mathbf{r}_t^{i,t} + \boldsymbol{\nu}_t^{t,i} \\ \mathbf{0}^T & 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega}_t^{t,i\wedge} & \boldsymbol{\nu}_t^{t,i} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{C}_{t,i} & \mathbf{r}_t^{i,t} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{5.12}$$

we derive a nonlinear kinematic equation that relates pose change to $\boldsymbol{\varpi}(t)$:

$$\dot{\mathbf{T}}_{t,i} = \boldsymbol{\varpi}(t)^\wedge\mathbf{T}_{t,i}. \tag{5.13}$$

As we are mostly interested in the pose change of the robot between two non-zero times, t_a and t_b, we aim to find a general expression involving the matrix $\mathbf{T}_{t,t_a}$. In order to replace the inertial reference frame, $\underrightarrow{\mathcal{F}}_i$, in (5.13) with the sensor frame at time t_a, $\underrightarrow{\mathcal{F}}_s(t_a)$, we begin with the expression,

$$\mathbf{T}_{t,i} = \mathbf{T}_{t,t_a}\mathbf{T}_{t_a,i}, \tag{5.14}$$

and take the derivative,

$$\dot{\mathbf{T}}_{t,i} = \dot{\mathbf{T}}_{t,t_a}\mathbf{T}_{t_a,i} + \underbrace{\mathbf{T}_{t,t_a}\dot{\mathbf{T}}_{t_a,i}}_{\text{zero}} = \dot{\mathbf{T}}_{t,t_a}\mathbf{T}_{t_a,i}, \tag{5.15}$$

noting that $\mathbf{T}_{t,t_a}\dot{\mathbf{T}}_{t_a,i}$ is zero due to the fact that $\mathbf{T}_{t_a,i}$ does not depend on t. By manipulating (5.13) and (5.15), we find the expected result,

$$\dot{\mathbf{T}}_{t,t_a} = \boldsymbol{\varpi}(t)^\wedge\mathbf{T}_{t,t_a}. \tag{5.16}$$

In order to find the relationship between a pose change, $\mathbf{T}_{t_b,t_a}$, and the robot velocity, $\boldsymbol{\varpi}(t)$, we must integrate this *non-commutative* differential equation over the interval $[t_a, t_b]$; notably, this integration must be performed using a method that preserves the properties of the transformation matrix. Recalling the exponential map, $\mathbf{T} = \exp(\boldsymbol{\xi}^\wedge)$, from Section 3.2.1, we note that for a constant velocity, $\boldsymbol{\varpi}$, we have the relationship $\boldsymbol{\xi} = \Delta t \cdot \boldsymbol{\varpi}$. Hence, the initial value problem in (5.16), where $\mathbf{T}_{t_a,t_a} = \mathbf{1}$, can be solved using the *ordered exponential* (i.e., product integral of infinitesimal exponentials):

$$\begin{aligned}\mathbf{T}_{t_b,t_a} &= \prod_{t_a}^{t_b} \exp(\boldsymbol{\varpi}(\tau)^\wedge d\tau) \\ &\equiv \lim_{\Delta t \to 0} \underbrace{\exp(\Delta t\, \boldsymbol{\varpi}(t_b - \Delta t)^\wedge)}_{\mathbf{T}_{t_b,t_b-\Delta t}} \cdots \underbrace{\exp(\Delta t\, \boldsymbol{\varpi}(t_a + \Delta t)^\wedge)}_{\mathbf{T}_{t_a+2\Delta t,t_a+\Delta t}} \underbrace{\exp(\Delta t\, \boldsymbol{\varpi}(t_a)^\wedge)}_{\mathbf{T}_{t_a+\Delta t,t_a}}.\end{aligned} \tag{5.17}$$

Unfortunately, this product integral must be computed numerically. Note that for very small pose changes, the exponential map can be approximated using

$$\exp(\boldsymbol{\xi}^\wedge) \approx \mathbf{1} + \boldsymbol{\xi}^\wedge. \tag{5.18}$$

Determining an appropriate integration timestep, Δt, is subject to the required accuracy of the integration, the typical length of an integration (in time), and the typical magnitude of $\boldsymbol{\varpi}(t)$; in practice we use a maximum step value of $\Delta t = 0.01$, with smaller steps when we wish to extract the pose at specific intermediate times (e.g., you may want the pose change at several measurement times, spaced less than 0.01 seconds apart).

The integral, presented in (5.17) relates the continuous-time velocity profile, $\boldsymbol{\varpi}(t)$, to a discrete pose change, $\mathbf{T}_{t_b,t_a}$, and is an important result, as it allow us to calculate the value of expected measurements that depend on pose (e.g., visual measurements of an estimated landmark) in our relative continuous-time formulation.

5.3.3 Perturbations to the Kinematics

In order to linearize terms involving a transform integrated from the velocity profile, using (5.17), we extend our discussion of kinematics to consider the effect of small changes to $\boldsymbol{\varpi}(t)$. Using the constraint-sensitive perturbation scheme,

$$\mathbf{T} = \exp(\boldsymbol{\xi}^\wedge) = \exp(\delta\boldsymbol{\xi}^\wedge)\exp(\bar{\boldsymbol{\xi}}^\wedge) \approx (\mathbf{1} + \delta\boldsymbol{\xi}^\wedge)\bar{\mathbf{T}}, \tag{5.19}$$

the kinematic equation, $\dot{\mathbf{T}} = \boldsymbol{\varpi}^\wedge \mathbf{T}$, becomes

$$\frac{d}{dt}((\mathbf{1} + \delta\boldsymbol{\xi}^\wedge)\bar{\mathbf{T}}) \approx (\bar{\boldsymbol{\varpi}} + \delta\boldsymbol{\varpi})^\wedge(\mathbf{1} + \delta\boldsymbol{\xi}^\wedge)\bar{\mathbf{T}}. \tag{5.20}$$

By expanding and dropping small terms, we find

$$\dot{\bar{\mathbf{T}}} + \delta\boldsymbol{\xi}^\wedge\dot{\bar{\mathbf{T}}} + \delta\dot{\boldsymbol{\xi}}^\wedge\bar{\mathbf{T}} \approx \bar{\boldsymbol{\varpi}}^\wedge\bar{\mathbf{T}} + \bar{\boldsymbol{\varpi}}^\wedge\delta\boldsymbol{\xi}^\wedge\bar{\mathbf{T}} + \delta\boldsymbol{\varpi}^\wedge\bar{\mathbf{T}}. \tag{5.21}$$

Using the nominal solution, $\dot{\bar{\mathbf{T}}} = \bar{\boldsymbol{\varpi}}^\wedge\bar{\mathbf{T}}$, and the identity, $\boldsymbol{\xi}_1^\wedge\boldsymbol{\xi}_2^\wedge - \boldsymbol{\xi}_2^\wedge\boldsymbol{\xi}_1^\wedge \equiv (\boldsymbol{\xi}_1^\curlywedge\boldsymbol{\xi}_2)^\wedge$, equation (5.21) can be manipulated to find,

$$\begin{aligned}
\delta\boldsymbol{\xi}^\wedge\bar{\boldsymbol{\varpi}}^\wedge\bar{\mathbf{T}} + \delta\dot{\boldsymbol{\xi}}^\wedge\bar{\mathbf{T}} &\approx \bar{\boldsymbol{\varpi}}^\wedge\delta\boldsymbol{\xi}^\wedge\bar{\mathbf{T}} + \delta\boldsymbol{\varpi}^\wedge\bar{\mathbf{T}} \\
\delta\dot{\boldsymbol{\xi}}^\wedge &\approx -\delta\boldsymbol{\xi}^\wedge\bar{\boldsymbol{\varpi}}^\wedge + \bar{\boldsymbol{\varpi}}^\wedge\delta\boldsymbol{\xi}^\wedge + \delta\boldsymbol{\varpi}^\wedge \\
&= (\bar{\boldsymbol{\varpi}}^\curlywedge\delta\boldsymbol{\xi})^\wedge + \delta\boldsymbol{\varpi}^\wedge \\
\delta\dot{\boldsymbol{\xi}} &\approx \bar{\boldsymbol{\varpi}}^\curlywedge\delta\boldsymbol{\xi} + \delta\boldsymbol{\varpi}.
\end{aligned} \tag{5.22}$$

Giving us the nominal and perturbed equations:

$$\dot{\bar{\mathbf{T}}}(t) = \bar{\boldsymbol{\varpi}}(t)^\wedge\bar{\mathbf{T}}(t), \tag{5.23a}$$

$$\delta\dot{\boldsymbol{\xi}}(t) = \bar{\boldsymbol{\varpi}}(t)^\curlywedge\delta\boldsymbol{\xi}(t) + \delta\boldsymbol{\varpi}(t), \tag{5.23b}$$

which can be integrated separately and combined to provide the full solutions. We have shown in (5.17) how the nonlinear nominal solution can be integrated numerically. The perturbed equation is a *linear time-varying* equation of the form

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t). \tag{5.24}$$

The general solution to this initial value problem is given by

$$\mathbf{x}(t) = \boldsymbol{\Phi}(t,0)\mathbf{x}(0) + \int_0^t \boldsymbol{\Phi}(t,\tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau, \tag{5.25}$$

where $\boldsymbol{\Phi}(t,\tau)$ is called the *state transition matrix* and satisfies

$$\dot{\boldsymbol{\Phi}}(t,\tau) = \mathbf{A}(t)\boldsymbol{\Phi}(t,\tau), \tag{5.26a}$$

$$\boldsymbol{\Phi}(\tau,\tau) = \mathbf{1}. \tag{5.26b}$$

The state transition matrix always exists and is unique, but it cannot always be found analytically. For our particular perturbed equation, we found that it can be expressed as:

$$\boldsymbol{\Phi}(t,\tau) = \bar{\boldsymbol{\mathcal{T}}}(t)\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}. \tag{5.27}$$

where we recall that $\boldsymbol{\mathcal{T}}$ is the *adjoint* transformation matrix in (3.44). To verify, we check that (5.27) satisfies the required conditions of a state transition matrix:

$$\dot{\boldsymbol{\Phi}}(t,\tau) = \frac{d}{dt}\bar{\boldsymbol{\mathcal{T}}}(t)\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1} = \dot{\bar{\boldsymbol{\mathcal{T}}}}(t)\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1} = \bar{\boldsymbol{\varpi}}(t)^\curlywedge\bar{\boldsymbol{\mathcal{T}}}(t)\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1} = \bar{\boldsymbol{\varpi}}(t)^\curlywedge\boldsymbol{\Phi}(t,\tau), \tag{5.28a}$$

$$\boldsymbol{\Phi}(\tau,\tau) = \bar{\boldsymbol{\mathcal{T}}}(\tau)\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1} = \mathbf{1}, \tag{5.28b}$$

where $\dot{\boldsymbol{\mathcal{T}}}(t) = \boldsymbol{\varpi}(t)^{\curlywedge}\boldsymbol{\mathcal{T}}(t)$ is the *adjoint* version of $\dot{\mathbf{T}}(t) = \boldsymbol{\varpi}(t)^{\wedge}\mathbf{T}(t)$. Using the general solution to the initial value problem, we have that:

$$\delta\boldsymbol{\xi}(t) = \bar{\boldsymbol{\mathcal{T}}}(t)\bar{\boldsymbol{\mathcal{T}}}(0)^{-1}\delta\boldsymbol{\xi}(0) + \bar{\boldsymbol{\mathcal{T}}}(t)\int_0^t \bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}\delta\boldsymbol{\varpi}(\tau)d\tau. \tag{5.29}$$

To verify again that this is indeed the correct solution, we can differentiate:

$$\begin{aligned}
\delta\dot{\boldsymbol{\xi}}(t) &= \dot{\bar{\boldsymbol{\mathcal{T}}}}(t)\bar{\boldsymbol{\mathcal{T}}}(0)^{-1}\delta\boldsymbol{\xi}(0) + \dot{\bar{\boldsymbol{\mathcal{T}}}}(t)\int_0^t \bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}\delta\boldsymbol{\varpi}(\tau)d\tau + \bar{\boldsymbol{\mathcal{T}}}(t)\bar{\boldsymbol{\mathcal{T}}}(t)^{-1}\delta\boldsymbol{\varpi}(t) \\
&= \bar{\boldsymbol{\varpi}}(t)^{\curlywedge}\underbrace{\left(\bar{\boldsymbol{\mathcal{T}}}(t)\bar{\boldsymbol{\mathcal{T}}}(0)^{-1}\delta\boldsymbol{\xi}(0) + \bar{\boldsymbol{\mathcal{T}}}(t)\int_0^t \bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}\delta\boldsymbol{\varpi}(\tau)d\tau\right)}_{\delta\boldsymbol{\xi}(t)} + \delta\boldsymbol{\varpi}(t) \\
&= \bar{\boldsymbol{\varpi}}(t)^{\curlywedge}\delta\boldsymbol{\xi}(t) + \delta\boldsymbol{\varpi}(t).
\end{aligned} \tag{5.30}$$

We now wish to consider the extension of (5.29) to a general pose change, $\mathbf{T}_{t_b,t_a}$, between two non-zero times, t_a and t_b. We begin by noting,

$$\mathbf{T}_{t_b,t_a} = \exp(\delta\boldsymbol{\xi}_{t_b,t_a}^{\wedge})\bar{\mathbf{T}}_{t_b,t_a} \approx \left(\mathbf{1} + \delta\boldsymbol{\xi}_{t_b,t_a}^{\wedge}\right)\bar{\mathbf{T}}_{t_b,t_a}. \tag{5.31}$$

Using the identity, $\mathbf{T}\exp(\mathbf{w}^{\wedge}) \equiv \exp((\boldsymbol{\mathcal{T}}\mathbf{w})^{\wedge})\mathbf{T}$, we then derive that,

$$\begin{aligned}
\mathbf{T}_{t_b,t_a} &= \mathbf{T}(t_b)\mathbf{T}(t_a)^{-1} \\
&= \exp(\delta\boldsymbol{\xi}(t_b)^{\wedge})\bar{\mathbf{T}}(t_b)\left(\exp(\delta\boldsymbol{\xi}(t_a)^{\wedge})\bar{\mathbf{T}}(t_a)\right)^{-1} \\
&= \exp(\delta\boldsymbol{\xi}(t_b)^{\wedge})\bar{\mathbf{T}}(t_b)\bar{\mathbf{T}}(t_a)^{-1}\exp(-\delta\boldsymbol{\xi}(t_a)^{\wedge}) \\
&= \exp(\delta\boldsymbol{\xi}(t_b)^{\wedge})\exp((-\bar{\boldsymbol{\mathcal{T}}}(t_b)\bar{\boldsymbol{\mathcal{T}}}(t_a)^{-1}\delta\boldsymbol{\xi}(t_a))^{\wedge})\bar{\mathbf{T}}_{t_b,t_a} \\
&\approx (\mathbf{1} + \delta\boldsymbol{\xi}(t_b)^{\wedge})\left(\mathbf{1} - (\bar{\boldsymbol{\mathcal{T}}}(t_b)\bar{\boldsymbol{\mathcal{T}}}(t_a)^{-1}\delta\boldsymbol{\xi}(t_a))^{\wedge}\right)\bar{\mathbf{T}}_{t_b,t_a} \\
&\approx \left(\mathbf{1} + (\delta\boldsymbol{\xi}(t_b) - \bar{\boldsymbol{\mathcal{T}}}(t_b)\bar{\boldsymbol{\mathcal{T}}}(t_a)^{-1}\delta\boldsymbol{\xi}(t_a))^{\wedge}\right)\bar{\mathbf{T}}_{t_b,t_a}.
\end{aligned} \tag{5.32}$$

Comparing (5.31) and (5.32), we find the relationship,

$$\delta\boldsymbol{\xi}_{t_b,t_a} \approx \delta\boldsymbol{\xi}(t_b) - \bar{\boldsymbol{\mathcal{T}}}(t_b)\bar{\boldsymbol{\mathcal{T}}}(t_a)^{-1}\delta\boldsymbol{\xi}(t_a). \tag{5.33}$$

To simplify the term $\delta\boldsymbol{\xi}(t_b) - \bar{\boldsymbol{\mathcal{T}}}(t_b)\bar{\boldsymbol{\mathcal{T}}}(t_a)^{-1}\delta\boldsymbol{\xi}(t_a)$, and relate the perturbation back to our velocity profile, $\boldsymbol{\varpi}(t)$, we substitute the solution for $\delta\boldsymbol{\xi}(t)$ found in (5.29):

$$\begin{aligned}
\delta\boldsymbol{\xi}(t_b) - \bar{\boldsymbol{\mathcal{T}}}(t_b)\bar{\boldsymbol{\mathcal{T}}}(t_a)^{-1}\delta\boldsymbol{\xi}(t_a) &= \left(\bar{\boldsymbol{\mathcal{T}}}(t_b)\bar{\boldsymbol{\mathcal{T}}}(0)^{-1}\delta\boldsymbol{\xi}(0) + \bar{\boldsymbol{\mathcal{T}}}(t_b)\int_0^{t_b}\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}\delta\boldsymbol{\varpi}(\tau)d\tau\right) \\
&\quad -\bar{\boldsymbol{\mathcal{T}}}(t_b)\bar{\boldsymbol{\mathcal{T}}}(t_a)^{-1}\left(\bar{\boldsymbol{\mathcal{T}}}(t_a)\bar{\boldsymbol{\mathcal{T}}}(0)^{-1}\delta\boldsymbol{\xi}(0) + \bar{\boldsymbol{\mathcal{T}}}(t_a)\int_0^{t_a}\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}\delta\boldsymbol{\varpi}(\tau)d\tau\right) \\
&= \bar{\boldsymbol{\mathcal{T}}}(t_b)\left(\int_0^{t_b}\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}\delta\boldsymbol{\varpi}(\tau)d\tau - \int_0^{t_a}\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}\delta\boldsymbol{\varpi}(\tau)d\tau\right) \\
&= \int_{t_a}^{t_b}\bar{\boldsymbol{\mathcal{T}}}_{t_b,\tau}\,\delta\boldsymbol{\varpi}(\tau)d\tau.
\end{aligned} \tag{5.34}$$

Substituting (5.34) back into (5.31), we find the following relationship between $\mathbf{T}_{t_b,t_a}$ and small changes in $\boldsymbol{\varpi}(t)$,

$$\mathbf{T}_{t_b,t_a} \approx \left(\mathbf{1} + \left(\int_{t_a}^{t_b} \bar{\boldsymbol{\mathcal{T}}}_{t_b,\tau}\, \delta\boldsymbol{\varpi}(\tau) d\tau\right)^{\wedge}\right) \bar{\mathbf{T}}_{t_b,t_a}. \tag{5.35}$$

Although it is initially unclear how one might isolate the time-varying parameter, $\delta\boldsymbol{\varpi}(t)$, since it is embedded in the integral, the following section will show how this is made manageable with the use of parametric basis functions.

The relationship in (5.35) between a pose change, $\mathbf{T}_{t_b,t_a}$, and a perturbation to the velocity profile, $\delta\boldsymbol{\varpi}(t)$, allows us to use this relative continuous-time formulation with the Gauss-Newton optimization technique described in Section 3.1.2. The nominal pose change, $\bar{\mathbf{T}}_{t_b,t_a}$, is integrated from the nominal velocity profile, $\bar{\boldsymbol{\varpi}}(t)$, using (5.17).

5.3.4 Parametric Batch Estimator

Trajectory Parameterization

In the relative, continuous-time formulation of the SLAM problem, we wish to estimate the 6D velocity of the robot, $\boldsymbol{\varpi}(t)$, which is expressed in the robot's sensor frame, over the time interval $t = [t_0, t_K]$. Moving forward in the derivation of the estimator, we define a parametric representation for the continuous-time state. Similar to the method described in Section 5.2, we use a weighted sum of temporal basis functions,

$$\boldsymbol{\varpi}(t) := \boldsymbol{\Psi}(t)\mathbf{c}, \qquad \boldsymbol{\Psi}(t) := \begin{bmatrix} \boldsymbol{\psi}_1(t) & \boldsymbol{\psi}_2(t) & \cdots & \boldsymbol{\psi}_{6B}(t) \end{bmatrix}, \tag{5.36}$$

where we have B temporal basis functions in each of the six problem dimensions, and each basis function, $\boldsymbol{\psi}_b(t)$, is a 6×1 column vector. The coefficients (i.e., state variables), $\mathbf{c} \in \mathbb{R}^{6B}$, can be used to vary the velocity estimate, $\boldsymbol{\varpi}(t)$; using a parametric representation, the estimation goal is to simply find the optimal values of these weights.

Owing to the requirement that our basis functions have *local support*, which will be discussed later, our implementation uses cubic B-spline basis functions (as seen in Figure 5.1); note our method is not tied to using only cubic B-splines – any basis function that exhibits local support is satisfactory.

Loop Closures

In addition to the relative trajectory, we also wish to estimate any loop-closure transformations between segments of the trajectory from distinctly different times. These loop

closures are stored as 4×4 homogeneous transformation matrices,

$$\mathbf{T}^{\text{LC}}_{t_2,t_1} := \mathbf{T}(t_2)\mathbf{T}(t_1)^{-1}, \tag{5.37}$$

between the sensor frame, $\underrightarrow{\mathcal{F}}_s(t)$, at the two distinct times, t_1 and t_2. As described in Section 3.3.2, these loop-closure transformations over-parameterize the trajectory solution provided by $\boldsymbol{\varpi}(t)$, but allow us to perform loop-closure optimizations using a window-style filter; these optimizations are constant time and generate locally accurate maps.

Landmarks

To complete the traditional SLAM state formulation, we also estimate the positions of the landmark parameters, $\boldsymbol{\ell}_j$. The landmarks are stored using homogeneous coordinates, as described in Section 3.2.2. Adopting the relative formulation of SLAM (with no privileged coordinates frames), it becomes important that each landmark position is stored locally; this is implemented by expressing, $\boldsymbol{\ell}_j$, in frame $\underrightarrow{\mathcal{F}}_s(t_j)$, the sensor pose at time t_j,

$$\boldsymbol{p}_{t_j}^{\ell_j,t_j} := \begin{bmatrix} \boldsymbol{\ell}_j \\ 1 \end{bmatrix} \in \mathbb{R}^4, \qquad \boldsymbol{\ell}_j = \mathbf{P}_{\boldsymbol{\ell},j}\boldsymbol{\ell}, \tag{5.38}$$

where we define t_j to be the first measurement time of the j^{th} landmark, and $\mathbf{P}_{\boldsymbol{\ell},j}$ is a projection matrix that selects the 3D position $\boldsymbol{\ell}_j$ from the state vector $\boldsymbol{\ell}$. In order to simplify the notation, we also denote $t_k := t_{kj}$, the time of measurement $\mathbf{y}_{kj}$.

Optimization Problem

The joint state that we wish to estimate is defined as $\mathbf{z} := \{\boldsymbol{\theta}, \boldsymbol{\ell}\}$, where the state $\boldsymbol{\theta}$ contains parameters related to robot pose, and $\boldsymbol{\ell}$ is the vector of landmark positions. In the open-loop case, $\boldsymbol{\theta} := \mathbf{c}$; however, if a loop closure is part of the optimization problem, then $\boldsymbol{\theta} := \{\mathbf{c}, \mathbf{T}^{\text{LC}}_{t_2,t_1}\}$. Recalling the standard SLAM/STEAM formulations, it follows that our measurement and prior objective functions are

$$J_m(\mathbf{z}) := \frac{1}{2}\sum_{kj} \mathbf{e}_{m_{kj}}^T \mathbf{R}_{kj}^{-1} \mathbf{e}_{m_{kj}}, \qquad J_p(\mathbf{c}) := \frac{1}{2}\int_{t=0}^{T} \mathbf{e}_u(\tau)^T \mathbf{Q}^{-1} \mathbf{e}_u(\tau) d\tau, \tag{5.39}$$

where J_m is a cost associated with the landmark measurements, and J_p is a cost associated with motion (i.e., a motion prior). Note that J_p does not include a prior term related to initial position; our initial uncertainty is zero because all (relative) estimates are performed with respect to the robocentric frame. Also, we note that loop-closure constraints are

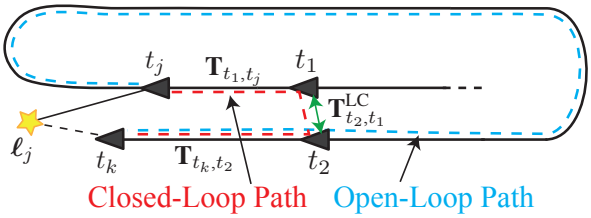


Figure 5.2: In order to formulate an expected measurement, $\mathbf{g}_{kj}(\mathbf{z})$, the point $\boldsymbol{\ell}_j$, which is expressed in the sensor frame at time t_j, needs to be transformed to the current observation time, t_k. In the general open-loop case (seen in blue), the transform, $\mathbf{T}_{t_k,t_j}$, is found by simply integrating the continuous-time velocity profile. However, the existence of loop closures create alternate integration paths (seen in red). Integration of the shortest temporal path is often the most desirable, as it incurs the least estimation error.

introduced in the estimation problem solely through the landmarks that are observed from *both* of the matched trajectory segments, as discussed in the next subsection.

Cost Terms

Recalling the nonlinear observation error from (3.10), and the camera model example from (3.58), we formulate our measurement errors as

$$\mathbf{e}_{m_{kj}}(\mathbf{z}) = \mathbf{y}_{kj} - \mathbf{g}_{kj}(\mathbf{z}), \quad \mathbf{g}_{kj}(\mathbf{z}) := \mathbf{k}(\mathbf{h}_{kj}(\mathbf{z})), \quad \mathbf{h}_{kj}(\mathbf{z}) = \mathbf{T}_{t_k,t_j}\, \boldsymbol{p}_{t_j}^{\ell_j,t_j} \tag{5.40}$$

where $\mathbf{y}_{kj}$ is our measurement, $\mathbf{g}_{kj}(\cdot)$ is the general nonlinear observation model, $\mathbf{k}(\cdot)$ is a nonlinear camera model, $\mathbf{h}_{kj}(\cdot)$ is the transformed homogeneous coordinate, $\mathbf{T}_{t_k,t_j}$ is the 4×4 homogeneous transform matrix that specifies the pose change in the sensor frame between the measurement times t_j and t_k, and $\boldsymbol{p}_{t_j}^{\ell_j,t_j}$ is the position of relative landmark $\boldsymbol{\ell}_j$ in homogeneous coordinates. For our experimental results, using lidar, we model $\mathbf{k}(\cdot)$ as the spherical camera model presented in Appendix Section A.2.

To evaluate this cost using the velocity profile, $\boldsymbol{\varpi}(t)$, we note that in the open-loop case, the unknown transforms, $\mathbf{T}_{t_k,t_j}$, are found by simply integrating over the continuous-time velocity profile. However, in the case that the shortest temporal path between t_j and t_k involves a loop closure, we note that $\mathbf{T}_{t_k,t_j}$ becomes the composition of the following transformation matrices,

$$\mathbf{T}_{t_k,t_j} = \mathbf{T}_{t_k,t_2}\mathbf{T}^{\text{LC}}_{t_2,t_1}\mathbf{T}_{t_1,t_j}, \tag{5.41}$$

where $\mathbf{T}_{t_k,t_2}$ and $\mathbf{T}_{t_1,t_j}$ are integrated using the velocity profile, and $\mathbf{T}^{\text{LC}}_{t_2,t_1}$ exists as a discrete loop-closure transformation between times t_1 and t_2. These alternate integration paths are illustrated in Figure 5.2.

Assuming no knowledge of the control signal, we model the motion as a zero-mean white noise on acceleration. This can be written as the Gaussian process

$$\mathbf{e}_u(t) := \dot{\boldsymbol{\varpi}}(t) = \mathbf{w}(t), \qquad \mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_C \delta(t - t')), \tag{5.42}$$

where $\mathbf{Q}_C$ is the power-spectral-density matrix and $\delta(\cdot)$ is Dirac's delta function.

Using the Kinematic Perturbation

In order to minimize $J(\mathbf{z})$ using our standard Gauss-Newton approach, we first make it quadratic in a small perturbation of the state, $\delta\mathbf{z} := \begin{bmatrix}\delta\boldsymbol{\theta}^T & \delta\boldsymbol{\ell}^T\end{bmatrix}^T$, and then find the optimal state update $\delta\mathbf{z}^\star$ by solving $\frac{\partial^T J}{\partial \delta \mathbf{z}} = \mathbf{0}$; in the open-loop case, $\delta\boldsymbol{\theta} := \delta\mathbf{c}$, and in the closed-loop case, $\delta\boldsymbol{\theta} := \begin{bmatrix}\delta\mathbf{c}^T & \delta\boldsymbol{\xi}_{t_2,t_1}^{\text{LC}\ T}\end{bmatrix}^T$. We note that the linearization of J_p takes the same approach as Furgale et al. (2012), described in Section 5.2, and therefore this section will focus only on the linearization of J_m.

From the temporal basis function definition, $\boldsymbol{\varpi}(t) := \boldsymbol{\Psi}(t)\mathbf{c}$, we can define an additive perturbation model, $\boldsymbol{\varpi}(t) = \bar{\boldsymbol{\varpi}}(t) + \delta\boldsymbol{\varpi}(t)$, such that $\bar{\boldsymbol{\varpi}}(t) + \delta\boldsymbol{\varpi}(t) = \boldsymbol{\Psi}(t)(\bar{\mathbf{c}} + \delta\mathbf{c})$, and therefore $\delta\boldsymbol{\varpi}(t) = \boldsymbol{\Psi}(t)\delta\mathbf{c}$. Substituting this relationship into (5.35), we find that

$$\mathbf{T}_{t_b,t_a} \approx \left(\mathbf{1} + \left(\int_{t_a}^{t_b} \bar{\boldsymbol{\mathcal{T}}}_{t_b,\tau}\boldsymbol{\Psi}(\tau)d\tau\ \delta\mathbf{c}\right)^\wedge\right)\bar{\mathbf{T}}_{t_b,t_a}, \tag{5.43}$$

and therefore our usual perturbation, $\delta\boldsymbol{\xi}_{t_b,t_a}$, to transform $\mathbf{T}_{t_b,t_a}$, is

$$\delta\boldsymbol{\xi}_{t_b,t_a} \approx \left(\int_{t_a}^{t_b} \bar{\boldsymbol{\mathcal{T}}}_{t_b,\tau}\boldsymbol{\Psi}(\tau)d\tau\right)\delta\mathbf{c}. \tag{5.44}$$

Recalling the example camera model linearization in (3.63), we apply our constraint-sensitive perturbations to an *open-loop* trajectory integration to find that

$$\mathbf{e}_{m_{kj}}(\bar{\mathbf{z}} + \delta\mathbf{z}) = \mathbf{y}_{kj} - \mathbf{k}(\bar{\mathbf{T}}_{t_k,t_j}\bar{\boldsymbol{p}}_{t_j}^{\ell_j,t_j}) - \mathbf{G}_{kj}\delta\mathbf{z}, \tag{5.45}$$

$$\mathbf{G}_{kj} := \begin{bmatrix}\mathbf{K}_{kj}\mathbf{H}_{\mathbf{x},kj}\mathbf{H}_{\mathbf{c},kj} & \mathbf{K}_{kj}\mathbf{H}_{\boldsymbol{\ell},kj}\mathbf{P}_{\boldsymbol{\ell},j}\end{bmatrix}, \tag{5.46}$$

where $\mathbf{K}_{kj}$, $\mathbf{H}_{\mathbf{x},kj}$, and $\mathbf{H}_{\boldsymbol{\ell},kj}$ were defined in (3.62) and (3.64), and from (5.44) we have

$$\mathbf{H}_{\mathbf{c},kj} := \int_{t_j}^{t_k} \bar{\boldsymbol{\mathcal{T}}}_{t_k,\tau}\boldsymbol{\Psi}(\tau)d\tau. \tag{5.47}$$

Note that when a loop closure is present in the shortest pose chain from t_j to t_k (i.e., $\mathbf{T}_{t_k,t_j} = \mathbf{T}_{t_k,t_2}\mathbf{T}_{t_2,t_1}^{\text{LC}}\mathbf{T}_{t_1,t_j}$), the formulation changes slightly. Using the same perturbation scheme, with the additional perturbation $\delta\boldsymbol{\xi}_{t_2,t_1}^{\text{LC}}$, it is straightforward to show that

$$\mathbf{G}_{kj} := \begin{bmatrix}\mathbf{K}_{kj}\mathbf{H}_{\mathbf{x},kj}\mathbf{H}_{\mathbf{c},kj} & \mathbf{K}_{kj}\mathbf{H}_{\mathbf{x},kj}\bar{\boldsymbol{\mathcal{T}}}_{t_k,t_2} & \mathbf{K}_{kj}\mathbf{H}_{\boldsymbol{\ell},kj}\mathbf{P}_{\boldsymbol{\ell},j}\end{bmatrix}, \tag{5.48}$$

where $\delta\mathbf{z} = \begin{bmatrix}\delta\mathbf{c}^T & \delta\boldsymbol{\xi}_{t_2,t_1}^{\text{LC}\ T} & \delta\boldsymbol{\ell}^T\end{bmatrix}^T$, and

$$\mathbf{H}_{\mathbf{c},kj} := \left(\int_{t_2}^{t_k} \bar{\boldsymbol{\mathcal{T}}}_{t_k,\tau}\boldsymbol{\Psi}(\tau)d\tau + \int_{t_j}^{t_1} \bar{\boldsymbol{\mathcal{T}}}_{t_k,\tau}\boldsymbol{\Psi}(\tau)d\tau\right). \tag{5.49}$$

Gauss-Newton Algorithm

Finally, connecting our state representation, cost terms, and Jacobians to the Gauss-Newton update equation in (3.25), we have that

$$(\mathbf{A}_{\text{meas}} + \mathbf{A}_{\text{pri}})\,\delta\mathbf{z}^\star = \mathbf{b}_{\text{meas}} + \mathbf{b}_{\text{pri}}. \tag{5.50}$$

where

$$\mathbf{A}_{\text{meas}} := \sum_{kj} \mathbf{G}_{kj}^T\mathbf{R}_{kj}^{-1}\mathbf{G}_{kj}, \qquad \mathbf{b}_{\text{meas}} := \sum_{kj} \mathbf{G}_{kj}^T\mathbf{R}_{kj}^{-1}\mathbf{e}_{m_{kj}}(\bar{\mathbf{z}}), \tag{5.51}$$

and it follows from (5.6) and (5.42) that

$$\mathbf{A}_{\text{pri}} := \begin{bmatrix}\int_0^T \dot{\boldsymbol{\Psi}}(\tau)^T\mathbf{Q}_C^{-1}\dot{\boldsymbol{\Psi}}(\tau)\,d\tau & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix}, \qquad \mathbf{b}_{\text{pri}} := -\mathbf{A}_{\text{pri}}\bar{\mathbf{z}}. \tag{5.52}$$

Note that the term $\mathbf{A}_{\text{pri}}$ only needs to be calculated once, as it does not depend on the state parameters. Following the Gauss-Newton approach, the solution to $\delta\mathbf{z}^\star$ is used to update the nominal solutions iteratively and to convergence. The pertinent state update equations are: $\bar{\mathbf{c}} \leftarrow \bar{\mathbf{c}} + \delta\mathbf{c}^\star$, $\bar{\boldsymbol{\ell}} \leftarrow \bar{\boldsymbol{\ell}} + \delta\boldsymbol{\ell}^\star$, and in the case of a loop closure, $\bar{\mathbf{T}}_{t_2,t_1}^{\text{LC}} \leftarrow \exp((\delta\boldsymbol{\xi}_{t_2,t_1}^{LC\ \star})^\wedge)\bar{\mathbf{T}}_{t_2,t_1}^{\text{LC}}$.

Implementation Efficiency

There are three major computational costs associated with running this estimator. The first major cost is the numerical integrations of (5.17) and (5.47). We note from the derivation of (5.34) that the integral term in $\mathbf{H}_{\mathbf{c},kj}$ can be written as

$$\int_{t_j}^{t_k} \bar{\boldsymbol{\mathcal{T}}}_{t_k,\tau}\boldsymbol{\Psi}(\tau)d\tau = \text{Ad}(\bar{\mathbf{T}}(t_k))(\mathbf{I}(t_k) - \mathbf{I}(t_j)), \tag{5.53}$$

where,

$$\mathbf{I}(t) := \int_0^t \text{Ad}(\bar{\mathbf{T}}(\tau))^{-1}\boldsymbol{\Psi}(\tau)d\tau. \tag{5.54}$$

Given that many of the required integral calculations will overlap, we aim to compute all the integral terms efficiently by conducting a single pass over the entire time period that we are currently optimizing. In the full batch optimization, the following equations are used iteratively for $t = [t_0, t_K]$,

$$\bar{\mathbf{T}}(t + \Delta t) \leftarrow \exp\left((\Delta t \cdot \boldsymbol{\Psi}(t)\bar{\mathbf{c}})^{\wedge}\right) \bar{\mathbf{T}}(t), \tag{5.55a}$$

$$\mathbf{I}(t + \Delta t) \leftarrow \mathbf{I}(t) + \Delta t \cdot \mathrm{Ad}(\bar{\mathbf{T}}(t))^{-1} \boldsymbol{\Psi}(t), \tag{5.55b}$$

where $\bar{\mathbf{T}}(0) = \mathbf{1}$ and $\mathbf{I}(0) = \mathbf{0}$. The integration step, Δt, is chosen so that we have intermediate terms for $\bar{\mathbf{T}}$ and $\mathbf{I}$ at each measurement time, t_{kj}. By storing the intermediate steps, we can compute $\bar{\mathbf{T}}_{t_k,t_j}$ and $\mathbf{G}_{kj}$ for each measurement.

The second major cost in this estimator is building the matrix $\mathbf{A}_{\text{meas}}$. This step can be exceptionally expensive depending on the choice of basis functions used to parameterize $\boldsymbol{\Psi}(t)$. Specifically, it is highly beneficial to use a basis function type that has *local support*. At any given time, we want only a subset of the basis functions, $\boldsymbol{\psi}_b(t)$, to be non-zero. Using local support introduces an exploitable sparsity to the matrix $\boldsymbol{\Psi}(t)$. As mentioned in Section 5.3.4, our implementation of the estimator uses cubic B-spline basis functions, because they have local support in addition to simple analytical integrals and derivatives.

The third cost is related to solving the linear system of equations in (5.50). To do this efficiently, we note that it can be written using the 2×2 block structure,

$$\begin{bmatrix} \mathbf{A}_{xx} & \mathbf{A}_{x\ell} \\ \mathbf{A}_{x\ell}^T & \mathbf{A}_{\ell\ell} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\theta}^{\star} \\ \delta\boldsymbol{\ell}^{\star} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_x \\ \mathbf{b}_\ell \end{bmatrix}. \tag{5.56}$$

Notably, the relative, continuous-time formulation of the problem changes the traditional sparsity pattern of $\mathbf{A}_{xx}$; this is due to the fact that $\mathbf{A}_{\text{pri}}$ is now band diagonal, and the relative transforms, $\mathbf{T}_{t_k,t_j}$, may depend on long strides of temporally related state variables in $\mathbf{c}$, and sometimes loop closures, $\mathbf{T}^{\text{LC}}_{t_2,t_1}$. However, since $\mathbf{A}_{\ell\ell}$ is still block-diagonal and is typically much larger than $\mathbf{A}_{xx}$, we may apply the usual Schur complement, discussed in Section 3.1.3, with no additional cost.

B-spline Window Filter

Here we describe our continuous-time approach to the incremental optimization methodology discussed in Section 3.3.2. In order to run the estimator online we optimize over a subset of the basis functions used in the time segments leading up to the most recent measurement. However, in order for this technique to achieve constant-time performance,

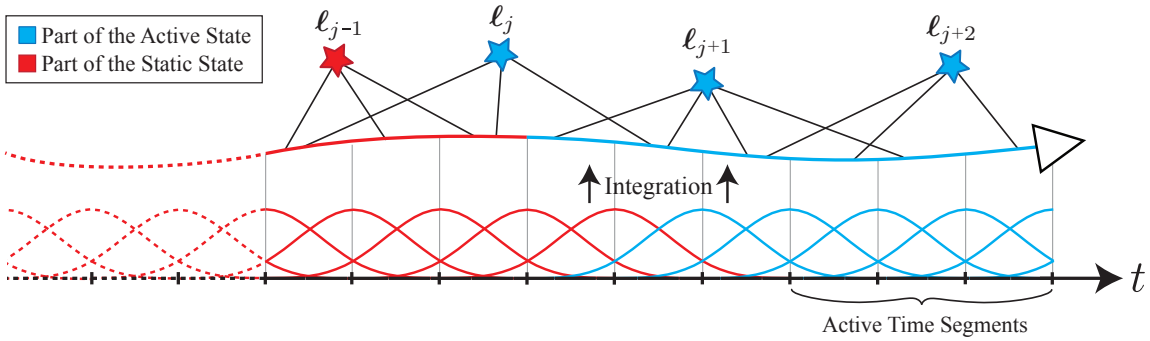


Figure 5.3: This figure depicts the use of active state parameters in a window-style filter. For illustration, uniformly weighted cubic B-spline basis functions are used to portray the parametric velocity profile estimate. The integrated robot trajectory, shown above the velocity profile, is depicted to align on the time axis, with the landmarks observed at the connected timestamps. Determining the *active state* begins by defining the active time segments; any basis function that influences the active time segments becomes part of the *active state*. Landmark variables are considered *active* if they are observed from a time that overlaps with an *active* basis function. The *static* region is required to calculate expected measurements for landmarks with observations from both *active* and *static* time intervals (e.g., $\boldsymbol{\ell}_j$).

the subset of basis functions must have temporally local effects on the robot trajectory, $\boldsymbol{\varpi}(t)$. For this to be true, it is a requirement that the basis functions used in the estimator have local support.

Before setting up the local batch optimization problem, we begin by identifying the *active state*, illustrated in Figure 5.3. The active subset of basis functions and all landmarks measured during the affected time period are added to a list of active parameters that will be updated in the optimization. All measurements of these active landmark parameters are included when building the terms $\mathbf{A}_{\text{meas}}$ and $\mathbf{b}_{\text{meas}}$. This may require the full nominal solution, $\bar{\mathbf{c}}$, as some of the *static* basis function coefficients may be required to evaluate $\bar{\boldsymbol{\varpi}}(t)$ near and outside the bounds of the *active* time period.

In the case of a loop closure, the *active state* selection changes slightly to include both the discrete loop-closure transformation and the temporal basis functions on both sides of the spatial link; the basis functions that are made *active* are the ones that overlap in time with any of the landmark observations that were matched across the loop closure.

Pose-Graph Relaxation

Although the relative SLAM problem formulation argues that a global map is not necessary for many common robotic tasks, a global map may be desirable or required in some

situations; for example, human inspection of a large-scale map, or human-in-the-loop decisions, such as macro-level path planning or goal setting.

In order to generate a globally consistent SLAM solution in a computationally tractable manner, a common practice is to implement *pose-graph relaxation* or *pose-graph SLAM*; the derivation of which is well known (Olson et al., 2006; Grisetti et al., 2007). The general concept of the optimization is to use the estimated discrete relative pose transforms as measurements in a new problem that aims to solve for the global poses.

To relax the relative continuous-time trajectory into global coordinates, we first discretize our solution into a set of relative transforms, $\mathbf{T}_{t_{d+1},t_d}$, which occur between times t_d and t_{d+1}, for $d = 1 \dots D$. Obtaining the relative transforms is straightforward using the integration scheme in (5.17),

$$\mathbf{T}_{t_{d+1},t_d} = \mathbf{T}(t_{d+1})\mathbf{T}(t_d)^{-1} = \prod_{t_d}^{t_{d+1}} \exp(\boldsymbol{\varpi}(\tau)^\wedge d\tau). \tag{5.57}$$

However, obtaining the uncertainty of the relative pose estimates,

$$\boldsymbol{\Sigma}_{d+1,d} := E[\delta\boldsymbol{\xi}_{t_{d+1},t_d}\delta\boldsymbol{\xi}_{t_{d+1},t_d}^T], \tag{5.58}$$

involves some additional derivation. Note that the result from Barfoot and Furgale (2014) on compounding poses using fourth-order terms is not directly applicable to finding the uncertainty of the transform, $\mathbf{T}_{t_{d+1},t_d}$, because $\delta\boldsymbol{\xi}(t_d)$ and $\delta\boldsymbol{\xi}(t_{d+1})$ are correlated.

Using the relationship in (5.35) it is straightforward to see that

$$\delta\boldsymbol{\xi}_{t_{d+1},t_d} \approx \left(\bar{\boldsymbol{\mathcal{T}}}(t_{d+1}) \int_{t_d}^{t_{d+1}} \bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}\delta\boldsymbol{\varpi}(\tau)d\tau\right), \tag{5.59}$$

and therefore,

$$\begin{aligned}
\boldsymbol{\Sigma}_{t_{d+1},t_d} =& E\Big[\delta\boldsymbol{\xi}_{t_{d+1},t_d}\delta\boldsymbol{\xi}_{t_{d+1},t_d}^T\Big] \\
\approx& E\left[\left(\bar{\boldsymbol{\mathcal{T}}}(t_{d+1}) \int_{t_d}^{t_{d+1}} \bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}\delta\boldsymbol{\varpi}(\tau)d\tau\right)\left(\bar{\boldsymbol{\mathcal{T}}}(t_{d+1}) \int_{t_d}^{t_{d+1}} \bar{\boldsymbol{\mathcal{T}}}(\tau')^{-1}\delta\boldsymbol{\varpi}(\tau')d\tau'\right)^T\right] \\
=& \bar{\boldsymbol{\mathcal{T}}}(t_{d+1}) \int_{t_d}^{t_{d+1}} \int_{t_d}^{t_{d+1}} \bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1} E\Big[\delta\boldsymbol{\varpi}(\tau)\delta\boldsymbol{\varpi}(\tau')^T\Big]\bar{\boldsymbol{\mathcal{T}}}(\tau')^{-T}d\tau'd\tau\bar{\boldsymbol{\mathcal{T}}}(t_{d+1})^T.
\end{aligned} \tag{5.60}$$

In practice, because we have parameterized the solution using basis functions, $\delta\boldsymbol{\varpi}(t) = \boldsymbol{\Psi}(t)\delta\mathbf{c}$, it is convenient to compute the covariance as follows,

$$\boldsymbol{\Sigma}_{t_{d+1},t_d} \approx \left(\bar{\boldsymbol{\mathcal{T}}}(t_{d+1}) \int_{t_d}^{t_{d+1}} \bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1}\boldsymbol{\Psi}(\tau)d\tau\right) E\big[\delta\mathbf{c}\,\delta\mathbf{c}^T\big]\left(\bar{\boldsymbol{\mathcal{T}}}(t_{d+1}) \int_{t_d}^{t_{d+1}} \bar{\boldsymbol{\mathcal{T}}}(\tau')^{-1}\boldsymbol{\Psi}(\tau')d\tau'\right)^T, \tag{5.61}$$

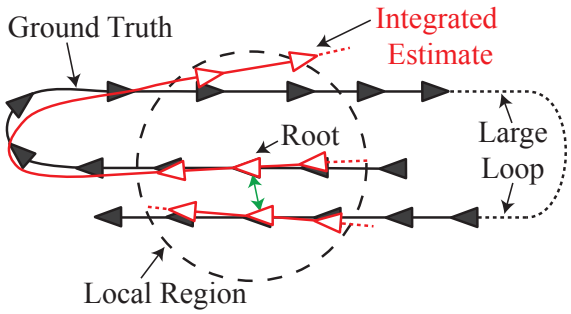


Figure 5.4: Illustration of the local RMS error calculation for a single root node. Note that open-loop integration (OLI) outside of the local region is needed to find the top two poses, while closed-loop integration (CLI) can be used to quickly find the lower poses.

where $E[\delta\mathbf{c}\,\delta\mathbf{c}^T]$ can be evaluated using part of the solution from our Gauss-Newton optimization problem in (5.50), after convergence. Using the Schur complement for an open-loop optimization, we know that

$$\mathrm{cov}(\delta\boldsymbol{\theta}^\star, \delta\boldsymbol{\theta}^\star) = E\left[\delta\mathbf{c}^\star\,\delta\mathbf{c}^{\star T}\right] = (\mathbf{A}_{xx} - \mathbf{A}_{x\ell}\mathbf{A}_{\ell\ell}^{-1}\mathbf{A}_{x\ell}^T)^{-1}, \tag{5.62}$$

and in a closed-loop optimization,

$$\mathrm{cov}(\delta\boldsymbol{\theta}^\star, \delta\boldsymbol{\theta}^\star) = \begin{bmatrix} E\left[\delta\mathbf{c}^\star\,\delta\mathbf{c}^{\star T}\right] & E\left[\delta\mathbf{c}^\star\,\delta\boldsymbol{\xi}_{t_2,t_1}^{LC\ \star T}\right] \\ E\left[\delta\boldsymbol{\xi}_{t_2,t_1}^{LC\ \star}\,\delta\mathbf{c}^{\star T}\right] & E\left[\delta\boldsymbol{\xi}_{t_2,t_1}^{LC\ \star}\,\delta\boldsymbol{\xi}_{t_2,t_1}^{LC\ \star T}\right] \end{bmatrix} = (\mathbf{A}_{xx} - \mathbf{A}_{x\ell}\mathbf{A}_{\ell\ell}^{-1}\mathbf{A}_{x\ell}^T)^{-1}, \tag{5.63}$$

where $E\left[\delta\boldsymbol{\xi}_{t_2,t_1}^{LC\ \star}\,\delta\boldsymbol{\xi}_{t_2,t_1}^{LC\ \star T}\right]$ is the uncertainty associated with $\mathbf{T}_{t_2,t_1}^{\mathrm{LC}}$.

5.3.5 Appearance-Based Lidar Experiment

In this section, we test and report results for our relative continuous-time SLAM technique. The estimator is tested using a set of real data that highlight our algorithm's ability to handle high-rate asynchronous measurements, while the robot undergoes motion in unstructured terrain. The 1.1km traversal, measurements models, and place recognition, are described in Appendix A.

Local Accuracy Metric

Before discussing the results of the estimator, we begin by introducing the average local-area root-mean-square (ALARMS) error, a new metric similar to the one introduced by Sibley et al. (2010), which we use to evaluate *local* metric accuracy. The ALARMS error calculation is illustrated in Figure 5.4, and proceeds as follows:

For each ground-truth node,

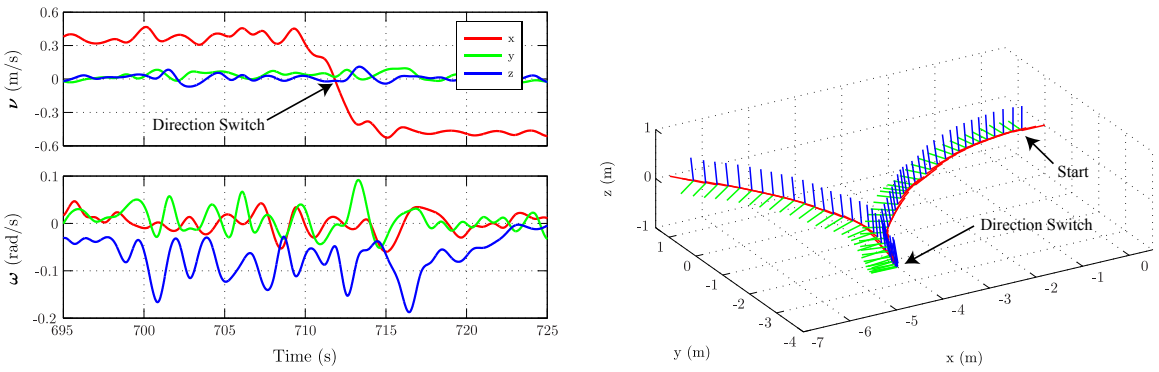
1. Search the ground truth (GPS) for all other nodes within some specified distance of the current (root) node – this is our *local region*
2. Perform a breadth first search (BFS) using the available topological information, starting with the root node, to determine the shortest paths to all other local nodes
3. Integrate along the found paths to obtain the pose of local nodes in the root node's coordinate frame
4. Fix the root node's pose to ground truth; in the absence of global orientation information, align the trajectories using all the integrated local node positions and ground truth counterparts
5. Compute RMS Euclidean error for the local region subgraph

The ALARMS error is then the average of all the calculated RMS errors terms, where each error term is generated by using one of the ground-truth nodes to form a local region. In the absence of loop closures, only temporal links contribute to the topological connectivity of the graph.

The purpose of this error metric is to evaluate the *local* (rather than *global*) metric accuracy of an algorithm. Enforcing a globally consistent solution (e.g., via pose-graph relaxation) before comparing to ground-truth prevents us from gaining insight to an algorithm's *local* accuracy. The ALARMS error metric evaluates accuracy within a fixed-size region (averaged over 'all of the regions'); by then varying the region size we can analyze the performance of an algorithm over different scales.

Open-Loop Solution

Processing the 6880 intensity images (in a VO-style fashion), we found 2,010,671 temporally matched SURF feature measurements, each with a unique timestamp. Assuming the use of a (brute-force) discrete-time batch estimator (with a motion prior), the state parameterization would require a pose estimate at each measurement time in the system; meaning that for the 1.1km dataset, it would naively require $6 \times 2{,}010{,}671 = 12{,}064{,}026$ state variables, excluding the 688,481 different landmark positions. Furthermore, the plain discrete-time solution is ill-conditioned without a motion prior and cannot be solved. Using basis functions to parameterize our continuous-time state, we show how the number of required state variables can be drastically reduced.



(a) Velocity Profile

(b) Integrated Trajectory

Figure 5.5: Example output from the relative continuous-time SLAM estimator using real data. The linear and angular velocity estimates in (a) directly correspond to the integrated 3D trajectory in (b); coordinate frames are plotted at a uniform time interval. The direction switch in forward (x) velocity can be easily observed in the integrated trajectory.

Before running the constant-time estimator on the entire 1.1km trajectory, we must decide on a spacing for the temporal basis functions and an appropriate window size over which to optimize. In order to train these two parameters, we use a 250m subsection of the full trajectory. To compare the results of the estimator to the GPS ground truth, we consider the integral of our velocity estimate, as seen in Figure 5.5.

To find an appropriate knot spacing, we iterated the full batch estimator on the training section, varying the number of time segments in the B-spline parameterization from 5 to 3000. The accuracy of the solutions are quantified by considering both the total root-mean-square (RMS) Euclidean error, comparing the integrated position of our velocity estimate to the DGPS tracks, and the ALARMS error, for a 10 metre local region. The estimate and DGPS tracks were put into a common reference frame by aligning the first 10 meters of each trajectory.

The result of varying the number of knots can be seen in Figure 5.6 (timing was broken down into the three major steps described in Section 5.3.4). Using the efficient methods described earlier, we see that the integration and build steps of the algorithm are relatively constant in cost as the number of knots increases. As expected, the solution to $\mathbf{A}\delta\mathbf{z}^\star = \mathbf{b}$ is cubic in complexity and begins to take a larger portion of the total time as the number of basis functions is increased. As the knot spacing is reduced, the parameterization is given enough degrees of freedom to properly represent the velocity profile of the robot; however, there are diminishing returns with respect to reducing error. For this subsection of the dataset, it was decided that 1200 time segments (a knot spacing of approximately

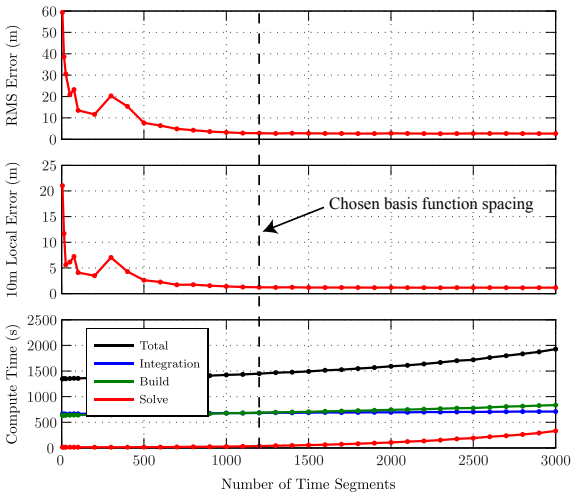


Figure 5.6: The full batch estimator was run on a 250m subsection of the full dataset. Varying the number of time segments in the parameterization from 5 to 3000, we aim to find an appropriate knot spacing to use over the entire trajectory. The chosen number of time segments was 1200 (a knot spacing of approximately 0.65 seconds).

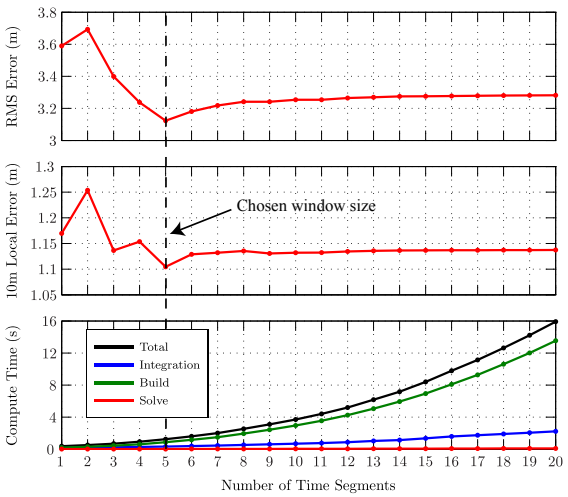


Figure 5.7: Using 1200 time segments in the B-spline parametrization over the 250m training segment, we varied the number of active time segments in each optimization window from 1 to 20. The chosen optimization window size for the final experiment was 5 time segments. The compute time is the average time it took to compute a single window optimization.

0.65 seconds) was acceptable to represent the robot trajectory.

In order to limit the growth rate in solve time, we applied our window-style filter to the 250m trajectory. In a similar fashion to the first experiment, we then varied the number of active time segments in the optimization window from 1 to 20. The results in Figure 5.7 indicate that a window size of only 5 time segments would be sufficient to use over the entire trajectory. The compute time shown in this experiment is the average amount of time taken to solve a single window optimization. Increasing window size has a clear negative effect on the amount of time spent integrating and building the terms needed in the solution.

The final result for the full trajectory estimate, seen in Figure 5.8, is compared to a discrete-time relative bundle adjustment implementation. To keep the discrete implementation's state size tractable and solvable, a single relative pose estimate is used for each of the image stacks and the laser scans are assumed to be taken instantaneously (i.e., no motion distortion). The open-loop integration (OLI) of our velocity estimate and discrete relative bundle adjustment solution are put into a common reference frame

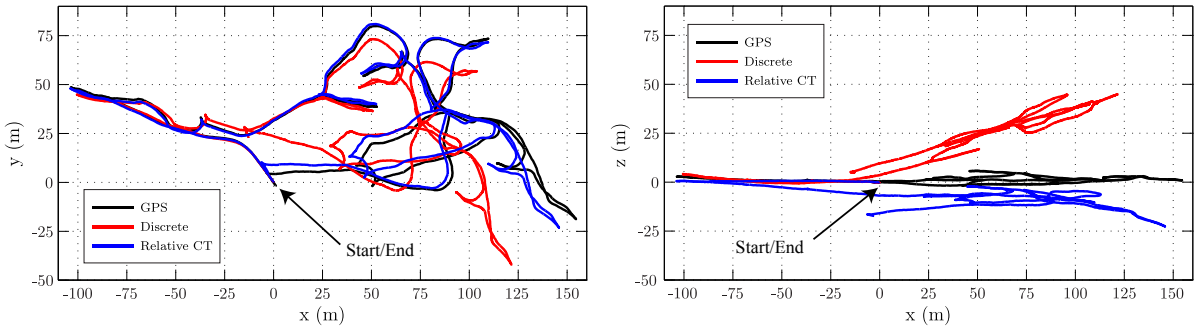


Figure 5.8: Top and side views of the robot trajectory over the whole 1.1km dataset. Our *Relative CT* estimator used a knot spacing of 0.65 seconds, and an active window size of 5 time segments. The relative discrete-time solution used an active window size of 6 images. DGPS tracks are in black, the discrete-time relative bundle adjustment solution is in red, and the open-loop integration (OLI) of the velocity profile estimate is in blue. Note the significant improvement made by properly accounting for the temporal nature of the lidar measurements.

with the DGPS tracks by aligning the first 30 meters of each trajectory. Compensating for motion distortion by using our continuous-time estimation scheme clearly produces superior results in contrast to the traditional discrete-time approach.

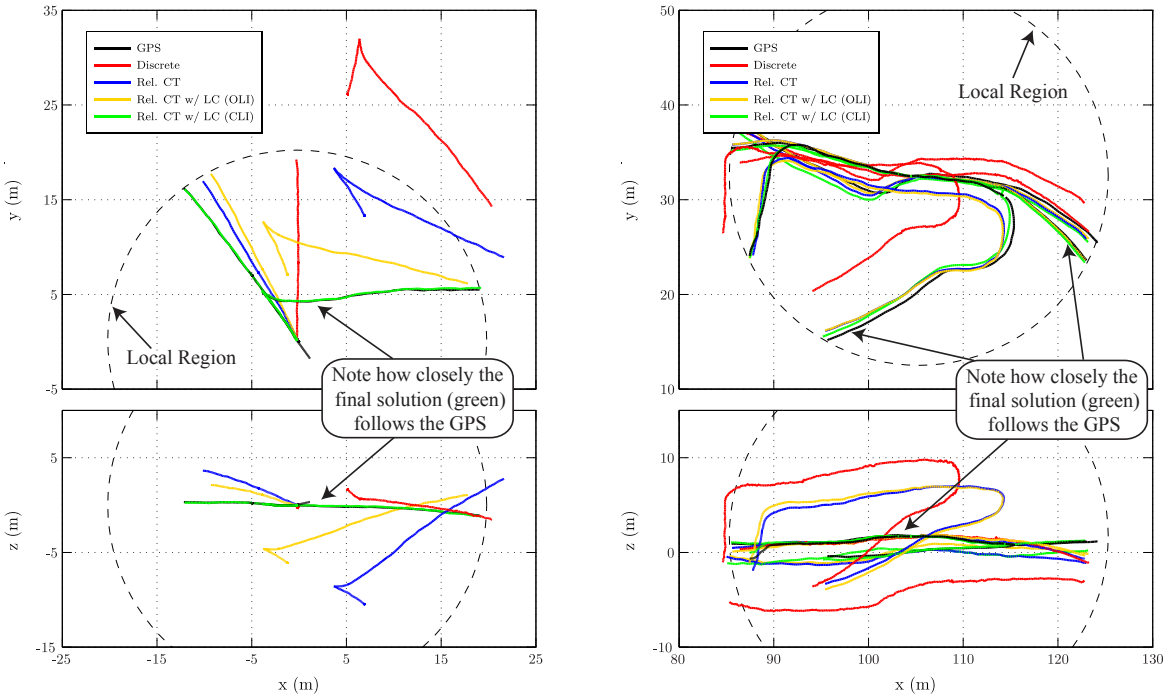
As suggested by the results of the first experiment, the estimator used a knot spacing of 0.65 seconds. This correlated to the use of only $6 \times 5{,}386 = 32{,}316$ state variables in the parameterization of $\boldsymbol{\varpi}(t)$. In relation to the second experiment, it was decided that the estimator should use a window size of 5 active time segments; this cost approximately 1.3 seconds per local batch optimization and led to a total run time of 1.6 hours[2].

Closed-Loop Solution

Using the visual place-recognition algorithm proposed by MacTavish and Barfoot (2014), followed by geometric verification, we were able to detect nine loop-closure locations (discussed in Appendix A.3). An optimization was then performed around each loop-closure for the spatial transformation and the basis functions overlapping in time with spatially matched landmark observations. In order to see how merging landmarks across loop closures improves the open-loop estimate, the solution is integrated using both an OLI and closed-loop integration (CLI) scheme.

Figure 5.9 shows the trajectory integrations for two 20 metre local regions, selected as part of ALARMS error calculations, which included loop closures. The first region, in

[2]Implemented in Matlab and timed with a 2.4GHz processor and 8GB of 1333MHz DDR3 RAM.



(a) Region 1 (b) Region 2

Figure 5.9: These figures show the trajectory integrations for ALARMS error calculations in two different areas; each calculation used a 20 m local region size and included a loop closure. The first region, shown in (a), is near the start/end of the 1.1km trajectory and contains a loop closure connecting the two trajectory segments. An incremental performance increase is clearly visible, with the final solution being very near the GPS ground truth. The second region, shown in (b), includes three trajectory segments, two of which are connected by a loop closure that spans nearly 200 m. Again, an incremental performance increase is clearly visible.

Figure 5.9(a), includes two trajectory segments, which contain the start and end of the 1.1km traversal. The second region, in Figure 5.9(b), includes three trajectory segments, two of which are connected by a loop closure that spans roughly 200m. Descriptions of the algorithms and the resultant RMS error for each region is shown in Table 5.1.

As expected, the discrete algorithm configuration, which does not consider motion distortion, performs poorly in comparison to all of the continuous-time solutions. Note that the integration of the trajectory in the second region included less open-loop distance, and therefore it is expected that the magnitude of errors in the results would be smaller. The expected reduction in error between the open-loop integrations of the relative continuous-time trajectory, with and without loop closure, is fairly minimal. In the first region, we see a fairly substantial improvement, which can be attributed to the small incremental improvements over the very long open-loop integration distance. In the second region,

Table 5.1: This table describes the different algorithm configurations that will be contrasted for local accuracy. The two estimators are the *Discrete* estimator, that assumes no motion distortion, and the *Relative CT* estimator using basis function on body-centric velocity. All configurations use the feature correspondences provided by the motion-compensated RANSAC algorithm presented in Chapter 4. The final two configurations take advantage of loop closure information provided by the place-recognition algorithm, but are integrated in different fashions; one using open-loop integration (OLI) and one using closed-loop integration (CLI). Finally, RMS errors are shown for the trajectory integrations investigated in Figure 5.9.

Label	Loop Closures	Integration Scheme	Region 1 RMS Err. [m]	Region 2 RMS Err. [m]
Discrete	No	OLI	13.95	6.32
Rel. CT	No	OLI	8.61	2.83
Rel. CT w/ LC (OLI)	Yes	OLI	4.41	2.89
Rel. CT w/ LC (CLI)	Yes	CLI	0.40	1.14

there is a slight increase in measured error; although it is possible that the additional measurements and fusion of landmarks reduced the accuracy of the estimation, it must be considered that the circular error probability (CEP) of our GPS is only 0.4m in the horizontal plane. Furthermore, this CEP is reported under ideal conditions and the vertical dimension is typically worst. Finally, we note the solution using CLI in the second region had more than double the error of the first region for a trajectory of similar size; if we consider only the two pieces of trajectory tied by loop closure in the second region, we find the RMS error of the final solution is only 0.44m, giving some insight to the importance of place recognition for increasing local estimation accuracy. We contrast the various algorithm configurations more broadly, over the full 1.1km trajectory, by comparing the ALARMS error for varying local region sizes, as seen in Figure 5.10. Lastly, we note that the number of available loop closures in this dataset was fairly low, but that the algorithm should scale well with additional loop closures; Sibley et al. (2010) show in their discrete-time relative bundle adjustment algorithm that the addition of many loop closures significantly reduces mapping error, even before global relaxation.

In an offline process, we discretized the continuous-time trajectory, solved for the covariance of the relative transforms, and performed pose-graph relaxation using the nine optimized loop closures. The resulting globally consistent trajectory estimate can be seen in Figure 5.11. While a globally consistent map of this quality is useful for tasks such as macro-level path planning, or low-frequency human-in-the-loop decisions, we reiterate that the strength of the relative formulation is a constant-time, locally accurate solution aimed at enabling autonomous tasks, such as navigation or object avoidance.

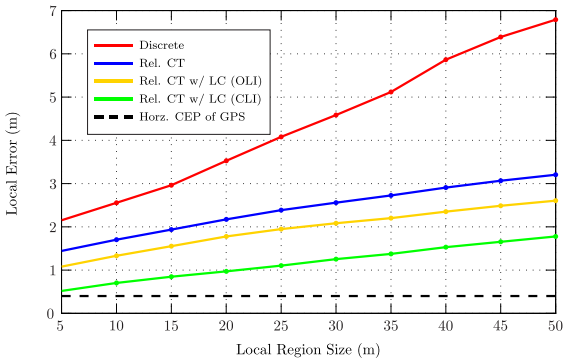


Figure 5.10: This figure shows ALARMS error for a number of local region sizes, using trajectory estimates of the entire 1.1km trajectory. Note the estimation error drift for continuous-time solutions is far lower than that of the discrete solution. Furthermore, the addition of loop closure improves both the open-loop and closed-loop results.

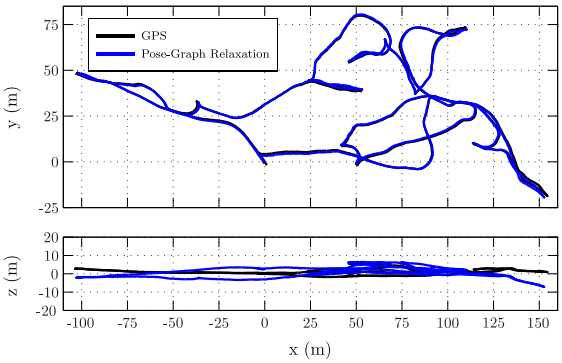


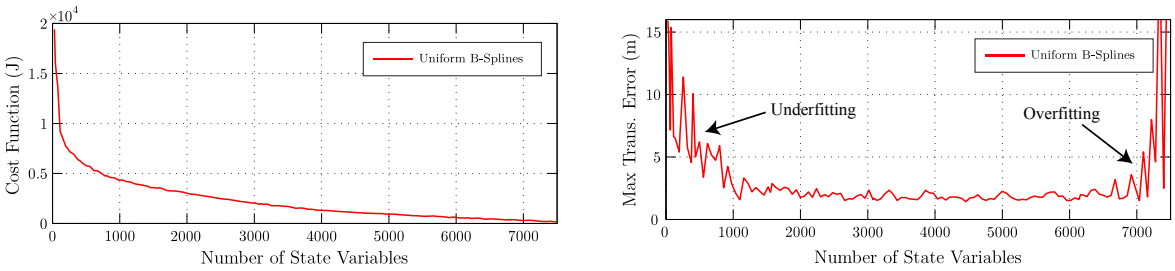
Figure 5.11: This figure shows the globally consistent trajectory estimate generated by discretizing the full 1.1km continuous-time trajectory, and performing a pose-graph relaxation using the nine available loop closures. The RMS error for the full trajectory is 2.9m. If we consider only the x-y plane, then the RMS error is 0.77m over the full 1.1km path.

From a navigational standpoint, the ability to close loops in constant time and generate highly accurate local maps is significant; especially with the growing popularity of techniques that aim to localize against previously traversed paths in order to repeat routes (Furgale and Barfoot, 2010; Churchill and Newman, 2012).

5.4 Discussion

The relative, parametric approach to STEAM described in this chapter provided good estimation accuracy and exhibits the properties that we wish to have in our ideal localization engine. However, the estimator performs much more slowly than would be hoped. The core issue being that we require numerical integration of the velocity profile to form expected measurements and Jacobians. Furthermore, the fully relative solution also adds a severe amount of density to the problem; while the general complexity is unaffected (using the Schur complement), the secondary sparsity is very poor.

The major issue that needs to be addressed in the parametric approach is the algorithm that determines the knot spacing. In this thesis, we made use of a uniform spacing that needed to be trained prior to the experiment; in general, this is not desirable. Although the knot-space training in Figure 5.6 would have us believe that it is safe to simply choose a very fine knot spacing, further experimentation has shown us both ends of the spectrum



(a) Objective Function Cost

(b) Translational Error

Figure 5.12: These figures illustrate the issue of underfitting and overfitting with uniformly spaced B-spline basis functions. This experiment used a 200m section of the appearance-based lidar dataset, but for simplicity, was conducted on a pose-graph relaxation problem. Note that as we add addition state variables (i.e., knots) the cost function decreases. However, while error decreases at first, the increased resolution eventually allows it to overfit the measurements.

exhibit poor behavior (see Figure 5.12). In an attempt to rectify this issue, the works of Oth et al. (2013) and Anderson et al. (2014)[3] attempt an adaptive parametric approach in which the *normalized innovation squared* (NIS) is to determine whether enough 'spline resolution' is available in a time segment; notably the use of NIS is *very* dependent on outlier rejection being performed robustly. Oth et al. (2013) extend the traditional B-spline method presented here by subdividing time segments with additional knots. The method that we proposed (Anderson et al., 2014) took an alternative approach in which a hierarchical wavelet representation was used; in this parametrization, the smoothest spline level is constructed from uniform B-splines, and is overlaid with hierarchical levels of finer wavelets functions. If a time segment was determined to need more resolution, we unlocked finer levels of wavelets in the temporal area. Both techniques showed promise, but further validation (expressly on the robustness of the algorithms) should be performed.

Looking ahead to Chapter 6, we note that the parametric approach suffers when the knot spacing is small because it does not focus on accurately modelling the motion prior. Instead, a weak prior is typically used and the inherent smoothness of the trajectory is chosen through the knot spacing. While this does provide a level of robustness, chosen by the engineer, it also begs the question: *"Could an extremely fine knot spacing be used reliably if we had a strong prior over an accurate motion model?"*. Although accurately modelling the prior is no simple feat, it inspires the exploration of Gaussian-process regression as an alternative trajectory representation. In the weight-space view of Gaussian-process regression, a GP can be thought of as a parametric spline with *infinite*

[3]Our work on wavelets (Anderson et al., 2014) was not included in this thesis due to page constraints.

temporal basis functions – the solution is constrained by putting much more focus on modelling a strong prior distribution.

5.5 Summary and Conclusions

In this chapter we have presented a batch, parametric, relative, singularity-free, and continuous-time SLAM estimator that is able to find an incremental approximation to the MAP problem in constant-time. The estimator presented in this chapter adopts the completely relative problem formulation by Sibley et al. (2009). However, as the proposed relative method is also a continuous-time solution, it maintains the ability to overcome the state-size tractability issues associated with high-rate and scanning sensors by using the parametric framework derived by Furgale et al. (2012). To attain this fusion of techniques, the open-loop trajectory is represented by a continuous-time estimate of the robot's velocity, while large-scale loop closures are described by discrete transformations that relate the robot's pose at two different times. The algorithm was validated using a 1.1 kilometer dataset of motion-distorted lidar intensity imagery. Using this dataset, we demonstrate that our algorithm is able to compensate for motion-distortion and provide odometry results much more suitable to navigation than a discrete-time counterpart. The methods and results presented in this chapter were initially published without loop-closure results, by Anderson and Barfoot (2013b), and then fully by Anderson et al. (2015b).

In summary, the contributions of this chapter are:

1. A continuous-time adaptation of the relative-coordinate formulation of SLAM. We demonstrate that by estimating the velocity profile of the robot, we can devise an incremental SLAM solution that runs in constant time – even at loop closure.
2. An explanation of how to perform sliding-window-style estimation with basis functions that have a local support greater than two.
3. An evaluation of the algorithm using lidar-style VO and appearance-based place recognition over a 1.1 kilometer traverse of unstructured, three-dimensional terrain.

This work was an important step towards developing real-time continuous-time solutions on the relative pose manifold. In particular, the knowledge we acquired from the derivations and experiments presented in this chapter greatly influenced the approach of our (final) GP-based algorithm (presented at the end of Chapter 6).

Chapter 6

Gaussian-Process Representations for Trajectories

In this chapter we explore novel extensions of the Gaussian-process (GP) regression approach to batch continuous-time trajectory estimation. The approach, first introduced by Tong et al. (2012, 2013), directly models the robot trajectory as a one-dimensional GP (where time is the independent variable) and uses batch estimation to solve for the robot state at the measurement times. Barfoot et al. (2014) extend this work by introducing a class of *exactly sparse* GP priors based on *linear* time-varying (LTV) stochastic differential equations (SDE); a sparse set of trajectory-smoothing terms are generated by using a physically motivated SDE, driven by white noise, to derive costs associated with temporal change in the *Markovian* state variables. The GP approach also provides a principled interpolation scheme based on the chosen motion model. This interpolation can be used both to query the robot state at any time and to heuristically compress the state vector using *keytimes* (as suggested by Tong et al. (2013)).

Inspired by the work of Tong et al. (2013) and Barfoot et al. (2014), this thesis extends the GP approach to trajectory estimation and the class of exactly sparse GP priors in several ways. First, we build upon the exactly sparse approach that used *linear*, time-varying SDEs, and show how to use GPs based on *nonlinear*, time-varying (NTV) SDEs, while maintaining the same level of sparsity. The algorithmic differences are discussed in detail and results are provided using comparable linear and nonlinear priors. Next, we show how the block-tridiagonal sparsity of the kernel matrix can be exploited to improve the computational performance of log-evidence hyperparameter training. We also propose a kinematically motivated and exactly sparse GP prior for

bodies that are translating and rotating in 3D space (i.e., $\mathbf{x}(t) \in SE(3)$). Using this fast, singularity-free, and physically-motivated method, we demonstrate the ability to interpolate measurement times from a sparser set of *keytimes* and draw connections to the parametric continuous-time estimation schemes. Furthermore, we describe an approach to adopt a *relative* continuous-time solution, drawing inspiration from the work of Sibley et al. (2009) and the results of Chapter 5. Finally, all of the algorithms are validated using data from real hardware experiments; the NTV SDE and hyperparameter extensions use a 2D mobile robot dataset and the 3D method is validated with both a stereo-camera dataset and the appearance-based lidar dataset described in Appendix A.

The novel contributions of this chapter are: (i) a methodology to use NTV SDEs to generate an *exactly sparse* GP prior for trajectories that belong to a vectorspace, (ii) an analysis of the complexities that arise from needing an operating point consisting of a full time-varying trajectory, (iii) an analysis of how data-driven hyperparameter training can also exploit the *exactly sparse* nature of our GP priors, (iv) an evaluation of both the nonlinear GP prior and the data-driven hyperparameter training for a 2D mobile robot dataset, (v) a theoretical extension of our linearization methodology for a NTV SDE that belongs to $SE(3)$, (vi) a piecewise continuous GP prior that uses a sequence of *local* LTI SDEs to closely approximate a desired NTV SDE, (vii) a description of the methodology we use to extend our GP estimator to the *relative* coordinate formulation of STEAM, (viii) an evaluation of our estimator's accuracy, consistency, computational performance, and interpolation functions, using a stereo camera dataset, and (ix) an evaluation using lidar-style VO and place recognition. The contributions presented in this chapter have appeared in two previous publications. First, the work of Barfoot et al. (2014) was extended by our work on NTV SDEs and hyperparameter training for publication in a journal article (Anderson et al., 2015a). Second, our exactly sparse GP prior for $SE(3)$ was published in the proceedings of a full-paper refereed conference (Anderson and Barfoot, 2015); note that the *relative* SLAM extension and appearance-based lidar results were not included in this publication.

6.1 Related Work

Gaussian processes have been used extensively in the field of *machine learning* to solve a variety of classification and regression problems; Rasmussen and Williams (2006) is an excellent introductory resource on this topic. For regression, GPs are typically used

to model an unknown continuous function that maps a set of known inputs to a set of (noisy) measured outputs. The GP model predicts the output (with uncertainty) for unmeasured input quantities by taking a data-driven approach; in essence, a prediction is made by querying the posterior (mean and covariance) of the output function, given: (i) a query input, (ii) continuous prior mean and covariance functions over the output space, and (iii) a set of measured outputs for known inputs (with uncertainty information). In contrast to this probabilistic and data-driven approach, *parametric* regression models typically require many non-trivial engineering decisions that can affect accuracy (e.g., the shape and spacing of basis functions).

In robotics, GPs have been used to accomplish a wide variety of tasks, such as modelling the distortion of a camera lens (Ranganathan and Olson, 2012), assessing terrain for driveability (Berczi et al., 2015), and even large-scale surface modelling (Vasudevan et al., 2009). Concerning robotic state estimation, GPs have been used as a dimensionality reduction technique (Ferris et al., 2007; Lawrence, 2003), a method to represent both nonlinear observation and motion models (Deisenroth et al., 2012; Ko and Fox, 2009, 2011), and even to map optical flow vectors directly to metric odometry estimates (Guizilini and Ramos, 2012). In contrast to these works, the GP-regression approach to continuous-time trajectory estimation, introduced by Tong et al. (2013), is distinctly different; instead of modelling the continuous function between an input and output, it proposes modelling the *latent* trajectory function, $\mathbf{x}(t)$, as a GP, where time is the input and the discrete observations are generated from a (nonlinear, noninvertible) measurement model, $\mathbf{g}(\mathbf{x}(t_k))$, that depends on the *hidden* trajectory value, $\mathbf{x}(t_k)$, at input time t_k.

Akin to the batch continuous-time estimation problem described in Section 3.4, the approach of Tong et al. (2013) is built upon the use of a continuous-time GP prior (consisting of both a mean and covariance function) and a set of discrete-time observations. Drawing some similarity to the discrete-time batch problem formulation, the GP-regression approach first solves for the trajectory *at the measurement times*, $\mathbf{x}(t_k)$, by formulating a Gauss-Newton estimator to iteratively improve the best guess of the (discretized) posterior trajectory. Post optimization, the GP linear prediction equations (Rasmussen and Williams, 2006) can then be used to query the mean and covariance of the posterior trajectory at other times of interest. Although the original derivation by Tong et al. (2013) was general, their early implementations of a GP prior failed to realize the *Markovian* nature of the problem and resulted in a profoundly slow batch trajectory estimation scheme – due to the need to invert a large, dense kernel matrix. The work of Barfoot

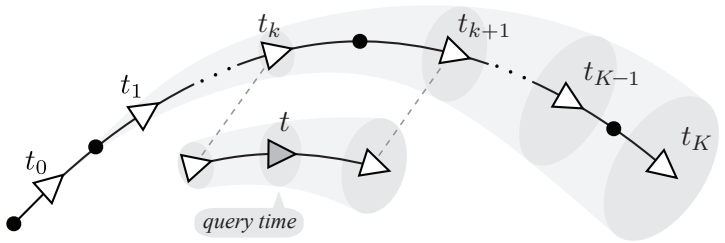


Figure 6.1: This figure shows a factor-graph (Dellaert and Kaess, 2006) depiction of an *exactly-sparse*, continuous-time, Gaussian-process prior, $\mathbf{x}(t) \sim \mathcal{GP}(\check{\mathbf{x}}(t), \check{\mathbf{P}}(t,t'))$, generated from a *Markovian* stochastic differential equation. Barfoot et al. (2014) show that the Markov property can be exploited in the batch continuous-time context in order to generate this type of prior, which has only K binary smoothing factors between consecutive measurement times (plus a unary factor for the initial condition). The triangles are used to depict queries of the trajectory state, $\mathbf{x}(t)$, at the measurement times and the factors (i.e., squared error terms) are represented by black dots. The most pleasing result of the GP-approach is that it provides a principled method of interpolating the trajectory (both mean and covariance) at any time, by storing the mean (triangles) and covariance (shaded ellipses) at the measurement times.

et al. (2014) revived the GP approach by making the retrospective observation that the Markov property exploited by classical (discrete- and continuous-time) recursive methods (Kalman, 1960; Kalman and Bucy, 1961) can also be exploited in the batch continuous-time context. Using LTV SDEs driven by white noise, Barfoot et al. (2014) derive a class of *exactly sparse* GP priors; the associated sparse set of prior costs can be connected to factor-graph theory (Kaess et al., 2008, 2012; Rosen et al., 2014), see Figure 6.1, and used in incremental estimators (Yan et al., 2014).

These prior works on applying GP regression to batch continuous-time trajectory estimation have demonstrated both a method to generate *exactly sparse* GP priors over trajectories in a *vectorspace* (Barfoot et al., 2014) and a method to make the (general, slow) GP approach work for trajectories in *three-dimensional space* (Tong et al., 2014). Building upon these works, the two major contributions of this chapter are: (i) the derivation of a class of exactly sparse GP priors generated from *nonlinear* time-varying SDEs, and (ii) a piecewise, *linear* GP prior that is *both* exactly sparse and handles trajectories belonging to the $SE(3)$ manifold.

Recalling the methods described in Chapter 5, we note that parametric continuous-time trajectory estimation methods are also relevant to this work. From a machine-learning standpoint, these parametric methods can also be viewed as a solution to the proposed regression problem, where some simplifying assumptions have been made about the form of the continuous function we wish to model. In particular, we note that existing literature

has proposed modelling the *true* motion using: (i) a set of piecewise constant-velocities (Ait-Aider et al., 2006; Davison et al., 2007; Hedborg et al., 2012; Dong and Barfoot, 2012; Anderson and Barfoot, 2013a), (ii) curves formed by splines (Bosse and Zlot, 2009; Bibby and Reid, 2010; Bosse et al., 2012; Zlot and Bosse, 2012, 2014), and (iii) finite sets of temporal basis functions (Furgale et al., 2012; Anderson and Barfoot, 2013b; Lovegrove et al., 2013; Oth et al., 2013; Anderson et al., 2014; Furgale et al., 2015). In addition to determining the *type* of interpolation that is best suited for a specific trajectory estimation problem, these parametric schemes must also make difficult choices regarding the *discretization* used in the model. For example, in the temporal basis function approach, decisions concerning knot spacing (which may even be nonuniform) have direct impacts on the fidelity of the model. In contrast to parametric schemes, the GP approach requires less micromanagement, but relies heavily on the continuous-time prior distribution to provide a smooth and sensible result.

6.2 Background – Gaussian Process Gauss-Newton

In this section, we review the mathematical preliminaries underlying the general *Gaussian Process Gauss-Newton* (GPGN) batch estimator proposed by Tong et al. (2012, 2013). In this continuous-time derivation, we will first consider the *localization-only* case (i.e., the map is known) for the state at the trajectory times, then show how the posterior trajectory can be queried at any time, and lastly, demonstrate how to reintroduce the landmarks parameters to the state and formulate the full STEAM problem.

6.2.1 GP Regression for Trajectory Estimation

We begin the derivation of the GP-regression approach to trajectory localization by formulating the problem we wish to solve. Similar to the probabilistic approach described in (3.65), we define the MAP localization problem

$$\hat{\mathbf{x}}(t) = \underset{\mathbf{x}(t)}{\operatorname{argmax}} \; p(\mathbf{x}(t) \,|\, \mathbf{u}(t), \boldsymbol{\ell}, \mathbf{y}), \tag{6.1}$$

where $\hat{\mathbf{x}}(t)$ is the posterior mean of the continuous-time trajectory, $\mathbf{x}(t)$, the control inputs $\mathbf{u}(t)$ and landmarks $\boldsymbol{\ell}$ are known a priori, and we define our landmark observations using the familiar measurement model,

$$\mathbf{y}_{kj} = \mathbf{g}(\mathbf{x}(t_k), \boldsymbol{\ell}_j) + \mathbf{n}_{kj}, \qquad t_1 < \cdots < t_K, \tag{6.2}$$

where $\mathbf{y}_{kj}$ is the measurement of (known) landmark $\boldsymbol{\ell}_j$ at time t_k, $\mathbf{g}(\cdot)$ is the nonlinear observation model, and $\mathbf{n}_{kj} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{kj})$ is zero-mean Gaussian measurement noise. In the GP-regression approach, the prior distribution of trajectories, $p(\mathbf{x}(t) \,|\, \mathbf{u}(t))$, is represented by using the Gaussian process

$$\mathbf{x}(t) \sim \mathcal{GP}(\check{\mathbf{x}}(t), \check{\mathbf{P}}(t, t')), \qquad t_0 < t, t', \tag{6.3}$$

where $\check{\mathbf{x}}(t)$ is the prior mean function and $\check{\mathbf{P}}(t, t')$ is the prior covariance function. From a machine-learning standpoint, we note that the prior mean function and prior covariance (sometimes referred to as the *kernel*) function can be specified in a variety of manners. However, given the nature of our physical problem, it is prudent to generate the prior from a continuous-time motion model (using the control inputs $\mathbf{u}(t)$).

To provide some intuition on how this can be accomplished, we share the white-noise-on-acceleration (i.e., constant-velocity) prior employed by the majority of the work by Tong et al. (2012, 2013, 2014). Considering a simple 2D trajectory,

$$\mathbf{x}(t) := \mathbf{p}(t) = \begin{bmatrix} x(t) & y(t) & \theta(t) \end{bmatrix}^T \in \mathbb{R}^3, \tag{6.4}$$

a constant-velocity prior is formed by specifying the process model

$$\ddot{\mathbf{x}}(t) = \mathbf{w}(t), \quad \mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_C \delta(t - t')), \quad \mathbf{u}(t) = \mathbf{0}, \tag{6.5}$$

where the process noise, $\mathbf{w}(t)$, is a zero-mean Gaussian process with (symmetric, positive-definite) power-spectral-density matrix, $\mathbf{Q}_C$. Avoiding the laborious details, we note that integrating this expression twice (using the Heaviside function to handle the Dirac delta function) results in the GP prior mean and covariance functions (Tong et al., 2013):

$$\check{\mathbf{x}}(t) = \check{\mathbf{x}}(t_0), \quad \check{\mathbf{P}}(t, t') = \left(\frac{\min(t, t')^2 \max(t, t')}{2} - \frac{\min(t, t')^3}{6} \right) \mathbf{Q}_C. \tag{6.6}$$

In order to connect these continuous-time models with the batch estimation machinery described in Chapter 3, we first discretize our continuous-time functions at the measurement times, t_k, such that

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}(t_0) \\ \vdots \\ \mathbf{x}(t_K) \end{bmatrix}, \quad \check{\mathbf{x}} = \begin{bmatrix} \check{\mathbf{x}}(t_0) \\ \vdots \\ \check{\mathbf{x}}(t_K) \end{bmatrix}, \quad \check{\mathbf{P}} = \begin{bmatrix} \check{\mathbf{P}}(t_i, t_j) \end{bmatrix}_{ij}. \tag{6.7}$$

Using the same vector and matrix discretization schemes, we also define the posterior mean, $\hat{\mathbf{x}}$, posterior covariance, $\hat{\mathbf{P}}$, operating point, $\bar{\mathbf{x}}$, and state perturbation, $\delta\mathbf{x}$. As

usual, we aim to construct an iterative MAP estimator by linearizing the measurement models about our best guess, $\bar{\mathbf{x}}$, and solving for the optimal update, $\delta\mathbf{x}^\star$, where we define $\hat{\mathbf{x}} = \bar{\mathbf{x}} + \delta\mathbf{x}^\star$. This is accomplished (without a specific set of prior functions) by using Bayesian inference. We begin by writing the joint likelihood

$$p\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\check{\mathbf{x}}\\ \bar{\mathbf{g}} + \mathbf{G}(\check{\mathbf{x}} - \bar{\mathbf{x}})\end{bmatrix}, \begin{bmatrix}\check{\mathbf{P}} & \check{\mathbf{P}}\mathbf{G}^T\\ \mathbf{G}\check{\mathbf{P}} & \mathbf{G}\check{\mathbf{P}}\mathbf{G}^T + \mathbf{R}\end{bmatrix}\right), \tag{6.8}$$

where the measurement model is linearized about our best guess,

$$\mathbf{y} = \begin{bmatrix}\mathbf{y}_{00}\\ \vdots\\ \mathbf{y}_{KL}\end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix}\mathbf{g}(\mathbf{x}(t_0))\\ \vdots\\ \mathbf{g}(\mathbf{x}(t_K))\end{bmatrix}, \quad \bar{\mathbf{g}} = \mathbf{g}|_{\bar{\mathbf{x}}}, \quad \mathbf{G} = \left.\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}}}, \quad \mathbf{R} = \operatorname{diag}\left(\mathbf{R}_{00}, \ldots, \mathbf{R}_{KL}\right). \tag{6.9}$$

Conditioning the state on the measurements, we find the Gaussian posterior:

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}\Bigg(\underbrace{\check{\mathbf{x}} + \check{\mathbf{P}}\mathbf{G}^T\left(\mathbf{G}\check{\mathbf{P}}\mathbf{G}^T + \mathbf{R}\right)^{-1}\left(\mathbf{y} - \bar{\mathbf{g}} - \mathbf{G}\left(\check{\mathbf{x}} - \bar{\mathbf{x}}\right)\right)}_{=\hat{\mathbf{x}}\text{, the posterior mean}}, \underbrace{\check{\mathbf{P}} - \check{\mathbf{P}}\mathbf{G}^T\left(\mathbf{G}\check{\mathbf{P}}\mathbf{G}^T + \mathbf{R}\right)^{-1}\mathbf{G}\check{\mathbf{P}}}_{=\hat{\mathbf{P}}\text{, the posterior covariance}}\Bigg). \tag{6.10}$$

Using the perturbation, $\hat{\mathbf{x}} = \bar{\mathbf{x}} + \delta\mathbf{x}^\star$, and the Sherman-Morrison-Woodbury identities,

$$\mathbf{AB}\left(\mathbf{CAB} + \mathbf{D}\right)^{-1} \equiv \left(\mathbf{A}^{-1} + \mathbf{BD}^{-1}\mathbf{C}\right)^{-1}\mathbf{BD}^{-1}, \tag{6.11}$$

$$\mathbf{A} - \mathbf{AB}\left(\mathbf{CAB} + \mathbf{D}\right)^{-1}\mathbf{CA} \equiv \left(\mathbf{A}^{-1} + \mathbf{BD}^{-1}\mathbf{C}\right)^{-1}, \tag{6.12}$$

we rearrange the posterior mean expression – via (6.11) – to find that

$$\left(\check{\mathbf{P}}^{-1} + \mathbf{G}^T\mathbf{R}^{-1}\mathbf{G}\right)\delta\mathbf{x}^\star = \check{\mathbf{P}}^{-1}(\check{\mathbf{x}} - \bar{\mathbf{x}}) + \mathbf{G}^T\mathbf{R}^{-1}(\mathbf{y} - \bar{\mathbf{g}}). \tag{6.13}$$

Noting that the posterior covariance expression – via (6.12) – is

$$\hat{\mathbf{P}} = \left(\check{\mathbf{P}}^{-1} + \mathbf{G}^T\mathbf{R}^{-1}\mathbf{G}\right)^{-1}, \tag{6.14}$$

we recognize that (6.13) can be viewed as the solution to the associated *maximum a posteriori* (MAP) problem. Our connection to the Gauss-Newton machinery is complete, noting that we can solve iteratively for $\delta\mathbf{x}^\star$ and then update the guess according to $\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} + \delta\mathbf{x}^\star$; upon convergence, we set $\hat{\mathbf{x}} = \bar{\mathbf{x}}$.

Concerning sparsity, we recall from our previous analysis of the batch discrete-time problem (see Section 3.1.3) that $\mathbf{G}^T\mathbf{R}^{-1}\mathbf{G}$ is block-diagonal (this is the upper-left corner of the typical *arrowhead matrix* in the full SLAM problem). Note that in this derivation we have not assumed a specific form for $\check{\mathbf{P}}(t,t')$; therefore, the inverse covariance, $\check{\mathbf{P}}^{-1}$, may in fact be *dense* if proper care is not taken when modelling the GP prior functions – this is the case for the prior in (6.6). A naive implementation will therefore lead to a complexity of $O(K^3)$ to solve the system of equations, where K is the number of measurement times.

6.2.2 GP Regression for STEAM

Transitioning to the full STEAM problem is a simple matter. Considering the landmarks to now be *unknown*, we formulate the joint state vector and Jacobians:

$$\mathbf{z} := \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\ell} \end{bmatrix}, \quad \boldsymbol{\ell} := \begin{bmatrix} \boldsymbol{\ell}_1 \\ \vdots \\ \boldsymbol{\ell}_L \end{bmatrix}, \quad \mathbf{G} := \begin{bmatrix} \mathbf{G}_{\mathbf{x}} & \mathbf{G}_{\boldsymbol{\ell}} \end{bmatrix}, \quad \mathbf{G}_{\mathbf{x}} := \left.\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}}}, \quad \mathbf{G}_{\boldsymbol{\ell}} := \left.\frac{\partial \mathbf{g}}{\partial \boldsymbol{\ell}}\right|_{\bar{\boldsymbol{\ell}}}. \tag{6.15}$$

Noting the similarity of our system of equations in (6.13), to the discrete-time batch estimation in (3.25), it is straightforward to determine that the Gauss-Newton STEAM estimator can be formulated using the update equation

$$\left(\overbrace{\begin{bmatrix} \check{\mathbf{P}}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}^{\mathbf{A}_{\text{pri}}} + \overbrace{\mathbf{G}^T\mathbf{R}^{-1}\mathbf{G}}^{\mathbf{A}_{\text{meas}}} \right) \delta\mathbf{z}^\star = \begin{bmatrix} \check{\mathbf{P}}^{-1}(\check{\mathbf{x}} - \bar{\mathbf{x}}) \\ \mathbf{0} \end{bmatrix} + \mathbf{G}^T\mathbf{R}^{-1}(\mathbf{y} - \bar{\mathbf{g}}), \tag{6.16}$$

where the solution for the optimal state perturbation $\delta\mathbf{z}^\star$ can be used iteratively to converge our best guess, $\bar{\mathbf{z}}$, to the MAP solution, $\hat{\mathbf{z}}$.

Recalling the discussion of sparsity in Section 3.1.3, we note that our state update equation has a familiar 2×2 block-form – see (3.28). In particular, we note that $\mathbf{A}_{\text{meas}}$ is the familiar *arrowhead matrix*. However, in contrast to the discrete-time solution, the contribution of our prior, $\check{\mathbf{P}}^{-1}$, is not necessarily block-tridiagonal. Using the Schur complement, or Cholesky decomposition, we can exploit either the structure of $\mathbf{A}_{xx}$ or $\mathbf{A}_{\ell\ell}$. In cases where $\check{\mathbf{P}}^{-1}$ is dense (or unknown), we typically exploit the block-diagonal nature of $\mathbf{A}_{\ell\ell}$ to solve the STEAM problem in $O(K^3 + K^2L)$ time.

6.2.3 Querying the Trajectory

In order to query the posterior mean and covariance of the continuous-time trajectory at other times of interest, τ, we employ the use of the standard linear GP interpolation equations (Rasmussen and Williams, 2006; Tong et al., 2013),

$$\hat{\mathbf{x}}(\tau) = \check{\mathbf{x}}(\tau) + \check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}(\hat{\mathbf{x}} - \check{\mathbf{x}}), \tag{6.17a}$$

$$\hat{\mathbf{P}}(\tau,\tau) = \check{\mathbf{P}}(\tau,\tau) + \check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}\Big(\hat{\mathbf{P}} - \check{\mathbf{P}}\Big)\check{\mathbf{P}}^{-T}\check{\mathbf{P}}(\tau)^T, \tag{6.17b}$$

where $\check{\mathbf{P}}(\tau) = \Big[\check{\mathbf{P}}(\tau,t_0) \quad \cdots \quad \check{\mathbf{P}}(\tau,t_K)\Big]$. In general, the posterior at some query times, $\hat{\mathbf{x}}(\tau)$, could also be found by inserting them into the state vector, $\mathbf{x}$, before performing the batch optimization; however, the (naive) computational cost of solving (6.13) would increase to $O((K+J)^3)$, where J is the number of query times. In contrast, the above interpolation equations have a (naive) complexity of only $O(K^2J)$. Notably, either method will result in the same interpolated posterior quantities (for a linear GP prior).

6.2.4 Interpolating Measurement Times

Thus far, we have derived a principled continuous-time batch estimation scheme to solve for the trajectory state, $\mathbf{x}(t_k)$, at every measurement time, t_k, $k = 1 \dots K$, and then query for other times of interest, $\mathbf{x}(\tau)$ – all without approximation. Given the ability to interpolate the mean, via (6.17a), storing the state at every measurement time may be excessive, especially in a scenario where the measurement rate is high in comparison to the smoothness of the robot kinematics (e.g., a 1000 Hz IMU or individually timestamped lidar measurements, mounted on a slow indoor platform).

Tong et al. (2013) discuss a heuristic scheme to remove some of the measurement times from the initial solve, which reduces computational cost at the expense of accuracy. It is proposed that by defining $\mathbf{x}$ to discretize the trajectory at some smaller set of *keytimes*, $t_n, n = 0 \dots N$, which may or may not align with some subset of the measurement times, the interpolation equation can be used to modify our measurement model as follows,

$$\mathbf{y}_{kj} = \mathbf{g}(\mathbf{x}(t_k), \boldsymbol{\ell}_j) = \mathbf{g}\Big(\check{\mathbf{x}}(t_k) + \check{\mathbf{P}}(t_k)\check{\mathbf{P}}^{-1}(\mathbf{x} - \check{\mathbf{x}}), \boldsymbol{\ell}_j\Big), \tag{6.18}$$

where the trajectory at the measurement time, $\mathbf{x}(t_k)$, is now queried from the compressed (*keytime*) state vector. The effect on the Jacobian, $\mathbf{G}$, is that each block-row now has multiple non-zero block columns (based on the sparsity of $\check{\mathbf{P}}(t_k)\check{\mathbf{P}}^{-1}$).

An important intuition is that the interpolated state at some measurement time, t_k, depends only on the state estimate, $\mathbf{x}$, and the prior functions, $\check{\mathbf{x}}(t)$ and $\check{\mathbf{P}}(t,t')$. Therefore, the effect of estimating $\mathbf{x}$ at some reduced number of times is that we obtain a smoothed solution; detailed trajectory information provided by high frequency measurements between *keytimes* is lost. Although this detail information is smoothed over, there are obvious computational savings in having a smaller state and a subtle benefit regarding the prevention of overfitting measurements.

6.3 Estimating Trajectories in $\mathbb{R}^N$

6.3.1 Background – A Class of Exactly Sparse GP Priors

In this section we review the material introduced by Barfoot et al. (2014) to generate a useful class of *exactly sparse* GP priors from *linear time-varying* (LTV) *stochastic differential equations* (SDE) of the form

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{v}(t) + \mathbf{L}(t)\mathbf{w}(t), \quad \mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_C\,\delta(t-t')), \tag{6.19}$$

where $\mathbf{x}(t)$ is our continuous-time trajectory, $\mathbf{v}(t)$ is a (known) exogenous input, $\mathbf{w}(t)$ is the process noise, and $\mathbf{F}(t)$, $\mathbf{L}(t)$ are time-varying system matrices. The general solution to this LTV SDE is

$$\mathbf{x}(t) = \boldsymbol{\Phi}(t,t_0)\mathbf{x}(t_0) + \int_{t_0}^{t} \boldsymbol{\Phi}(t,s)\left(\mathbf{v}(s) + \mathbf{L}(s)\mathbf{w}(s)\right) ds, \tag{6.20}$$

where $\boldsymbol{\Phi}(t,s)$ is known as the *transition matrix*.

Mean and Covariance Functions

Using the first and second moments of (6.20), we note the mean function

$$\check{\mathbf{x}}(t) = E[\mathbf{x}(t)] = \boldsymbol{\Phi}(t,t_0)\check{\mathbf{x}}_0 + \int_{t_0}^{t} \boldsymbol{\Phi}(t,s)\mathbf{v}(s)\, ds, \tag{6.21}$$

and covariance function,

$$\begin{aligned}\check{\mathbf{P}}(t,t') &= E\left[(\mathbf{x}(t) - \check{\mathbf{x}}(t))(\mathbf{x}(t') - \check{\mathbf{x}}(t'))^T\right] \\ &= \boldsymbol{\Phi}(t,t_0)\check{\mathbf{P}}_0\boldsymbol{\Phi}(t',t_0)^T + \int_{t_0}^{\min(t,t')} \boldsymbol{\Phi}(t,s)\mathbf{L}(s)\mathbf{Q}_C\mathbf{L}(s)^T\boldsymbol{\Phi}(t',s)^T\, ds,\end{aligned} \tag{6.22}$$

that fully specify a prior distribution of trajectories, based on the LTV SDE in (6.19). In order to split up the integrals at the sequence of measurement times, $t_0 < t_1 < t_2 < \cdots < t_K$, we define the quantities

$$\mathbf{v}_k(\tau) := \int_{t_k}^{\tau} \boldsymbol{\Phi}(\tau, s)\mathbf{v}(s)\, ds, \tag{6.23a}$$

$$\mathbf{Q}_k(\tau) := \int_{t_k}^{\tau} \boldsymbol{\Phi}(\tau, s)\mathbf{L}(s)\mathbf{Q}_C\mathbf{L}(s)^T\boldsymbol{\Phi}(\tau, s)^T\, ds, \tag{6.23b}$$

and note that the expressions can be simplified using the *recursive* schemes

$$\check{\mathbf{x}}(\tau) = \boldsymbol{\Phi}(\tau, t_k)\check{\mathbf{x}}(t_k) + \mathbf{v}_k(\tau), \qquad t_k < \tau < t_{k+1} \tag{6.24a}$$

$$\check{\mathbf{P}}(\tau, \tau') = \boldsymbol{\Phi}(\tau, t_k)\check{\mathbf{P}}(t_k, t_k)\boldsymbol{\Phi}(\tau', t_k)^T + \mathbf{Q}_k(\min(\tau, \tau')), \qquad t_k < \tau, \tau' < t_{k+1} \tag{6.24b}$$

with the initial condition, $\mathbf{x}(t_0) \sim \mathcal{N}(\check{\mathbf{x}}_0, \check{\mathbf{P}}_0)$. Recalling the state discretization at the measurement times, $\mathbf{x}$, we can then write the prior in *lifted form*

$$\mathbf{x} \sim \mathcal{N}\left(\check{\mathbf{x}}, \check{\mathbf{P}}\right) = \mathcal{N}\left(\mathbf{F}\mathbf{v}, \mathbf{F}\mathbf{Q}\mathbf{F}^T\right), \tag{6.25}$$

where $\mathbf{Q} = \operatorname{diag}\left(\check{\mathbf{P}}_0, \mathbf{Q}_0(t_1), \ldots, \mathbf{Q}_{K-1}(t_K)\right)$ and

$$\mathbf{F} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\Phi}(t_1, t_0) & \mathbf{1} & \cdots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\Phi}(t_2, t_0) & \boldsymbol{\Phi}(t_2, t_1) & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\Phi}(t_{K-1}, t_0) & \boldsymbol{\Phi}(t_{K-1}, t_1) & \cdots & \mathbf{1} & \mathbf{0} \\ \boldsymbol{\Phi}(t_K, t_0) & \boldsymbol{\Phi}(t_K, t_1) & \cdots & \boldsymbol{\Phi}(t_K, t_{K-1}) & \mathbf{1} \end{bmatrix}, \; \mathbf{v} = \begin{bmatrix} \check{\mathbf{x}}_0 \\ \mathbf{v}_0(t_1) \\ \vdots \\ \mathbf{v}_{K-1}(t_K) \end{bmatrix}. \tag{6.26}$$

Perhaps the most important result of this new structure is that

$$\check{\mathbf{P}}^{-1} = (\mathbf{F}\mathbf{Q}\mathbf{F}^T)^{-1} = \mathbf{F}^{-T}\mathbf{Q}^{-1}\mathbf{F}^{-1}, \tag{6.27}$$

is *exactly* block-tridiagonal, where the inverse of the *lifted* transition matrix is

$$\mathbf{F}^{-1} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ -\boldsymbol{\Phi}(t_1, t_0) & \mathbf{1} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{\Phi}(t_2, t_1) & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \cdots & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & -\boldsymbol{\Phi}(t_K, t_{K-1}) & \mathbf{1} \end{bmatrix}, \tag{6.28}$$

which is a block-lower-bidiagonal matrix.

Querying the Trajectory

Recalling the linear prediction equations presented in (6.17), Barfoot et al. (2014) make the keen observation that by applying this sparse structure, the vital term, $\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}$, has exactly two non-zero block-columns

$$\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1} = \Big[\mathbf{0} \;\; \cdots \;\; \mathbf{0} \;\; \underbrace{\boldsymbol{\Lambda}(\tau)}_{\substack{\text{block}\\ \text{col. } k}} \;\; \underbrace{\boldsymbol{\Omega}(\tau)}_{\substack{\text{block}\\ \text{col. } k+1}} \;\; \mathbf{0} \;\; \cdots \;\; \mathbf{0}\Big], \tag{6.29}$$

where $t_k \leq \tau < t_{k+1}$,

$$\boldsymbol{\Lambda}(\tau) = \boldsymbol{\Phi}(\tau, t_k) - \boldsymbol{\Omega}(\tau)\boldsymbol{\Phi}(t_{k+1}, t_k), \quad \boldsymbol{\Omega}(\tau) = \mathbf{Q}_k(\tau)\boldsymbol{\Phi}(t_{k+1}, \tau)^T\mathbf{Q}_k(t_{k+1})^{-1}. \tag{6.30}$$

Inserting this into (6.17), the interpolation equations simplify to

$$\hat{\mathbf{x}}(\tau) = \check{\mathbf{x}}(\tau) + \begin{bmatrix}\boldsymbol{\Lambda}(\tau) & \boldsymbol{\Omega}(\tau)\end{bmatrix}\begin{bmatrix}\hat{\mathbf{x}}_k - \check{\mathbf{x}}_k \\ \hat{\mathbf{x}}_{k+1} - \check{\mathbf{x}}_{k+1}\end{bmatrix}, \tag{6.31a}$$

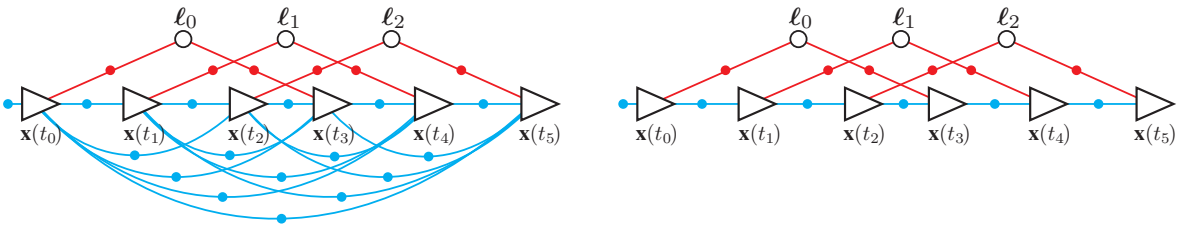
$$\hat{\mathbf{P}}(\tau,\tau) = \check{\mathbf{P}}(\tau,\tau) + \begin{bmatrix}\boldsymbol{\Lambda}(\tau) & \boldsymbol{\Omega}(\tau)\end{bmatrix}\left(\begin{bmatrix}\hat{\mathbf{P}}_{k,k} & \hat{\mathbf{P}}_{k,k+1} \\ \hat{\mathbf{P}}_{k+1,k} & \hat{\mathbf{P}}_{k+1,k+1}\end{bmatrix} - \begin{bmatrix}\check{\mathbf{P}}_{k,k} & \check{\mathbf{P}}_{k,k+1} \\ \check{\mathbf{P}}_{k+1,k} & \check{\mathbf{P}}_{k+1,k+1}\end{bmatrix}\right)\begin{bmatrix}\boldsymbol{\Lambda}(\tau)^T \\ \boldsymbol{\Omega}(\tau)^T\end{bmatrix}, \tag{6.31b}$$

where the prior quantities $\check{\mathbf{x}}(\tau)$ and $\check{\mathbf{P}}(\tau,\tau)$ are computed using (6.24). Notably, the final *sparse* interpolation scheme involves *only* the temporally adjacent terms (i.e., terms associated with times t_k and t_{k+1}); this can be viewed as a product of the *Markov* property. Furthermore, the interpolation can be computed efficiently, in $O(1)$ time, and is derived with no approximations.

Exactly Sparse GP STEAM

Returning to the STEAM problem in (6.16), we note that by using a GP prior generated from an LTV SDE, we have that $\check{\mathbf{P}}^{-1} = \mathbf{F}^{-T}\mathbf{Q}^{-1}\mathbf{F}^{-1}$, which is block-tridiagonal. Recalling that $\mathbf{G}^T\mathbf{R}^{-1}\mathbf{G}$ (for the STEAM problem) is the *arrowhead matrix*, a factor-graph depiction of the full problem sparsity is presented in Figure 6.2. The cost of STEAM, using an *exactly sparse* GP prior, is either of order $O(L^3+L^2K+J)$ or $O(K^3+K^2L+J)$, depending on the way we choose to exploit $\mathbf{A}_{xx}$ or $\mathbf{A}_{\ell\ell}$.

Recalling that the block-tridiagonal term, $\mathbf{F}^{-T}\mathbf{Q}^{-1}\mathbf{F}^{-1}$, is generated from our continuous-time motion model, we note that this is the exact form of our discrete-time SLAM problem in (3.25) – despite having used no approximations and a *continuous-time* motion model.



(a) General dense (non-Markovian) GP prior.

(b) Exactly sparse (*Markovian*) GP prior.

Figure 6.2: This figure illustrates factor-graph representations (Dellaert and Kaess, 2006) of our STEAM problem using a general (dense) GP prior and an *exactly sparse* GP prior (exploiting the *Markov* property). Triangles are used to represent temporal samples of the trajectory state at the measurement times, $\mathbf{x}(t_k)$, and hollow circles are used to represent the landmark parameters, $\boldsymbol{\ell}_j$. The red (binary) factors represent costs associated with the visual measurements, $\mathbf{y}_{kj}$, and blue factors represent costs (i.e., smoothing terms) associated with the GP prior; note the initial blue (unary) factor is a prior over the trajectory variables at time t_0. The crisscross pattern of (red) landmark associations depict how a sweeping-type sensor might behave across two 'frames'; the landmarks are observed in an initial sweep, (t_0, t_1, t_2), and then again, (t_3, t_4, t_5).

Furthermore, Barfoot et al. (2014) note that when the motion and observation models are *linear*, the GP approach is equivalent to classical fixed-interval smoothing. Arguably, this approach can be viewed as a reinterpretation of the standard discrete-time SLAM formulation; however, we recall that the continuous-time GP interpretation adds an important piece of information that is missing from the discrete-time counterpart: an interpolation equation (for both mean and covariance) that is derived directly from the chosen continuous-time motion model. It follows that the time-tested, discrete-time batch SLAM solution can be viewed as a *special case* of the GP-regression approach to STEAM, where the motion model has been discretized (i.e., performs an Euler step on the continuous-time model). This relationship between discrete and continuous-time approaches has also been discovered in work on recursive estimation (Särkkä, 2006).

An important intuition we draw from this approach is that choosing a LTV SDE that can be written in the form of (6.19) *dictates* the parameters that we must estimate in our trajectory, $\mathbf{x}(t)$. In essence, to maintain the block-tridiagonal sparsity we must choose a *Markovian* trajectory representation that matches the form of our prior. For example, the white-noise-on-acceleration prior presented in (6.5) does not exhibit this sparsity because Tong et al. (2013) only estimated *position*, $\mathbf{x}(t) = \mathbf{p}(t)$. Instead, to use a prior on acceleration, we must estimate *both* position and velocity, $\mathbf{x}(t) = \begin{bmatrix} \mathbf{p}(t)^T & \dot{\mathbf{p}}(t)^T \end{bmatrix}^T$, or even just velocity, $\mathbf{x}(t) = \dot{\mathbf{p}}(t)$.

Linear Constant-Velocity Prior Example

In this section, we set up the exactly sparse version of the 'constant-velocity' GP prior presented in (6.5); this prior will be used later in our 2D mobile robot experiment. Recalling that we set $\ddot{\mathbf{p}}(t) = \mathbf{w}(t)$, where $\mathbf{p}(t) = \begin{bmatrix} x(t) & y(t) & \theta(t) \end{bmatrix}^T$, we now write the prior in the form of (6.19) with

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{p}(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix}, \ \mathbf{F}(t) = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \ \mathbf{v}(t) = \mathbf{0}, \ \mathbf{L}(t) = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix}. \tag{6.32}$$

The resulting expression is a linear, time-invariant (LTI) SDE, with the associated *transition matrix*

$$\mathbf{\Phi}(t, s) = \begin{bmatrix} \mathbf{1} & (t-s)\mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}. \tag{6.33}$$

Substituting these terms into (6.23b), it is straightforward to find that

$$\mathbf{Q}_k(\tau) = \begin{bmatrix} \frac{1}{3}\Delta\tau^3\mathbf{Q}_C & \frac{1}{2}\Delta\tau^2\mathbf{Q}_C \\ \frac{1}{2}\Delta\tau^2\mathbf{Q}_C & \Delta\tau\mathbf{Q}_C \end{bmatrix}, \tag{6.34}$$

where $\Delta\tau := \tau - t_k$ and the inverse is

$$\mathbf{Q}_k(\tau)^{-1} = \begin{bmatrix} 12\Delta\tau^{-3}\mathbf{Q}_C^{-1} & -6\Delta\tau^{-2}\mathbf{Q}_C^{-1} \\ -6\Delta\tau^{-2}\mathbf{Q}_C^{-1} & 4\Delta\tau^{-1}\mathbf{Q}_C^{-1} \end{bmatrix}. \tag{6.35}$$

Using these expressions, we are able to analytically compute all of the terms in $\mathbf{P}^{-1}$ needed for our batch continuous-time STEAM update equation.

6.3.2 Nonlinear Time-Varying Stochastic Differential Equations

At this point, we begin to describe our novel contributions towards the GP-regression approach to batch continuous-time trajectory estimation. In reality, most systems are inherently nonlinear and cannot be accurately described by a LTV SDE in the form of (6.19). Moving forward, we show how these results concerning sparsity can be applied to nonlinear, time-varying (NTV) stochastic differential equations (SDE) of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t)), \tag{6.36}$$

where $\mathbf{f}(\cdot)$ is a nonlinear function, $\mathbf{x}(t)$ is the state, $\mathbf{u}(t)$ is a (known) exogenous input, and $\mathbf{w}(t)$ is white process noise. Notably, unlike the previous formulations presented in this thesis, we have included the noise parameter, $\mathbf{w}(t)$, as an input to the nonlinear motion

model – this allows for greater freedom in describing how process noise is injected into $\mathbf{x}(t)$. To perform GP regression with a nonlinear process model, we begin by linearizing the SDE about a continuous-time operating point $\bar{\mathbf{x}}(t)$,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t)) \approx \mathbf{f}(\bar{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{0}) + \mathbf{F}(t)(\mathbf{x}(t) - \bar{\mathbf{x}}(t)) + \mathbf{L}(t)\mathbf{w}(t), \tag{6.37}$$

where

$$\mathbf{F}(t) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{0}}, \quad \mathbf{L}(t) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{w}}\right|_{\bar{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{0}}. \tag{6.38}$$

Setting

$$\mathbf{v}(t) = \mathbf{f}(\bar{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{0}) - \mathbf{F}(t)\bar{\mathbf{x}}(t) \tag{6.39}$$

lets us rewrite (6.37) in the familiar LTV SDE form,

$$\dot{\mathbf{x}}(t) \approx \mathbf{F}(t)\mathbf{x}(t) + \mathbf{v}(t) + \mathbf{L}(t)\mathbf{w}(t), \tag{6.40}$$

where $\mathbf{F}(t)$, $\mathbf{v}(t)$, and $\mathbf{L}(t)$ are known functions of time, since $\bar{\mathbf{x}}(t)$ is known. Setting the operating point, $\bar{\mathbf{x}}(t)$, to our best guess of the underlying trajectory at each iteration of GP regression, we note a similarity in nature to the discrete-time, recursive method of Bell (1994); in essence, our approach offers a continuous-discrete version of the Gauss-Newton estimator, using the Gaussian-process-regressor type of approximate bridging between the measurement times.

Mean and Covariance Functions

Although the equations for calculating the mean, $\check{\mathbf{x}}$, and covariance, $\check{\mathbf{P}}$, remain the same as in (6.21) and (6.22), there are a few algorithmic differences and issues that arise due to the new dependence on the continuous-time operating point, $\bar{\mathbf{x}}(t)$. One difference (in contrast to using a LTV SDE), is that we must now recalculate $\check{\mathbf{x}}$ and $\check{\mathbf{P}}$ at each iteration of the optimization (since the linearization point is updated). The main algorithmic issue that presents itself is that the calculation of $\check{\mathbf{x}}$ and $\check{\mathbf{P}}$ require integrations involving $\mathbf{F}(t)$, $\mathbf{v}(t)$, and $\mathbf{L}(t)$, which in turn require the evaluation of $\bar{\mathbf{x}}(t)$ over the time period $t \in [t_0, t_K]$. Since an estimate of the posterior mean, $\bar{\mathbf{x}}$, is stored only at times of interest, we must make use of the efficient (for our particular choice of process model) GP interpolation equation derived in Section 6.3.1,

$$\bar{\mathbf{x}}(\tau) = \check{\mathbf{x}}(\tau) + \check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}(\bar{\mathbf{x}} - \check{\mathbf{x}}). \tag{6.41}$$

The problem is that the above interpolation depends (once again) on $\check{\mathbf{x}}$ and $\check{\mathbf{P}}$, which are the prior variables for which we want to solve. To rectify this circular dependence, we take advantage of the iterative nature of GP regression and choose to evaluate (6.41) using the values of $\check{\mathbf{x}}(\tau)$, $\check{\mathbf{x}}$, and $\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}$ from the previous iteration.

Another issue with nonlinear process models is that identifying an analytical expression for the state transition matrix, $\boldsymbol{\Phi}(t,s)$ (which is dependent on the form of $\mathbf{F}(t)$), can be very challenging. Fortunately, the transition matrix can also be calculated numerically via the integration of the normalized fundamental matrix, $\boldsymbol{\Upsilon}(t)$, where

$$\dot{\boldsymbol{\Upsilon}}(t) = \mathbf{F}(t)\boldsymbol{\Upsilon}(t), \quad \boldsymbol{\Upsilon}(0) = \mathbf{1}. \tag{6.42}$$

Storing $\boldsymbol{\Upsilon}(t)$ at times of interest, the transition matrix can then be computed using

$$\boldsymbol{\Phi}(t,s) = \boldsymbol{\Upsilon}(t)\boldsymbol{\Upsilon}(s)^{-1}. \tag{6.43}$$

In contrast to the LTV SDE system, using a nonlinear process model causes additional computational costs (primarily due to numerical integrations), but complexity remains linear in the length of the trajectory (at each iteration), and therefore continues to be computationally tractable.

Querying the Trajectory

In this section, we discuss the algorithmic details of the GP interpolation procedure as it pertains to a NTV SDE process model. Recall the standard linear GP interpolation formulas presented in (6.17), for a single query time, $t_k \leq \tau < t_{k+1}$:

$$\begin{aligned}
\hat{\mathbf{x}}(\tau) &= \check{\mathbf{x}}(\tau) + \check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}(\hat{\mathbf{x}} - \check{\mathbf{x}}), \\
\hat{\mathbf{P}}(\tau,\tau) &= \check{\mathbf{P}}(\tau,\tau) + \check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}\Big(\hat{\mathbf{P}} - \check{\mathbf{P}}\Big)\check{\mathbf{P}}^{-T}\check{\mathbf{P}}(\tau)^T.
\end{aligned}$$

The final iteration of GP regression (where $\hat{\mathbf{x}} = \bar{\mathbf{x}}$), provides values for $\check{\mathbf{x}}$, $\check{\mathbf{P}}$, $\hat{\mathbf{x}}$, and $\hat{\mathbf{P}}$; however, obtaining values for $\check{\mathbf{x}}(\tau)$, $\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}$ (recall the sparse structure in (6.29)), and $\check{\mathbf{P}}(\tau,\tau)$ is not trivial. The suggestion made previously was to use values from the previous (or in this case, final) iteration. Thus far, the method of storing these continuous-time functions has been left ambiguous.

The naive way to store $\check{\mathbf{x}}(\tau)$, $\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}$, and $\check{\mathbf{P}}(\tau,\tau)$ is to keep the values at all numerical integration timesteps; the memory requirement of this is proportional to the length of the trajectory. During the optimization procedure this naive method may in fact be

preferable as it reduces computation time in lieu of additional storage (which is fairly cheap using current technology). However, for long-term storage, a method that uses a smaller memory footprint (at the cost of additional computation) may be desirable. The remainder of this section will focus on identifying the minimal storage requirements, such that queries remain $O(1)$ complexity.

We begin by examining the mean function; substituting (6.24a) and (6.39), into the GP interpolation formula above, we have

$$\hat{\mathbf{x}}(\tau) = \boldsymbol{\Phi}(\tau, t_k)\check{\mathbf{x}}_k + \int_{t_k}^{\tau} \boldsymbol{\Phi}(\tau, s)\big(\mathbf{f}(\hat{\mathbf{x}}(s), \mathbf{u}(s), \mathbf{0}) - \mathbf{F}(s)\hat{\mathbf{x}}(s)\big)\, ds + \check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}(\hat{\mathbf{x}} - \check{\mathbf{x}}). \quad (6.45)$$

It is straightforward to see how $\hat{\mathbf{x}}(\tau)$ can be simultaneously numerically integrated with the normalized fundamental matrix from (6.42), provided we can evaluate the term $\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}$. Although we chose to examine the mean function, a similar conclusion can be drawn by examining the covariance function in (6.24b). Recalling the sparse structure of $\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}$ for an LTV SDE process model, where

$$\boldsymbol{\Lambda}(\tau) = \boldsymbol{\Phi}(\tau, t_k) - \boldsymbol{\Omega}(\tau)\boldsymbol{\Phi}(t_{k+1}, t_k), \quad \boldsymbol{\Omega}(\tau) = \mathbf{Q}_k(\tau)\boldsymbol{\Phi}(t_{k+1}, \tau)^T\mathbf{Q}_k(t_{k+1})^{-1},$$

we are able to draw two conclusions. First, in the case that an analytical expression for $\boldsymbol{\Phi}(t, s)$ is unavailable, we must store $\boldsymbol{\Upsilon}(t_k)$ at the times of interest $t_k, k = 1 \dots K$, since any numerical integration will involve 'future' values of $\boldsymbol{\Upsilon}(t)$ (via the evaluation of $\boldsymbol{\Phi}(t_{k+1}, t_k)$, $\boldsymbol{\Phi}(t_{k+1}, \tau)$, and $\mathbf{Q}_k(t_{k+1})^{-1}$). Second, in the case that $\mathbf{L}(t)$ is a time-varying matrix, we must store $\mathbf{Q}_k(t_{k+1})^{-1}, k = 0 \dots K-1$, since the evaluation of $\mathbf{Q}_k(t_{k+1})^{-1}$ requires $\mathbf{L}(s)$ (evaluated at $\hat{\mathbf{x}}(s)$) over the time period $[t_{k-1}, t_k]$. The memory requirements of this alternative are still proportional to the length of the trajectory, but are greatly reduced in contrast to the naive method; the added cost is that any new query requires numerical integration from the nearest time t_k to the query time τ (which is order $O(1)$).

Nonlinear Constant-Velocity Prior Example

Since the main source of acceleration (and probably disturbance) in a standard 2D ground-robot model is body-centric (i.e., the actuated wheels), we extend the linear GP prior example from Section 6.3.1 by investigating the use of an alternate 'constant-velocity' prior, with white noise affecting acceleration in the robot body frame, $\dot{\boldsymbol{\nu}}(t) = \mathbf{w}(t)$. This *nonlinear* prior can be written as,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t)) = \begin{bmatrix} \mathbf{0} & \mathbf{R}_{IB}(\theta(t)) \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{x}(t) + \mathbf{u}(t) + \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix} \mathbf{w}(t), \quad (6.46)$$

where we recall that $\theta(t)$ is a component of $\mathbf{x}(t)$, and

$$\mathbf{R}_{IB}(\theta(t)) = \begin{bmatrix} \cos\theta(t) & -\sin\theta(t) & 0 \\ \sin\theta(t) & \cos\theta(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.47}$$

is a rotation matrix between the inertial and robot body frame. The associated *Markovian* state is then $\mathbf{x}(t) = \begin{bmatrix} \mathbf{p}(t)^T & \boldsymbol{\nu}(t)^T \end{bmatrix}^T$, where $\boldsymbol{\nu}(t) = \begin{bmatrix} v(t) & u(t) & \omega(t) \end{bmatrix}^T = \mathbf{R}_{IB}(\theta(t))^T \dot{\mathbf{p}}(t)$ is the robot-oriented velocity (with longitudinal, lateral, and rotational components).

Linearizing about an arbitrary operating point, $\bar{\mathbf{x}}(t)$, the components $\mathbf{F}(t)$, $\mathbf{v}(t)$ and $\mathbf{L}(t)$ from (6.40) are straightforward to derive. Similar to the linear prior example, described in Section 6.3.1, we define the exogenous input $\mathbf{u}(t) = \mathbf{0}$; however, recalling (6.39), we note that $\mathbf{v}(t) \neq \mathbf{0}$. Since expressions for $\boldsymbol{\Phi}(t, s)$, and by extension $\mathbf{Q}_k(t_{k+1})$, are not obvious, we will rely on numerical integration for their evaluation.

6.3.3 Training the Hyperparameters

As with any GP regression, we have *hyperparameters* associated with our mean and covariance functions. In this problem, the pertinent hyperparameters are $\check{\mathbf{x}}_0$, $\check{\mathbf{P}}_0$ and $\mathbf{Q}_C$. While we will typically have some information about the initial state (and uncertainty) of the robot, the matrix $\mathbf{Q}_C$ can be difficult to determine and affects both the smoothness and length-scale of the class of functions we are considering as motion priors. The standard approach to selecting this parameter is to use a training dataset (with ground-truth), and perform optimization using the log marginal likelihood (log-evidence) or its approximation as the objective function (Rasmussen and Williams, 2006). We begin with the marginal likelihood equation,

$$\log p(\mathbf{y}|\mathbf{Q}_C) = -\frac{1}{2}(\mathbf{y} - \check{\mathbf{x}})^T \mathbf{P}_\sigma^{-1}(\mathbf{y} - \check{\mathbf{x}}) - \frac{1}{2}\log|\mathbf{P}_\sigma| - \frac{n}{2}\log 2\pi, \tag{6.48a}$$

$$\mathbf{P}_\sigma = \check{\mathbf{P}}(\mathbf{Q}_C) + \sigma_w^2 \mathbf{1}, \tag{6.48b}$$

where $\mathbf{y}$ is a stacked vector of state observations (ground-truth measurements) with additive noise $\mathcal{N}(\mathbf{0}, \sigma_w^2 \mathbf{1})$, $\check{\mathbf{x}}$ is a stacked vector of the mean equation evaluated at the observation times, $t_w, w = 1 \ldots W$, and $\check{\mathbf{P}}$ is the covariance matrix associated with $\check{\mathbf{x}}$ and generated using the hyperparameters $\mathbf{Q}_C$. Taking partial derivatives of the marginal likelihood with respect to the hyperparameters, we get

$$\frac{\partial}{\partial \mathbf{Q}_{C_{ij}}} \log p(\mathbf{y}|\mathbf{Q}_C) = \frac{1}{2}(\mathbf{y} - \check{\mathbf{x}})^T \mathbf{P}_\sigma^{-1} \frac{\partial \mathbf{P}_\sigma}{\partial \mathbf{Q}_{C_{ij}}} \mathbf{P}_\sigma^{-1}(\mathbf{y} - \check{\mathbf{x}}) - \frac{1}{2}\operatorname{tr}\left(\mathbf{P}_\sigma^{-1} \frac{\partial \mathbf{P}_\sigma}{\partial \mathbf{Q}_{C_{ij}}}\right), \tag{6.49}$$

where we have used $\frac{\partial \mathbf{P}_\sigma^{-1}}{\partial \mathbf{Q}_{C_{ij}}} = -\mathbf{P}_\sigma^{-1}\frac{\partial \mathbf{P}_\sigma}{\partial \mathbf{Q}_{C_{ij}}}\mathbf{P}_\sigma^{-1}$.

The typical complexity of hyperparameter training is bottlenecked at $O(W^3)$ due to the inversion of $\mathbf{P}_\sigma$ (which is typically dense). Given $\mathbf{P}_\sigma^{-1}$, the complexity is then typically bottlenecked at $O(W^2)$ due to the calculation of $\frac{\partial \mathbf{P}_\sigma}{\partial \mathbf{Q}_{C_{ij}}}$. Fortunately, in the present case, the computation of the log marginal likelihood can also be done efficiently due to the sparseness of the inverse kernel matrix, $\check{\mathbf{P}}^{-1}$. Despite the addition of observation noise, $\sigma_w^2\mathbf{1}$, causing $\mathbf{P}_\sigma^{-1}$ to be a dense matrix, we are still able to take advantage of our sparsity by applying the Sherman-Morrison-Woodbury identity to find that

$$\mathbf{P}_\sigma^{-1} = \check{\mathbf{P}}^{-1} - \check{\mathbf{P}}^{-1}\left(\check{\mathbf{P}}^{-1} + \frac{1}{\sigma_w^2}\mathbf{1}\right)^{-1}\check{\mathbf{P}}^{-1} = \mathbf{F}^{-T}\mathbf{Q}_\sigma^{-1}\mathbf{F}^{-1}, \tag{6.50}$$

where we define

$$\mathbf{Q}_\sigma^{-1} := \left(\mathbf{Q}^{-1} - \mathbf{Q}^{-1}\mathbf{F}^{-1}\left(\check{\mathbf{P}}^{-1} + \frac{1}{\sigma_w^2}\mathbf{1}\right)^{-1}\mathbf{F}^{-T}\mathbf{Q}^{-1}\right). \tag{6.51}$$

Although explicitly computing $\mathbf{P}_\sigma^{-1}$ is of order $O(W^2)$, we note that the product with a vector, $\mathbf{P}_\sigma^{-1}\mathbf{v}$, can be computed in $O(W)$ time; this is easily observable given that $\mathbf{Q}^{-1}$ is block-diagonal, $\mathbf{F}^{-1}$ is lower block-bidiagonal, and the product $(\check{\mathbf{P}}^{-1} + \frac{1}{\sigma_w^2}\mathbf{1})^{-1}\mathbf{v}$ can be computed in $O(W)$ time using Cholesky decomposition (since $\check{\mathbf{P}}^{-1} + \frac{1}{\sigma_w^2}\mathbf{1}$ is block-tridiagonal). While the two terms in (6.49) can be combined into a single trace function, it is simpler to study their computational complexity separately. Starting with the first term, we have

$$\frac{1}{2}(\mathbf{y}-\check{\mathbf{x}})^T\mathbf{P}_\sigma^{-1}\frac{\partial \mathbf{P}_\sigma}{\partial \mathbf{Q}_{C_{ij}}}\mathbf{P}_\sigma^{-1}(\mathbf{y}-\check{\mathbf{x}}) = \frac{1}{2}(\mathbf{y}-\check{\mathbf{x}})^T\mathbf{F}^{-T}\mathbf{Q}_\sigma^{-1}\frac{\partial \mathbf{Q}}{\partial \mathbf{Q}_{C_{ij}}}\mathbf{Q}_\sigma^{-1}\mathbf{F}^{-1}(\mathbf{y}-\check{\mathbf{x}}), \tag{6.52}$$

where

$$\frac{\partial \mathbf{Q}}{\partial \mathbf{Q}_{C_{ij}}} = \operatorname{diag}\left(\mathbf{0}, \frac{\partial \mathbf{Q}_1}{\partial \mathbf{Q}_{C_{ij}}}, \ldots, \frac{\partial \mathbf{Q}_K}{\partial \mathbf{Q}_{C_{ij}}}\right), \tag{6.53}$$

and

$$\frac{\partial \mathbf{Q}_j}{\partial \mathbf{Q}_{C_{ij}}} := \int_{t_{j-1}}^{t_j} \mathbf{\Phi}(t_j, s)\mathbf{L}(s)\frac{\partial \mathbf{Q}_C}{\partial \mathbf{Q}_{C_{ij}}}\mathbf{L}(s)^T\mathbf{\Phi}(t_j, s)^T\, ds, \quad \frac{\partial \mathbf{Q}_C}{\partial \mathbf{Q}_{C_{ij}}} = \mathbf{1}_{i,j}, \tag{6.54}$$

where $\mathbf{1}_{i,j}$ is a projection matrix with a 1 at the i^{th} row and j^{th} column. Taking advantage of the sparse matrices and previously mentioned fast matrix-vector products, it is clear that (6.52) can be computed in $O(W)$ time (in contrast to the typical $O(W^3)$ time).

Examining the second term, we have

$$\frac{1}{2}\operatorname{tr}\left(\mathbf{P}_\sigma^{-1}\frac{\partial \mathbf{P}_\sigma}{\partial \mathbf{Q}_{C_{ij}}}\right) = \frac{1}{2}\operatorname{tr}\left(\left(\mathbf{F}^{-T}\mathbf{Q}_\sigma^{-1}\mathbf{F}^{-1}\right)\left(\mathbf{F}\frac{\partial \mathbf{Q}}{\partial \mathbf{Q}_{C_{ij}}}\mathbf{F}^T\right)\right) = \frac{1}{2}\operatorname{tr}\left(\frac{\partial \mathbf{Q}}{\partial \mathbf{Q}_{C_{ij}}}\mathbf{Q}_\sigma^{-1}\right), \quad (6.55)$$

which can only be computed in $O(W^2)$, due to the form of $\mathbf{Q}_\sigma^{-1}$. In general, the total complexity of training for this sparse class of prior is then bottlenecked at $O(W^2)$. A complexity of $O(W)$ can only be achieved by ignoring the additive measurement noise, $\sigma_w^2\mathbf{1}$; revisiting the second term of (6.49), and setting $\mathbf{P}_\sigma = \check{\mathbf{P}}$, we find that

$$\frac{1}{2}\operatorname{tr}\left(\check{\mathbf{P}}^{-1}\frac{\partial \check{\mathbf{P}}}{\partial \mathbf{Q}_{C_{ij}}}\right) = \frac{1}{2}\operatorname{tr}\left(\frac{\partial \mathbf{Q}}{\partial \mathbf{Q}_{C_{ij}}}\mathbf{Q}^{-1}\right), \quad (6.56)$$

which can be computed in time $O(W)$. The effect of ignoring the measurement noise, $\sigma_w^2\mathbf{1}$, is that the trained hyperparameters will result in an underconfident prior; the degree of this underconfidence depends on the magnitude of the noise we are choosing to ignore. If accurate ground-truth measurements are available and the size of the training dataset is very large, this approximation may be beneficial.

6.3.4 Mobile Robot Experiment

To validate our nonlinear extension against the linear method of Barfoot et al. (2014) and the dense method of Tong et al. (2013), we utilize the same 2D mobile robot dataset (depicted in Figure 6.3). The experiment used a P3-AT mobile robot, equipped with a Hokuyo URG-04LX-UG01 laser rangefinder, driving through a 'forest' of 17 plastic-tube landmarks. The dataset includes odometry measurements, range/bearing measurements (to the plastic-tubes), and ground-truth for both the robot trajectory and landmark positions (provided by a Vicon motion capture system).

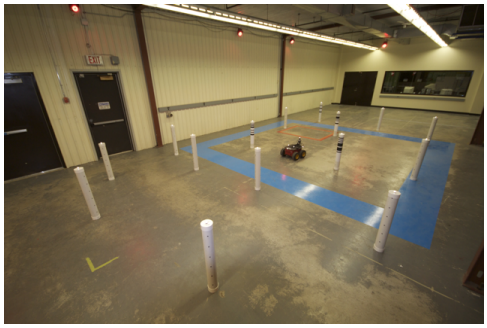


Figure 6.3: Photograph of the experiment conducted by Tong et al. (2013).

Four estimators are implemented for comparison: (i) the algorithm described by Tong et al. (2013), *GP-Pose-Dense*, which has a dense inverse kernel matrix, (ii) a naive version of *Markovian* STEAM estimator, *GP-Traj-Dense*, which is based on the LTI SDE prior example described in Section 6.3.1, but does not exploit sparsity, (iii) a sparsity-exploiting version of the STEAM estimator, *GP-Traj-Sparse-LTI*, which uses the same LTI SDE prior, and (iv) a sparsity-exploiting STEAM estimator, *GP-Traj-Sparse-NTV*, that uses the NTV SDE

Table 6.1: This table contains results from the k-folds hyperparameter training for both the LTI and NTV priors described in Sections 6.3.1 and 6.3.2. Specifically, it contains the mean and standard deviation of the three hyperparameters (diagonal of $\mathbf{Q}_C$) and average training time for a set of hyperparameters

	$\mathbf{Q}_{C,11}$ $[\mu \pm \sigma]$	$\mathbf{Q}_{C,22}$ $[\mu \pm \sigma]$	$\mathbf{Q}_{C,33}$ $[\mu \pm \sigma]$	Average Time [s]
LTI	1.679e-2 ± 9.077e-4	7.841e-3 ± 6.863e-4	5.930e-2 ± 1.817e-3	22.56
NTV	2.811e-2 ± 6.022e-4	2.244e-4 ± 1.923e-5	5.993e-2 ± 1.710e-3	65.16

prior example described in Sections 6.3.2. Note that we do not implement nonlinear versions of the first two estimators, *GP-Pose-Dense* and *GP-Traj-Dense*, because they are only of interest with regard to computational performance.

Measurement Models

We have access to two types of measurements: range/bearing to landmarks (using a laser rangefinder) and wheel odometry (in the form of robot-oriented velocity). For both state parameterizations, the range/bearing measurement model takes the form

$$\mathbf{y}_{kj} = \mathbf{g}_{\text{rb}}(\mathbf{x}(t_k), \boldsymbol{\ell}_j) + \mathbf{n}_{kj} = \begin{bmatrix} \sqrt{(\ell_{j,1} - x(t_k))^2 + (\ell_{j,2} - y(t_k))^2} \\ \text{atan2}(\ell_{j,2} - y(t_k), \ell_{j,1} - x(t_k)) \end{bmatrix} + \mathbf{n}_{kj}. \tag{6.57}$$

The wheel odometry measurement model gives the longitudinal and rotational speeds of the robot, taking the form

$$\mathbf{y}_k = \mathbf{g}_{\text{wo}}(\mathbf{x}(t_k)) + \mathbf{n}_k = \begin{bmatrix} \cos\theta(t_k) & \sin\theta(t_k) & 0 \\ 0 & 0 & 1 \end{bmatrix} \dot{\mathbf{p}}(t_k) + \mathbf{n}_k \tag{6.58}$$

for the LTI SDE prior, and

$$\mathbf{y}_k = \mathbf{g}_{\text{wo}}(\mathbf{x}(t_k)) + \mathbf{n}_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{\nu}(t_k) + \mathbf{n}_k \tag{6.59}$$

for the NTV SDE prior. Note that in both cases the velocity information is extracted easily from the state since we are estimating it directly.

Hyperparameters

For this experiment, we obtained $\mathbf{Q}_C$ for both the LTI and NTV priors by constraining it to be a diagonal matrix and taking the data-driven training approach using log marginal likelihood (with ground-truth measurements) described in Section 6.3.3. Specifically, five

Table 6.2: This table contains estimation error results from both the k-folds cross-validation and roughly 15-minute test set for both the LTI and NTV priors.

	Average K-fold Validation Error			Test Set Error		
	Trans. Err. [m]	Angular Err. [deg]	Landmark Err. [m]	Trans. Err. [m]	Angular Err. [deg]	Landmark Err. [m]
LTI	7.493e-2	3.282	0.100	7.419e-2	2.274	8.246e-2
NTV	8.002e-2	3.626	0.111	7.301e-2	2.284	9.059e-2

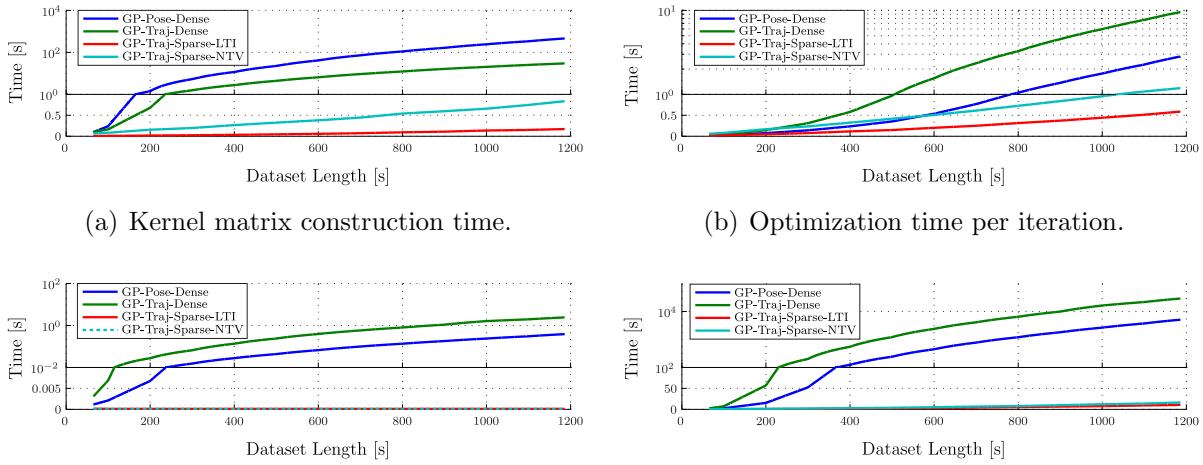
minutes of the roughly 20-minute dataset were set aside for training; the five minutes of training data are then split into ten 30-second trajectories (each having 300 ground-truth measurements) that are used for k-folds training and cross-validation. The remaining 15 minutes of data are used as a test set to perform a final evaluation of the estimator.

Results from the training, validation, and testing can be found in Tables 6.1 and 6.2. Notably, each of the ten folds of hyperparameter training produced a very similar set of values for our model of $\mathbf{Q}_C$. Since the performance metric we are most interested in is the accuracy of the estimator, the k-fold cross-validations and final test were performed by solving the proposed GP regression problem with the given sets of hyperparameters and then comparing the trajectory estimates to ground-truth. The average validation errors over the ten folds are fairly good for both the LTI and NTV priors, but more importantly were very representative of the errors on the final test set. An interesting advantage of the NTV prior is that it is rotationally invariant; note that the values of $\mathbf{Q}_{C,11}$ and $\mathbf{Q}_{C,22}$ for the LTI prior (related to x and y motions, respectively) are distinctly different (a product of the training data).

Given that our hyperparameters have a physical interpretation (in this case, related to our prior knowledge of vehicle accelerations), we note that the goal of training should be to find a $\mathbf{Q}_C$ that generalizes well for a specific platform (it cannot be assumed that a single choice of $\mathbf{Q}_C$ will generalize well for different types of vehicles). With regard to the sensitivity of the hyperparameters, it is our experience that when relatively high-frequency measurements are available, the estimator is not overly sensitive to a more inclusive (slightly underconfident) prior; rather, it is more 'dangerous' to choose a restrictive (overconfident) prior that may preclude the true underlying motion.

Computational Cost

In order to quantify the computational gain associated with using an exactly-sparse GP prior, we time the four batch estimators for varying lengths of the dataset (notably,



(a) Kernel matrix construction time.

(b) Optimization time per iteration.

(c) Interpolation time per additional query time.

(d) Total computation time.

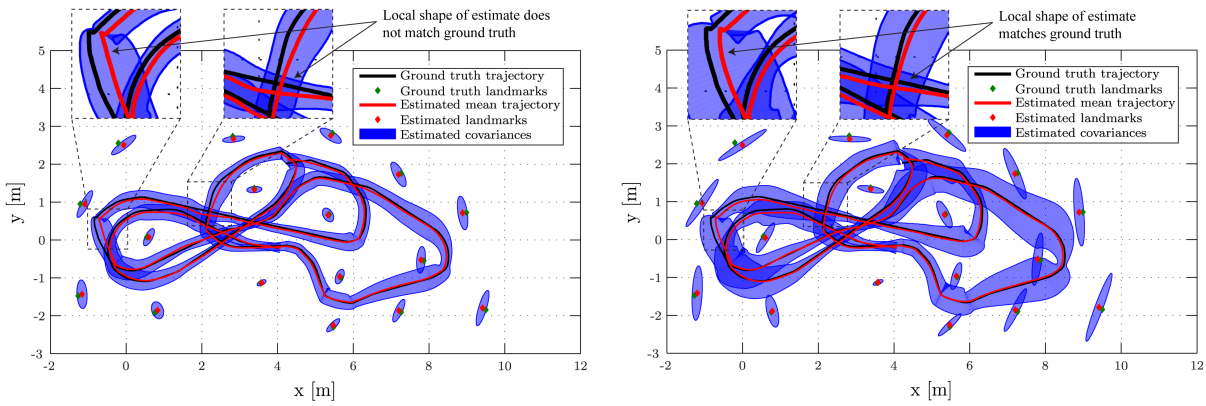
Figure 6.4: This figure presents the computational costs associated with the four batch STEAM estimators. Notably, the *sparse* algorithms exhibit either constant or linear cost (in trajectory length) for all of the timed components (including the full solution). The estimators were timed in Matlab with a 2.4GHz i7 processor and 8GB of 1333MHz DDR3 RAM. Note the change from a linear to a log scale in the upper part of each plot.

we use the full 20 minutes of data for timing purposes). The timing results, presented in Figure 6.4, are broken down into several pieces. Note that the *GP-Pose-Dense* and *GP-Traj-Dense* estimators are significantly slower in all aspects since they use/assume a dense kernel matrix. For the exactly sparse estimators, *GP-Traj-Sparse-LTI* and *GP-Traj-Sparse-NTV*, we verify that the interpolation time is constant for a single query, and that the kernel construction time, iteration time, and total time are all linear in the number of measurements; these approaches significantly outperform the other algorithms.

The additional cost of the *GP-Traj-Sparse-NTV* algorithm over the *GP-Traj-Sparse-LTI* algorithm in kernel construction time is due to the linearization and numerical integration of the prior mean and covariance. The optimization time of the *GP-Traj-Sparse-NTV* algorithm is also affected because the kernel matrix must be reconstructed from a new linearization of the prior during every optimization iteration. Finally, the *GP-Traj-Sparse-NTV* algorithm also incurs some numerical integration cost in interpolation time, but it is constant and very small.

Increasing Nonlinearity

In a problem with fairly accurate and high-rate measurements, we note that both the *GP-Traj-Sparse-LTI* and *GP-Traj-Sparse-NTV* estimators provide similar accuracy; a



(a) *GP-Traj-Sparse-LTI* trajectory sample.

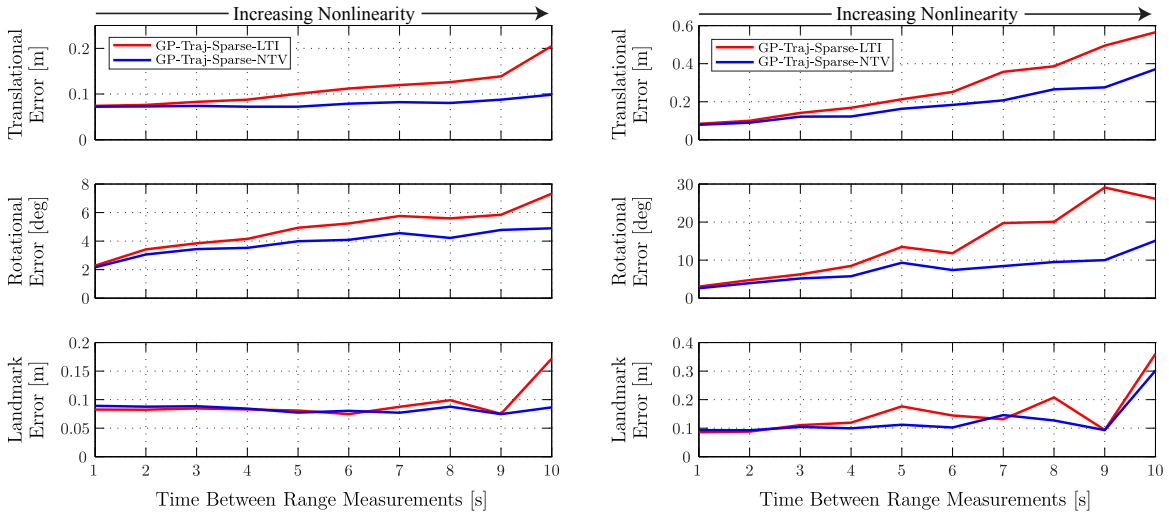
(b) *GP-Traj-Sparse-NTV* trajectory sample.

Figure 6.5: The smooth trajectories and 3σ covariance envelope estimates produced by the *GP-Traj-Sparse* estimators (both linear and nonlinear) over a short segment of the dataset.

qualitative result is depicted in Figure 6.5, where both the posterior mean and covariance of the continuous-time trajectory are queried and plotted at a rate of 10 Hz. Notably, the *GP-Traj-Sparse-NTV* algorithm differed slightly from the others (the dense estimator result is not shown, but was remarkably similar to that of the sparse linear prior); qualitatively, we found that the trajectory estimated by the *GP-Traj-Sparse-NTV* algorithm more accurately matched the local shape of the ground-truth on many occasions (note the insets in Figure 6.5). Also, it is clear from the plotted 3σ covariance envelope that the estimate from the *GP-Traj-Sparse-NTV* algorithm tends to be more uncertain.

In order to expose the benefit of a nonlinear prior based on the expected motion of the vehicle, we increase the nonlinearity of the problem by reducing measurement frequency. The result of varying range measurement frequency, with and without the use of odometry measurements, is shown in Figure 6.6. In general, it is clear that as the interval between range measurements is increased, the *GP-Traj-Sparse-NTV* estimator is able to produce a more accurate estimate of the continuous-time pose (translation and rotation) than the *GP-Traj-Sparse-LTI* estimator. In the case that the 1 Hz odometry measurements are available, as seen in Figure 6.6(a), the difference in the rotation estimates is small, because the *GP-Traj-Sparse-LTI* estimator has a reasonable amount of information about its heading; however, in the case that the odometry measurements are unavailable, as seen in Figure 6.6(b), the advantage of the nonlinear prior, provided by the *GP-Traj-Sparse-NTV* estimator, becomes more prominent.

In order to gain some qualitative intuition about how the continuous-time estimates are affected by the reduction of measurement frequency, Figure 6.7 shows the trajectories



(a) RMS errors using odometry measurements.

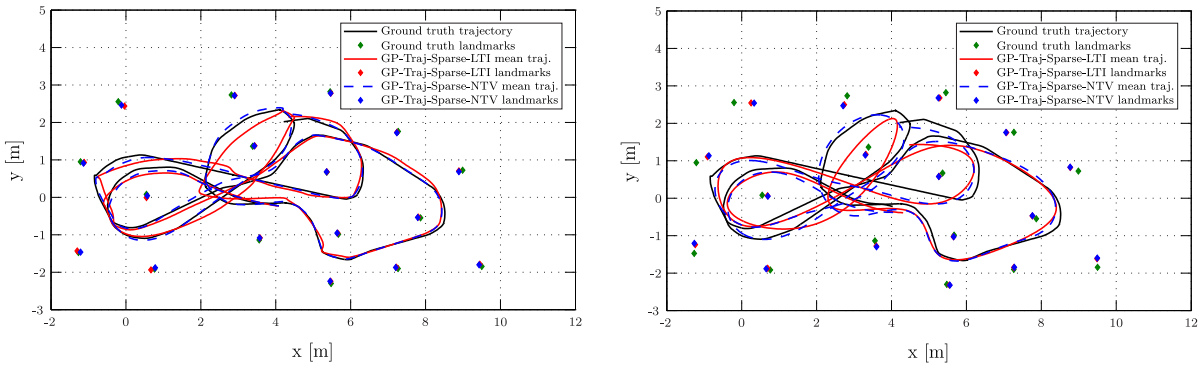
(b) RMS errors without odometry measurements.

Figure 6.6: Plots comparing use of the *GP-Traj-Sparse-LTI* and *-NTV* algorithms for an increasingly nonlinear problem; the dataset was made more nonlinear by varying the interval between available range measurements. The results in (a) used the odometry measurements available at 1 Hz, while (b) was made to be even more nonlinear by excluding them. The plots show that for a small interval between range measurements, both estimators perform similarly; however, as the interval was increased the estimate provided by the nonlinear prior is consistently better in both translation and angular error.

for the same small subsection presented in Figure 6.5, where an interval between range measurements of 7 seconds was used. In both plots, it is clear that the *GP-Traj-Sparse-NTV* estimator matches the ground-truth more closely, as previously indicated by the error plots in Figure 6.6.

6.4 Estimating Trajectories in $SE(3)$

Using what we have learned from our GP-based STEAM derivations (where thus far $\mathbf{x}(t) \in \mathbb{R}^N$), we now wish to formulate a fast and effective GP prior for bodies rotating and translation in three dimensions (i.e., $\mathbf{x}(t) \in SE(3)$). Notably, previous GP approaches have described a method to work with trajectories in 3D (Tong et al., 2014) and how to exploit the Markov property in a vector space context (Barfoot et al., 2014), but not at the same time. This section will provide: (i) a review of the 3D adaptation proposed by Tong et al. (2014), (ii) analysis on how our method of handling NTV SDEs (from Section 6.3.2) could be applied to $SE(3)$, (iii) a practical GP prior for $SE(3)$ that is made



(a) With odometry measurements.

(b) Without odometry measurements.

Figure 6.7: Plots showing the *GP-Traj-Sparse-LTI* and *-NTV* estimates for the same small trajectory subsection as Figure 6.5, with an interval between range measurements of 7 seconds. Results in (a) used odometry measurements, while (b) did not.

efficient by approximating nonlinear kinematics in a piecewise fashion, (iv) a description of methods that allow us to process high-rate data and loop-closures in real time, and (v) experimental validation using the appearance-based lidar dataset.

6.4.1 Background – Adapting for $SE(3)$

We begin by discussing the GP prior formulation and assumptions used by Tong et al. (2014) to formulate a 3D adaptation of their GPGN algorithm. Recalling the problem formulation described in Section 6.2.1, we now shift the GP regression problem to consider state variables that belong to the matrix Lie group $SE(3)$. In continuous-time, Tong et al. (2014) define the trajectory state as

$$\mathbf{x}(t) := \mathbf{T}(t) = \mathbf{T}_{t,i}, \quad \underrightarrow{\mathcal{F}}_t := \underrightarrow{\mathcal{F}}_{\text{rob}}(t), \tag{6.60}$$

where $\underrightarrow{\mathcal{F}}_{\text{rob}}(t)$ is the time-varying robot frame and $\mathbf{T}(t)$ is the transformation that describes the robot pose (at time t) with respect to an inertial frame, $\underrightarrow{\mathcal{F}}_i$. Following our standard approach to GP-based STEAM, we discretize the trajectory at the measurement times; notably, this process mirrors the standard, three-dimensional, discrete-time SLAM formulation, discussed in Section 3.2.3, where we have the discrete state collection,

$$\mathbf{x} = \{\mathbf{T}_{0,i}, \mathbf{T}_{1,i}, \ldots, \mathbf{T}_{k,i}, \ldots, \mathbf{T}_{K,i}\}, \quad \underrightarrow{\mathcal{F}}_k := \underrightarrow{\mathcal{F}}_{\text{rob}}(t_k), \tag{6.61}$$

and the associated perturbation vector

$$\delta\mathbf{x} := \begin{bmatrix} \delta\boldsymbol{\xi}_{0,i}^T & \ldots & \delta\boldsymbol{\xi}_{k,i}^T & \ldots & \delta\boldsymbol{\xi}_{K,i}^T \end{bmatrix}^T, \tag{6.62}$$

where the elements of $\delta\mathbf{x}$ can be used to update the states in $\mathbf{x}$ with the constraint-sensitive scheme, $\bar{\mathbf{T}}_{k,i} \leftarrow \exp(\delta\boldsymbol{\xi}_{k,i}^{\wedge})\bar{\mathbf{T}}_{k,i}$. It is straightforward to see how the 'observation portion' of the 3D estimation problem can be handled in the same way as the discrete-time solution: recalling the discrete-measurement cost terms, from (3.58),

$$\mathbf{e}_{m_{kj}}(\mathbf{z}) = \mathbf{y}_{kj} - \mathbf{g}_{kj}(\mathbf{z}), \quad \mathbf{g}_{kj}(\mathbf{z}) := \mathbf{g}(\mathbf{x}_k, \boldsymbol{\ell}_j) = \mathbf{k}(\mathbf{T}_{k,i}\, \boldsymbol{p}_i^{\ell_j,i}),$$

where $\mathbf{z} = \{\mathbf{x}, \boldsymbol{\ell}\}$, $\mathbf{T}_{k,i}$ transforms the homogeneous landmark point, $\boldsymbol{p}_i^{\ell_j,i}$, into the sensor frame at time t_k, and $\mathbf{k}(\cdot)$ is a nonlinear sensor model. However, it is not obvious how a GP (which describes a continuous distribution over a *vectorspace*) can be used to represent a trajectory prior that belongs to $SE(3)$. In the formulation proposed by Tong et al. (2014), the 'constant-velocity' prior from (6.5) is (naively) applied to the vector space $\boldsymbol{\xi}(t) = \ln(\mathbf{T}(t))^{\vee} \in \mathbb{R}^6$, such that

$$\ddot{\boldsymbol{\xi}}(t) = \mathbf{w}(t), \quad \mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_C\delta(t-t')), \quad \mathbf{u}(t) = \mathbf{0}, \tag{6.63}$$

which results in the GP prior,

$$\boldsymbol{\xi}(t) \;\sim\; \mathcal{GP}\bigg(\check{\boldsymbol{\xi}}(t_0), \frac{\min(t,t')^2\max(t,t')}{2} - \frac{\min(t,t')^3}{6}\bigg). \tag{6.64}$$

Tong et al. (2014) then employ a dual-state formulation, such that when a $SE(3)$ term is required, we have $\mathbf{x}(t) = \mathbf{T}(t) = \exp(\boldsymbol{\xi}(t)^{\wedge})$, and when a vector-space term is required, we have $\mathbf{x}(t) = \boldsymbol{\xi}(t) = \ln(\mathbf{T}(t))^{\vee}$. There are two downfalls to this approach: (i) estimating only the pose variables, $\mathbf{T}(t)$, causes the 'constant-velocity' GP prior to become dense over the discrete state, $\mathbf{x}$, and (ii) the 3×1 angular component of $\boldsymbol{\xi}(t)$ violates the assumed vector space by exhibiting discontinuities over large rotations. From our experience using *Markov* trajectories, we know that the former issue can be rectified by re-introducing the velocity variables, $\dot{\boldsymbol{\xi}}(t)$; however, we will expand upon the latter issue to ensure that we avoid it in our future GP prior proposals.

Specifically, we note that the discontinuity issue presents itself through the term $\check{\mathbf{P}}^{-1}(\bar{\mathbf{x}} - \check{\mathbf{x}})$, which is found in both the Gauss-Newton update, (6.13), and interpolation equation, (6.17a). In essence, the assumption made by the GP is that the elements, $\ln(\bar{\mathbf{T}}_{k,i})^{\vee} - \ln(\check{\mathbf{T}}_{k,i})^{\vee}$, of the column vector difference, $\bar{\mathbf{x}} - \check{\mathbf{x}}$, will change smoothly across the temporally sequential discretization, $t_0 < t_1 < \ldots < t_K$. The repercussion of violating this assumption is most easily viewed in the *exactly sparse* version of the problem, where we can write the cost of the prior in the 'factor' form:

$$J_p = \frac{1}{2}(\mathbf{x} - \check{\mathbf{x}})^T\check{\mathbf{P}}^{-1}(\mathbf{x} - \check{\mathbf{x}}) = \frac{1}{2}(\mathbf{x}_0 - \check{\mathbf{x}}_0)^T\check{\mathbf{P}}_0^{-1}(\mathbf{x}_0 - \check{\mathbf{x}}_0) + \frac{1}{2}\sum_k \mathbf{e}_{u_k}^T\mathbf{Q}_k(t_{k+1})^{-1}\mathbf{e}_{u_k} \tag{6.65}$$

where the errors terms, $\mathbf{e}_{u_k}$, are *exactly* (Barfoot et al., 2014)

$$\mathbf{e}_{u_k} = (\mathbf{x}_{k+1} - \check{\mathbf{x}}_{k+1}) - \boldsymbol{\Phi}(t_{k+1}, t_k)(\mathbf{x}_k - \check{\mathbf{x}}_k). \tag{6.66}$$

Notably, use of the $\ln(\cdot)^\vee$ operator will cause a large discontinuity to occur between $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ when the two rotation parameters straddle the $-\pi/\pi$ radian threshold; the GP prior will view this discontinuity as a large angular 'error' and will inflict a strong penalization that is not warranted. In effect, priors that exhibit this issue are only useful for *short* or *specific* trajectories that do not exhibit large rotations. Finally, we note that Tong et al. (2014) make the (ad hoc) assumption that vector space differences, $\ln(\bar{\mathbf{T}})^\vee - \ln(\check{\mathbf{T}})^\vee$, can be replaced with the operation $\ln(\bar{\mathbf{T}}\check{\mathbf{T}}^{-1})^\vee$; although their proposed prior mean is not time-varying (i.e., $\check{\mathbf{T}}(t) = \check{\mathbf{T}}(t_0)$), it is noted that this assumption would help to avoid discontinuities if the prior quantities, $\check{\mathbf{T}}_{k,i}$, closely modelled the estimates $\bar{\mathbf{T}}_{k,i}$.

6.4.2 Nonlinear GP Regression for $SE(3)$

Moving forward, we now wish to exploit the *exactly sparse* GP priors described in Section 6.3 for trajectories that belong to $SE(3)$. Recalling the nonlinear kinematics discussed in Chapter 5, we are motivated to use the *nonlinear* SDE

$$\dot{\mathbf{T}}(t) = \boldsymbol{\varpi}(t)^\wedge \mathbf{T}(t), \tag{6.67a}$$

$$\dot{\boldsymbol{\varpi}}(t) = \mathbf{w}'(t), \qquad \mathbf{w}'(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}'_C \delta(t - t')), \tag{6.67b}$$

where $\boldsymbol{\varpi}(t) = \begin{bmatrix} \boldsymbol{\nu}(t) \\ \boldsymbol{\omega}(t) \end{bmatrix} \in \mathbb{R}^6$ is the body-centric velocity (with linear and angular components, $\boldsymbol{\nu}(t)$ and $\boldsymbol{\omega}(t)$), and the zero-mean, white-noise Gaussian process, $\mathbf{w}'(t)$, is injected directly onto the body-centric acceleration, $\dot{\boldsymbol{\varpi}}(t)$. It follows that the *Markovian* trajectory state we wish to estimate is

$$\mathbf{x}(t) = \{\mathbf{T}(t), \boldsymbol{\varpi}(t)\} \in SE(3) \times \mathbb{R}^6. \tag{6.68}$$

Recalling the (novel) GP linearization methodology, presented in Section 6.3.2, we now show how it can be extended to $SE(3)$ by using the familiar constraint-sensitive perturbations

$$\mathbf{T}(t) = \exp(\delta\boldsymbol{\xi}(t)^\wedge)\bar{\mathbf{T}}(t), \qquad \boldsymbol{\varpi}(t) = \bar{\boldsymbol{\varpi}}(t) + \delta\boldsymbol{\varpi}(t). \tag{6.69}$$

Recalling the perturbed kinematics derived in Chapter 5, (5.23), we apply a similar methodology to the SDE in (6.67) to find the nominal and perturbed components:

$$\dot{\bar{\mathbf{T}}}(t) = \bar{\boldsymbol{\varpi}}(t)^{\wedge}\bar{\mathbf{T}}(t), \tag{6.70a}$$

$$\underbrace{\begin{bmatrix}\delta\dot{\boldsymbol{\xi}}(t)\\ \delta\dot{\boldsymbol{\varpi}}(t)\end{bmatrix}}_{\delta\dot{\mathbf{x}}(t)} \approx \underbrace{\begin{bmatrix}\bar{\boldsymbol{\varpi}}(t)^{\curlywedge} & \mathbf{1}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}}_{\mathbf{F}(t)}\underbrace{\begin{bmatrix}\delta\boldsymbol{\xi}(t)\\ \delta\boldsymbol{\varpi}(t)\end{bmatrix}}_{\delta\mathbf{x}(t)} + \underbrace{\begin{bmatrix}\mathbf{0}\\ -\dot{\bar{\boldsymbol{\varpi}}}(t)\end{bmatrix}}_{\mathbf{v}(t)} + \underbrace{\begin{bmatrix}\mathbf{0}\\ \mathbf{1}\end{bmatrix}}_{\mathbf{L}(t)}\mathbf{w}'(t). \tag{6.70b}$$

Following the standard GP approach, we discretize the trajectory state, $\mathbf{x}(t)$, at the measurement times, t_k, to form the discrete state collection,

$$\mathbf{x} = \{\mathbf{T}_{0,i}, \boldsymbol{\varpi}_0, \ldots, \mathbf{T}_{k,i}, \boldsymbol{\varpi}_k \ldots, \mathbf{T}_{K,i}, \boldsymbol{\varpi}_K\}, \quad \underrightarrow{\mathcal{F}}_k := \underrightarrow{\mathcal{F}}_{\text{rob}}(t_k), \tag{6.71}$$

with the associated perturbation vector

$$\delta\mathbf{x} := \left[\delta\boldsymbol{\xi}(t_0)^T\, \delta\boldsymbol{\varpi}(t_0)^T\, \ldots\, \delta\boldsymbol{\xi}(t_k)^T\, \delta\boldsymbol{\varpi}(t_k)^T\, \ldots\, \delta\boldsymbol{\xi}(t_K)^T\, \delta\boldsymbol{\varpi}(t_K)^T\right]^T. \tag{6.72}$$

In order to utilize the nonlinear SDE (which includes non-vectorspace state variables) in the GP-regression-based localization problem, (6.13), we first note that the vector difference, $\check{\mathbf{x}} - \bar{\mathbf{x}}$, can be replaced with expected value of the perturbation, $E[\delta\mathbf{x}]$. In essence, $E[\delta\mathbf{x}]$ can be thought of as the difference between $\check{\mathbf{x}}$ and $\bar{\mathbf{x}}$ (e.g., over the $SE(3)$ manifold). This gives us the equivalent Gauss-Newton update equation:

$$\left(\check{\mathbf{P}}^{-1} + \mathbf{G}^T\mathbf{R}^{-1}\mathbf{G}\right)\, \delta\mathbf{x}^{\star} = \check{\mathbf{P}}^{-1}E[\delta\mathbf{x}] + \mathbf{G}^T\mathbf{R}^{-1}(\mathbf{y} - \bar{\mathbf{g}}), \tag{6.73}$$

where we require expressions for the 'expected' perturbation, $E[\delta\mathbf{x}]$, and inverse covariance, $\check{\mathbf{P}}^{-1}$. Recalling the methodology from Section 6.3.2 once more, we note that these terms can be found by numerically integrating the SDE in (6.70b), via (6.21) and (6.22). With some effort, we also determine that the corresponding transition function for (6.70b) is:

$$\boldsymbol{\Phi}(t,\tau) = \begin{bmatrix}\bar{\boldsymbol{\mathcal{T}}}(t)\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1} & \bar{\boldsymbol{\mathcal{T}}}(t)\int_\tau^t \bar{\boldsymbol{\mathcal{T}}}(s)^{-1}ds\\ \mathbf{0} & \mathbf{1}\end{bmatrix}, \tag{6.74}$$

which can be verified by checking the conditions

$$\dot{\boldsymbol{\Phi}}(t,\tau) = \mathbf{F}(t)\boldsymbol{\Phi}(t,\tau), \qquad \boldsymbol{\Phi}(\tau,\tau) = \mathbf{1}. \tag{6.75}$$

The first condition is verified by using the second fundamental theorem of calculus,

$$\begin{aligned}\dot{\mathbf{\Phi}}(t,\tau) &= \begin{bmatrix} \frac{d}{dt}\bar{\boldsymbol{\mathcal{T}}}(t)\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1} & \frac{d}{dt}\bar{\boldsymbol{\mathcal{T}}}(t)\int_\tau^t \bar{\boldsymbol{\mathcal{T}}}(s)^{-1}ds \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \dot{\bar{\boldsymbol{\mathcal{T}}}}(t)\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1} & \dot{\bar{\boldsymbol{\mathcal{T}}}}(t)\int_\tau^t \bar{\boldsymbol{\mathcal{T}}}(s)^{-1}ds + \bar{\boldsymbol{\mathcal{T}}}(t)\bar{\boldsymbol{\mathcal{T}}}(t)^{-1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \bar{\boldsymbol{\varpi}}(t)^\wedge\bar{\boldsymbol{\mathcal{T}}}(t)\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1} & \bar{\boldsymbol{\varpi}}(t)^\wedge\bar{\boldsymbol{\mathcal{T}}}(t)\int_\tau^t \bar{\boldsymbol{\mathcal{T}}}(s)^{-1}ds + \mathbf{1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\ &= \begin{bmatrix} \bar{\boldsymbol{\varpi}}(t)^\wedge & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{\Phi}(t,\tau) = \mathbf{F}(t)\mathbf{\Phi}(t,\tau), \end{aligned} \tag{6.76}$$

and the second by simple substitution

$$\mathbf{\Phi}(\tau,\tau) = \begin{bmatrix} \bar{\boldsymbol{\mathcal{T}}}(\tau)\bar{\boldsymbol{\mathcal{T}}}(\tau)^{-1} & \bar{\boldsymbol{\mathcal{T}}}(\tau)\int_\tau^\tau \bar{\boldsymbol{\mathcal{T}}}(s)^{-1}ds \\ \mathbf{0} & \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}. \tag{6.77}$$

To evaluate all of the numerical integrals (such as the one in $\mathbf{\Phi}(t,\tau)$), we recall that we must maintain a continuous-time operating point, $\bar{\mathbf{x}}(t)$. Furthermore, to construct the terms $\mathbf{Q}_k(t_{k+1})^{-1}$ (which can be used to build $\check{\mathbf{P}}^{-1}$), the substitution of $\mathbf{\Phi}(t,\tau)$ into (6.23b) results in a *double numerical integral* of $\bar{\boldsymbol{\mathcal{T}}}(t)$. Alas, while this 3D extension of the nonlinear technique presented in Section 6.3.2 could be used to formulate an MAP estimator for the dynamically motivated GP prior in (6.67), the amount of numerical integration and bookkeeping required in this scheme is off-putting. Instead, since our goal is to produce a scheme that is both fast and effective, we will investigate the use of a piecewise approximation that allows us to closely model the nonlinear kinematics in (6.67) when the motion between measurement times is small, or of nearly 'constant-velocity'.

6.4.3 A Piecewise, Locally Linear GP Prior for $SE(3)$

Motivated by our investigation of the *global* nonlinear kinematics in (6.67), we note that many of the laborious numerical integrals that arise from linearization can be (greatly) simplified by assuming that the velocity estimate, $\bar{\boldsymbol{\varpi}}(t)$, is piecewise-constant (disjoint at the measurement times). This observation motivates us to more closely inspect the nature of the nonlinear SDE. Specifically, our aim is to break the *global* nonlinear SDE into an *equivalent*, piecewise sequence of *local* SDEs (using the Markov property) and then to find a reasonable approximation that can be used to make the piecewise segments *linear*.

Temporal Discretization of the Nonlinear Kinematics

The formulation of our approximation begins by deriving an *equivalent* expression for the nonlinear kinematics in (6.67), using the Lie algebra space, $\mathfrak{se}(3)$. Using the identity

$$\frac{d}{dt}\exp(\mathbf{A}(t)) = \int_0^1 \exp(\alpha\mathbf{A}(t))\frac{d\mathbf{A}(t)}{dt}\exp((1-\alpha)\mathbf{A}(t))d\alpha, \tag{6.78}$$

and the exponential map, $\mathbf{T}(t) = \exp(\boldsymbol{\xi}(t)^\wedge)$, we rearrange (6.67) to find that

$$\boldsymbol{\varpi}(t) = \Big(\dot{\mathbf{T}}(t)\mathbf{T}(t)^{-1}\Big)^\vee = \left(\int_0^1 \mathbf{T}(t)^\alpha\dot{\boldsymbol{\xi}}(t)^\wedge\mathbf{T}(t)^{-\alpha}d\alpha\right)^\vee = \int_0^1 \boldsymbol{\mathcal{T}}(t)^\alpha d\alpha\,\dot{\boldsymbol{\xi}}(t). \tag{6.79}$$

Recalling the definition of $\boldsymbol{\mathcal{J}}(\cdot)$, the (left) Jacobian of $SE(3)$, in (3.43), it follows that

$$\dot{\boldsymbol{\xi}}(t) = \boldsymbol{\mathcal{J}}(\boldsymbol{\xi}(t))^{-1}\boldsymbol{\varpi}(t), \tag{6.80}$$

which is an equivalent expression for the nonlinear kinematics in (6.67a). Using this new relationship, the nonlinear SDE in (6.67) can be written as

$$\begin{bmatrix}\dot{\boldsymbol{\xi}}(t)\\ \dot{\boldsymbol{\varpi}}(t)\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \boldsymbol{\mathcal{J}}(\boldsymbol{\xi}(t))^{-1}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{\xi}(t)\\ \boldsymbol{\varpi}(t)\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ \mathbf{1}\end{bmatrix}\mathbf{w}'(t). \tag{6.81}$$

Notably, linearizing this new expression for the NTV SDE will result in an equally overwhelming collection of numerical integrals. However, this new form reveals an interesting case (i.e., $\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}(t)) \approx \mathbf{1}$) where the NTV SDE *may* be closely approximated by a LTI SDE. Analyzing the relationship in (6.80), we note two conditions for which $\dot{\boldsymbol{\xi}}(t) \approx \boldsymbol{\varpi}(t)$: (i) when $\boldsymbol{\xi}(t)$ is small, we have that $\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}(t)) \approx \mathbf{1}$, and (ii) when $\boldsymbol{\xi}(t)$ and $\boldsymbol{\varpi}(t)$ are nearly parallel (i.e., $\boldsymbol{\xi}(t)^\curlywedge\boldsymbol{\varpi}(t) \approx \mathbf{0}$) we have that $\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}(t))^{-1}\boldsymbol{\varpi}(t) \approx \boldsymbol{\varpi}(t)$; this second condition is fulfilled when velocity is near constant (i.e., $\boldsymbol{\xi}(t) \approx (t-t_0)\boldsymbol{\varpi}(t)$).

In general, it is not reasonable to assume that $\boldsymbol{\xi}(t)$ will remain small or parallel to $\boldsymbol{\varpi}(t)$ over *long* trajectories. However, we note that the Markov property allows us to discretize the NTV SDE in a piecewise fashion. In the context of our GP regression problem, it is logical for us to perform this discretization at the sequence of measurement times, $t_0 < t_1 < \ldots < t_K$. Injecting a discretization point into $\dot{\mathbf{T}}(t)$, we have that

$$\dot{\mathbf{T}}(t) = \frac{d}{dt}\Big(\mathbf{T}(t)\mathbf{T}(t_k)^{-1}\mathbf{T}(t_k)\Big) = \dot{\mathbf{T}}_{t,t_k}\mathbf{T}(t_k), \quad \underrightarrow{\boldsymbol{\mathcal{F}}}_t := \underrightarrow{\boldsymbol{\mathcal{F}}}_{\text{rob}}(t). \tag{6.82}$$

Equating this to (6.67a) and rearranging, we (unsurprisingly) find that

$$\dot{\mathbf{T}}_{t,t_k} = \boldsymbol{\varpi}(t)^\wedge\mathbf{T}(t)\mathbf{T}(t_k)^{-1} = \boldsymbol{\varpi}(t)^\wedge\mathbf{T}_{t,t_k}. \tag{6.83}$$

Defining the *local* pose variable

$$\boldsymbol{\xi}_k(t) := \ln(\mathbf{T}(t)\mathbf{T}(t_k)^{-1})^\vee, \quad t_k \leq t \leq t_{k+1}, \tag{6.84}$$

we also (unsurprisingly) find that

$$\dot{\boldsymbol{\xi}}_k(t) = \boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_k(t))^{-1}\boldsymbol{\varpi}(t). \tag{6.85}$$

Therefore, by using the *local* pose variable definition, $\boldsymbol{\xi}_k(t)$, the NTV SDE in (6.81) can be broken up into the sequence of *local* NTV SDEs:

$$\begin{bmatrix} \dot{\boldsymbol{\xi}}_k(t) \\ \dot{\boldsymbol{\varpi}}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_k(t))^{-1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_k(t) \\ \boldsymbol{\varpi}(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix} \mathbf{w}'(t), \quad t_k \leq t \leq t_{k+1}. \tag{6.86}$$

Notably, the concatenation of these *local* NTV SDEs results in a *global* NTV SDE that is equivalent to both (6.67) and (6.81). Recalling our analysis of (6.80), we are now able to make the (far more) reasonable assumption that $\dot{\boldsymbol{\xi}}_k(t) \approx \boldsymbol{\varpi}(t)$ (and by extension $\ddot{\boldsymbol{\xi}}_k(t) \approx \dot{\boldsymbol{\varpi}}(t)$) when the *local* motion, $\boldsymbol{\xi}_k(t)$, is small or of nearly constant velocity, over the (short) time interval $t \in [t_k, t_{k+1}]$. However, rather than make approximations directly to (6.86) (i.e., $\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_k(t))^{-1} \approx \mathbf{1}$), we instead choose to construct an *exact* sequence of *local* LTI SDEs (that approximate the *local* NTV SDEs):

$$\overbrace{\begin{bmatrix} \dot{\boldsymbol{\xi}}_k(t) \\ \ddot{\boldsymbol{\xi}}_k(t) \end{bmatrix}}^{\dot{\boldsymbol{\gamma}}_k(t)} = \overbrace{\begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}^{\mathbf{F}(t)} \overbrace{\begin{bmatrix} \boldsymbol{\xi}_k(t) \\ \dot{\boldsymbol{\xi}}_k(t) \end{bmatrix}}^{\boldsymbol{\gamma}_k(t)} + \overbrace{\begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix}}^{\mathbf{L}(t)} \mathbf{w}(t), \quad t_k \leq t \leq t_{k+1}, \tag{6.87}$$

where we assert that $\mathbf{w}'(t) \approx \mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_C\delta(t-t'))$, when velocity is near constant. Notably, by choosing to use an *exact* SDE, we will also have *exact* expressions for both the GP-prior cost terms and the interpolation equations. It follows from the (familiar) form of (6.87) that the general solution to our *local* LTI SDE is

$$\boldsymbol{\gamma}_k(t) = \boldsymbol{\Phi}(t, t_k)\boldsymbol{\gamma}_k(t_k) + \int_{t_k}^{t} \boldsymbol{\Phi}(t, s)\mathbf{L}(t)\mathbf{w}(s)\, ds, \quad t_k \leq t \leq t_{k+1}, \tag{6.88}$$

where the state transition function is simply

$$\boldsymbol{\Phi}(t, t') = \begin{bmatrix} \mathbf{1} & (t-t')\mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}. \tag{6.89}$$

Notably, in the absence of noise (i.e., $\mathbf{w}'(t) = \mathbf{w}(t) = \mathbf{0}$), the integral of the global SDE and concatenated local LTI SDEs will produce *equivalent* constant-velocity trajectories

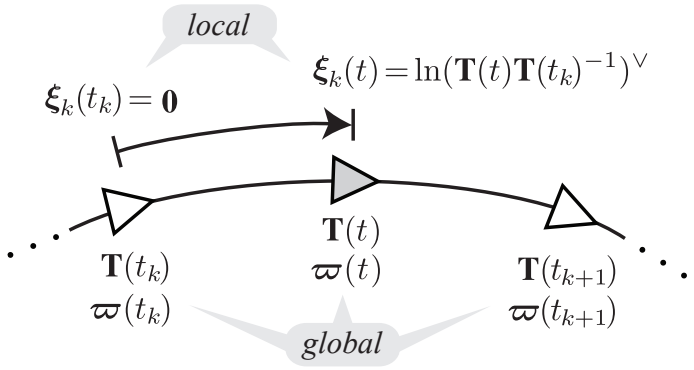


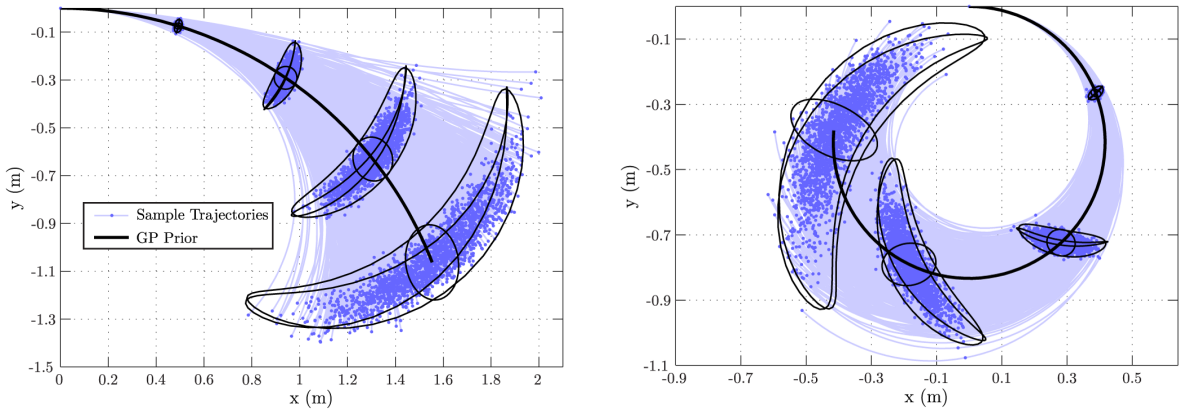
Figure 6.8: This figure depicts the relationships between the *local* pose variable, $\boldsymbol{\xi}_k(t)$, and the *global* trajectory state, $\mathbf{x}(t) = \{\mathbf{T}(t), \boldsymbol{\varpi}(t)\}$. The proposed piecewise GP prior approximates the kinematically motivated SDE in (6.67) by defining a sequence of *local* LTI SDEs, where we assume that the local variable, $\ddot{\boldsymbol{\xi}}_k(t)$, is *noisy* between the measurements times, t_k and t_{k+1}.

(using the same initial conditions). Relating this *local* GP prior definition back to our (desired) *global* state representation, (6.68), we note that it is simple to convert from $\mathbf{x}(t) = \{\mathbf{T}(t), \boldsymbol{\varpi}(t)\}$ to $\boldsymbol{\gamma}_k(t) = \{\boldsymbol{\xi}_k(t), \dot{\boldsymbol{\xi}}_k(t)\}$ by using the equations in (6.84) and (6.85); the relationship between $\boldsymbol{\xi}_k(t)$ and the *global* state variables is depicted in Figure 6.8. Exploiting these relationships (through substitution), it becomes straightforward to define a GP prior from the sequence of local LTI SDEs in (6.87), while estimating the state representation $\mathbf{x}(t) = \{\mathbf{T}(t), \boldsymbol{\varpi}(t)\}$ – all without approximation. Furthermore, we note that the mean and covariance associated with $\mathbf{x}(t)$ can be propagated *continuously* across the measurement times (from one local SDE to the next).

In summary our sequence of local LTI SDEs: (i) accomplish the result of a global SDE in a piecewise fashion, and (ii) approximate the full nonlinear SDE in (6.67) when motion between each pair of measurement times, t_k and t_{k+1}, is either small or of nearly constant velocity. This is verified experimentally by comparing the mean and covariance of our concatenated GP prior to a Monte Carlo simulation of (6.67) (see Figure 6.9).

Analytical Nonlinear Smoothing Factors for SE(3)

In order to linearize our GP prior cost with respect to the *global* state representation, $\mathbf{x}(t)$, we opt to take a factor-graph view of the STEAM problem (recall Figure 6.2). Notably, the objective function cost associated with the *exactly sparse* class of GP priors can be



(a) Low Angular Velocity (b) High Angular Velocity

Figure 6.9: This figure compares the GP prior formed from our concatenated local LTI SDEs (solid black constant-velocity mean and 3σ 'banana' covariances) against a Monte Carlo simulation of the nonlinear SDE from equation (6.67) (the 2000 blue trajectory samples). To make the comparison easier to visualize, noise was injected only in the (body-centric) x-component of translation and z-component of rotation. The banana shapes are 3σ great circles of the six-degree-of-freedom uncertainty hyperellipsoid, mapped onto the xy-plane.

reformulated as (Barfoot et al., 2014):

$$J_p(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\check{\mathbf{x}})^T\check{\mathbf{P}}^{-1}(\mathbf{x}-\check{\mathbf{x}}) = \frac{1}{2}\mathbf{e}_{p_0}^T\check{\mathbf{P}}_0^{-1}\mathbf{e}_{p_0} + \frac{1}{2}\sum_k \mathbf{e}_{u_k}^T\mathbf{Q}_k(t_{k+1})^{-1}\mathbf{e}_{u_k}, \tag{6.90}$$

where $\mathbf{e}_{p_0}^T\check{\mathbf{P}}_0^{-1}\mathbf{e}_{p_0}$ is a unary factor associated with position and velocity at the initial time, t_0, and the (binary) smoothing terms have the *exact* form:

$$\mathbf{e}_{u_k} = \boldsymbol{\gamma}_k(t_{k+1}) - \check{\boldsymbol{\gamma}}_k(t_{k+1}) - \boldsymbol{\Phi}(t_{k+1},t_k)\left(\boldsymbol{\gamma}_k(t_k) - \check{\boldsymbol{\gamma}}_k(t_k)\right), \tag{6.91}$$

with the (familiar) analytical covariance expression

$$\mathbf{Q}_k(\tau) = \begin{bmatrix} \frac{1}{3}\Delta\tau_k^3\mathbf{Q}_C & \frac{1}{2}\Delta\tau_k^2\mathbf{Q}_C \\ \frac{1}{2}\Delta\tau_k^2\mathbf{Q}_C & \Delta\tau_k\mathbf{Q}_C \end{bmatrix}, \quad \mathbf{Q}_k(\tau)^{-1} = \begin{bmatrix} 12\Delta\tau_k^{-3}\mathbf{Q}_C^{-1} & -6\Delta\tau_k^{-2}\mathbf{Q}_C^{-1} \\ -6\Delta\tau_k^{-2}\mathbf{Q}_C^{-1} & 4\Delta\tau_k^{-1}\mathbf{Q}_C^{-1} \end{bmatrix}, \tag{6.92}$$

where $\Delta\tau_k := \tau - t_k$. Without exogenous control inputs, the prior mean is simply

$$\check{\boldsymbol{\gamma}}_k(t) = E[\boldsymbol{\gamma}_k(t)] = \boldsymbol{\Phi}(t,t_k)\check{\boldsymbol{\gamma}}_k(t_k), \tag{6.93}$$

and our expression for $\mathbf{e}_{u_k}$ simplifies to

$$\mathbf{e}_{u_k} = \boldsymbol{\gamma}_k(t_{k+1}) - \boldsymbol{\Phi}(t_{k+1},t_k)\boldsymbol{\gamma}_k(t_k). \tag{6.94}$$

Substituting the expressions from (6.84) and (6.85) into $\boldsymbol{\gamma}_k(t)$,

$$\boldsymbol{\gamma}_k(t) = \begin{bmatrix} \boldsymbol{\xi}_k(t) \\ \dot{\boldsymbol{\xi}}_k(t) \end{bmatrix} = \begin{bmatrix} \ln(\mathbf{T}(t)\mathbf{T}_{k,i}^{-1})^{\vee} \\ \boldsymbol{\mathcal{J}}(\ln(\mathbf{T}(t)\mathbf{T}_{k,i}^{-1})^{\vee})^{-1}\,\boldsymbol{\varpi}(t) \end{bmatrix}, \tag{6.95}$$

it follows that our smoothing terms have the *exact* nonlinear form:

$$\mathbf{e}_{u_k} = \begin{bmatrix} \ln(\mathbf{T}_{k+1,i}\mathbf{T}_{k,i}^{-1})^{\vee} - (t_{k+1} - t_k)\,\boldsymbol{\varpi}_k \\ \boldsymbol{\mathcal{J}}(\ln(\mathbf{T}_{k+1,i}\mathbf{T}_{k,i}^{-1})^{\vee})^{-1}\,\boldsymbol{\varpi}_{k+1} - \boldsymbol{\varpi}_k \end{bmatrix}, \tag{6.96}$$

where $\mathbf{T}_{k,i}$, $\boldsymbol{\varpi}_k$, $\mathbf{T}_{k+1,i}$, and $\boldsymbol{\varpi}_{k+1}$ are temporally adjacent elements of the discretized *global* state vector, $\mathbf{x}$, defined in (6.71). Inspecting (6.96), it is pleasing that the result is intuitive: beginning with a prior akin to white-noise-on-acceleration (i.e., $\ddot{\boldsymbol{\xi}}_k(t) = \mathbf{w}(t)$), the global state variables are penalized for deviating from a body-centric, constant-velocity trajectory. In the case that $\boldsymbol{\varpi}_k = \boldsymbol{\varpi}_{k+1}$, and $\mathbf{T}_{k+1,i} = \exp((t_{k+1} - t_k)\boldsymbol{\varpi}_k^{\wedge})\,\mathbf{T}_{k,i}$ (which agrees with the nonlinear kinematics in (6.67)), we have that $\mathbf{e}_{u_k} = \mathbf{0}$. Furthermore, these *local* errors are unaffected by discontinuities (assuming that the rotation between consecutive measurement times is less than π). At this point, we *may* wish to make the simplifying assumption that $\boldsymbol{\mathcal{J}}(\ln(\mathbf{T}_{k+1,i}\mathbf{T}_{k,i}^{-1})^{\vee})^{-1}\,\boldsymbol{\varpi}_{k+1} \approx \boldsymbol{\varpi}_{k+1}$; however, for completeness, this thesis will continue to use the *exact* form above.

Concerning the initial factor, $\mathbf{e}_{p_0}^T \check{\mathbf{P}}_0^{-1} \mathbf{e}_{p_0}$, we note that in the STEAM context we will typically *lock* the initial pose (but not the velocity). Defining $\mathbf{e}_{p_0} := \boldsymbol{\varpi}_0 - \check{\boldsymbol{\varpi}}_0$, the total set of *hyperparameters* in our approach is: $\mathbf{Q}_C$, $\check{\boldsymbol{\varpi}}_0$, and $\check{\mathbf{P}}_0$. In practice, when little is known about the *a priori* initial velocity, $\check{\boldsymbol{\varpi}}_0$, it is often safe to over-inflate the covariance matrix $\check{\mathbf{P}}_0$; this allows the initial posterior estimate, $\hat{\boldsymbol{\varpi}}_0$, to be determined primarily through the binary smoothing terms.

Efficient GP-Based STEAM for $SE(3)$

In this section, we demonstrate how our sparse set of nonlinear prior factors can be incorporated into the STEAM problem. We begin by recalling the optimization problem we wish to solve

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}}\ J(\mathbf{z}), \quad J(\mathbf{z}) = J_m(\mathbf{z}) + J_p(\mathbf{x}), \tag{6.97}$$

where $\mathbf{z} = \{\mathbf{x}, \boldsymbol{\ell}\}$, $J(\mathbf{z})$ is our objective function, $J_m(\mathbf{z})$ contains the costs associated with the measurements/observations, and $J_p(\mathbf{x})$ is the cost of our GP prior. The form of our GP prior cost, $J_p(\mathbf{x})$, was previously shown in (6.90), where the *exact* nonlinear cost terms,

$\mathbf{e}_{u_k}$, are defined in (6.91). Since the trajectory state, $\mathbf{x}(t)$, has been discretized at the measurement times, it is straightforward to see that $J_m(\mathbf{z})$ has the familiar form

$$J_m(\mathbf{z}) := \frac{1}{2}\sum_{kj} \mathbf{e}_{m_{kj}}^T \mathbf{R}_{kj}^{-1} \mathbf{e}_{m_{kj}}, \tag{6.98}$$

where

$$\mathbf{e}_{m_{kj}}(\mathbf{z}) = \mathbf{y}_{kj} - \mathbf{g}_{kj}(\mathbf{z}), \quad \mathbf{g}_{kj}(\mathbf{z}) := \mathbf{k}(\mathbf{h}_{kj}(\mathbf{z})), \quad \mathbf{h}_{kj}(\mathbf{z}) = \mathbf{T}_{k,i}\, \boldsymbol{p}_i^{\ell_j,i}, \tag{6.99}$$

and $\mathbf{R}_{kj}$ is the covariance of the measurement noise.

In order to leverage our standard Gauss-Newton machinery, described in Section 3.1.2, we linearize our cost terms using the constraint-sensitive perturbations:

$$\hat{\mathbf{T}}_{k,i} = \exp((\delta\boldsymbol{\xi}_{k,i})^\wedge)\bar{\mathbf{T}}_{k,i}, \quad \hat{\boldsymbol{\varpi}}_k = \bar{\boldsymbol{\varpi}}_k + \delta\boldsymbol{\varpi}_k, \quad \hat{\boldsymbol{\ell}}_j = \bar{\boldsymbol{\ell}}_j + \delta\boldsymbol{\ell}_j. \tag{6.100}$$

It follows that the standard linearized measurement costs are simply

$$\mathbf{e}_{m_{kj}}(\bar{\mathbf{z}} + \delta\mathbf{z}) = \mathbf{y}_{kj} - \mathbf{k}(\bar{\mathbf{T}}_{k,i}\, \bar{\boldsymbol{p}}_i^{\ell_j,i}) - \mathbf{G}_{kj}\delta\mathbf{z}, \tag{6.101}$$

$$\mathbf{G}_{kj} := \begin{bmatrix} \mathbf{K}_{kj}\mathbf{H}_{\mathbf{x},kj}\mathbf{P}_{\mathbf{x},k} & \mathbf{K}_{kj}\mathbf{H}_{\boldsymbol{\ell},kj}\mathbf{P}_{\boldsymbol{\ell},j} \end{bmatrix}, \tag{6.102}$$

where $\mathbf{K}_{kj}$, $\mathbf{H}_{\mathbf{x},kj}$, and $\mathbf{H}_{\boldsymbol{\ell},kj}$ were defined in (3.62) and (3.64), and $\mathbf{P}_{\mathbf{x},k}$ and $\mathbf{P}_{\boldsymbol{\ell},j}$ are projection matrices that select the k^{th} element of $\mathbf{x}$ and the j^{th} of $\boldsymbol{\ell}$. The linearization of the prior costs, in (6.96), require more effort. We begin by deriving the (approximate) relationships we require; recalling the relationship from Chapter 3, in (3.42):

$$\mathbf{T}_{b,a} = \exp(\boldsymbol{\xi}_{b,a}^\wedge) \approx \exp(\delta\boldsymbol{\xi}_{b,a}^\wedge)\exp(\bar{\boldsymbol{\xi}}_{b,a}^\wedge) \approx \exp\big((\bar{\boldsymbol{\xi}}_{b,a} + \boldsymbol{\mathcal{J}}(\bar{\boldsymbol{\xi}}_{b,a})^{-1}\delta\boldsymbol{\xi}_{b,a})^\wedge\big), \tag{6.103}$$

it is straightforward to see that

$$\ln(\mathbf{T}_{b,a})^\vee \approx \ln(\bar{\mathbf{T}}_{b,a})^\vee + \boldsymbol{\mathcal{J}}_{b,a}^{-1}\delta\boldsymbol{\xi}_{b,a}, \qquad \boldsymbol{\mathcal{J}}_{b,a} := \boldsymbol{\mathcal{J}}(\ln(\bar{\mathbf{T}}_{b,a})^\vee), \tag{6.104}$$

which leads us to the (approximate) linearization

$$\begin{aligned} \boldsymbol{\mathcal{J}}(\ln(\mathbf{T}_{b,a})^\vee)^{-1}\boldsymbol{\varpi} &\approx \boldsymbol{\mathcal{J}}\big(\ln(\bar{\mathbf{T}}_{b,a})^\vee + \boldsymbol{\mathcal{J}}(\ln(\bar{\mathbf{T}}_{b,a})^\vee)^{-1}\delta\boldsymbol{\xi}_{b,a}\big)^{-1}(\bar{\boldsymbol{\varpi}} + \delta\boldsymbol{\varpi}) \\ &\approx \boldsymbol{\mathcal{J}}_{b,a}^{-1}\bar{\boldsymbol{\varpi}} - \frac{1}{2}\big(\boldsymbol{\mathcal{J}}_{b,a}^{-1}\delta\boldsymbol{\xi}_{b,a}\big)^\curlywedge\bar{\boldsymbol{\varpi}} + \boldsymbol{\mathcal{J}}_{b,a}^{-1}\delta\boldsymbol{\varpi}, \\ &= \boldsymbol{\mathcal{J}}_{b,a}^{-1}\bar{\boldsymbol{\varpi}} + \frac{1}{2}\bar{\boldsymbol{\varpi}}^\curlywedge\boldsymbol{\mathcal{J}}_{b,a}^{-1}\delta\boldsymbol{\xi}_{b,a} + \boldsymbol{\mathcal{J}}_{b,a}^{-1}\delta\boldsymbol{\varpi}. \end{aligned} \tag{6.105}$$

Finally, using the relationship $\delta\boldsymbol{\xi}_{b,a} \approx \delta\boldsymbol{\xi}_b - \boldsymbol{\mathcal{T}}_{b,a}\delta\boldsymbol{\xi}_a$, combined with the expressions in (6.104) and (6.105), our linearized prior costs are

$$\mathbf{e}_{u_k} \approx \begin{bmatrix} \ln(\mathbf{T}_{k+1,i}\mathbf{T}_{k,i}^{-1})^\vee - (t_{k+1} - t_k)\,\bar{\boldsymbol{\varpi}}_k \\ \boldsymbol{\mathcal{J}}(\ln(\mathbf{T}_{k+1,i}\mathbf{T}_{k,i}^{-1})^\vee)^{-1}\,\bar{\boldsymbol{\varpi}}_{k+1} - \bar{\boldsymbol{\varpi}}_k \end{bmatrix} - \mathbf{F}_k\,\delta\mathbf{z}, \tag{6.106}$$

correct to first order, where

$$\mathbf{F}_k = \begin{bmatrix} \boldsymbol{\mathcal{J}}_{k+1,k}^{-1}\bar{\boldsymbol{\mathcal{T}}}_{k+1,k} & (t_{k+1}-t_k)\mathbf{1} & -\boldsymbol{\mathcal{J}}_{k+1,k}^{-1} & \mathbf{0} \\ \frac{1}{2}\boldsymbol{\varpi}_{k+1}^{\wedge}\boldsymbol{\mathcal{J}}_{k+1,k}^{-1}\bar{\boldsymbol{\mathcal{T}}}_{k+1,k} & \mathbf{1} & -\frac{1}{2}\boldsymbol{\varpi}_{k+1}^{\wedge}\boldsymbol{\mathcal{J}}_{k+1,k}^{-1} & -\boldsymbol{\mathcal{J}}_{k+1,k}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{x_k} \\ \mathbf{P}_{x_{k+1}} \end{bmatrix}. \quad (6.107)$$

Although our analytical Jacobian, $\mathbf{F}_k$, uses some fairly approximate relationships, we note that this will only affect the optimal Gauss-Newton *step direction*, $\delta\mathbf{z}^\star$, and not the optimal posterior value, $\hat{\mathbf{z}}$. Continuing with the standard Gauss-Newton approach, we substitute the linearized error terms into the cost function, $J(\bar{\mathbf{z}}+\delta\mathbf{z})$, and take the derivative with respect to $\delta\mathbf{z}$ to reveal the familiar update equation:

$$(\mathbf{A}_{\text{meas}} + \mathbf{A}_{\text{pri}})\,\delta\mathbf{z}^\star = \mathbf{b}_{\text{meas}} + \mathbf{b}_{\text{pri}}, \quad (6.108)$$

where

$$\mathbf{A}_{\text{meas}} := \sum_{kj} \mathbf{G}_{kj}^T \mathbf{R}_{kj}^{-1} \mathbf{G}_{kj}, \qquad \mathbf{b}_{\text{meas}} := \sum_{kj} \mathbf{G}_{kj}^T \mathbf{R}_{kj}^{-1} \mathbf{e}_{m_{kj}}(\bar{\mathbf{z}}), \quad (6.109a)$$

$$\mathbf{A}_{\text{pri}} := \begin{bmatrix} \sum_k \mathbf{F}_k^T \mathbf{Q}_k(t_{k+1})^{-1}\mathbf{F}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \qquad \mathbf{b}_{\text{pri}} := \begin{bmatrix} \sum_k \mathbf{F}_k^T \mathbf{Q}_k(t_{k+1})^{-1}\mathbf{e}_{u_k}(\bar{\mathbf{x}}) \\ \mathbf{0} \end{bmatrix}. \quad (6.109b)$$

Similar to our previous *exactly sparse* STEAM derivations, in Section 6.3, we note that $\mathbf{A}_{\text{meas}}$ is the standard *arrowhead matrix* and the non-zero corner of $\mathbf{A}_{\text{pri}}$ is block-tridiagonal. The MAP solution is then found by solving for the optimal perturbation, $\delta\mathbf{z}^\star$, and updating the best guess of the posterior mean, $\bar{\mathbf{z}}$, iteratively and to convergence. The pertinent state update equations are: $\bar{\mathbf{T}}_{k,i} \leftarrow \exp((\delta\boldsymbol{\xi}_{k,i}^\star)^\wedge)\bar{\mathbf{T}}_{k,i}$, $\bar{\boldsymbol{\varpi}}_k \leftarrow \bar{\boldsymbol{\varpi}}_k + \delta\boldsymbol{\varpi}_k^\star$, and $\bar{\boldsymbol{\ell}} \leftarrow \bar{\boldsymbol{\ell}} + \delta\boldsymbol{\ell}^\star$.

Querying the Trajectory Mean

In order to query the posterior trajectory, $\hat{\mathbf{x}}(t)$, at other times of interest, we now show how the standard linear prediction equations (Rasmussen and Williams, 2006) can be used with our piecewise prior. Beginning with the trajectory mean, we recall the sparse interpolation formulas from (6.31) (Barfoot et al., 2014):

$$\hat{\boldsymbol{\gamma}}_k(\tau) = \check{\boldsymbol{\gamma}}_k(\tau) + \begin{bmatrix} \boldsymbol{\Lambda}(\tau) & \boldsymbol{\Omega}(\tau) \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\gamma}}_k(t_k) - \check{\boldsymbol{\gamma}}_k(t_k) \\ \hat{\boldsymbol{\gamma}}_k(t_{k+1}) - \check{\boldsymbol{\gamma}}_k(t_{k+1}) \end{bmatrix}, \quad t_k \le \tau \le t_{k+1} \quad (6.110)$$

where

$$\boldsymbol{\Lambda}(\tau) = \boldsymbol{\Phi}(t,t_k) - \boldsymbol{\Omega}(\tau)\boldsymbol{\Phi}(t_{k+1},t_k), \quad \boldsymbol{\Omega}(\tau) = \mathbf{Q}_k(\tau)\boldsymbol{\Phi}(t_{k+1},t)^T\mathbf{Q}_k(t_{k+1})^{-1}.$$

Recalling that the terms $\check{\boldsymbol{\gamma}}_k(t_k)$ and $\check{\boldsymbol{\gamma}}_k(t_{k+1})$ do not appear in (6.96), it is unsurprising that the interpolation can be simplified to

$$\hat{\boldsymbol{\gamma}}_k(\tau) = \boldsymbol{\Lambda}(\tau)\hat{\boldsymbol{\gamma}}_k(t_k) + \boldsymbol{\Omega}(\tau)\hat{\boldsymbol{\gamma}}_k(t_{k+1}). \tag{6.111}$$

where

$$\hat{\boldsymbol{\gamma}}_k(t_k) = \begin{bmatrix} \mathbf{0} \\ \hat{\boldsymbol{\varpi}}(t_k) \end{bmatrix}, \; \hat{\boldsymbol{\gamma}}_k(t_{k+1}) = \begin{bmatrix} \ln(\hat{\mathbf{T}}_{k+1,i}\hat{\mathbf{T}}_{k,i}^{-1})^{\vee} \\ \boldsymbol{\mathcal{J}}(\ln(\hat{\mathbf{T}}_{k+1,i}\hat{\mathbf{T}}_{k,i}^{-1})^{\vee})^{-1}\, \hat{\boldsymbol{\varpi}}_{k+1} \end{bmatrix}. \tag{6.112}$$

Reformulating this expression for our *global* state representation, using (6.84) and (6.85), we are able to write the *exact* formulas:

$$\hat{\mathbf{T}}(\tau) = \exp\Big((\boldsymbol{\Lambda}_1(\tau)\hat{\boldsymbol{\gamma}}_k(t_k) + \boldsymbol{\Omega}_1(\tau)\hat{\boldsymbol{\gamma}}_k(t_{k+1}))^{\wedge}\Big)\hat{\mathbf{T}}_{k,i}, \tag{6.113a}$$

$$\hat{\boldsymbol{\varpi}}(\tau) = \boldsymbol{\mathcal{J}}(\ln(\hat{\mathbf{T}}(\tau)\hat{\mathbf{T}}_{k,i}^{-1})^{\vee})\,(\boldsymbol{\Lambda}_2(\tau)\hat{\boldsymbol{\gamma}}_k(t_k)+\boldsymbol{\Omega}_2(\tau)\hat{\boldsymbol{\gamma}}_k(t_{k+1}))\,, \tag{6.113b}$$

where $\boldsymbol{\Lambda}(\tau) = \begin{bmatrix} \boldsymbol{\Lambda}_1(\tau) \\ \boldsymbol{\Lambda}_2(\tau) \end{bmatrix}$, $\boldsymbol{\Omega}(\tau) = \begin{bmatrix} \boldsymbol{\Omega}_1(\tau) \\ \boldsymbol{\Omega}_2(\tau) \end{bmatrix}$. Note that the interpolation of $\hat{\boldsymbol{\varpi}}(\tau)$ depends on $\hat{\mathbf{T}}(\tau)$ due to the term $\boldsymbol{\mathcal{J}}(\ln(\hat{\mathbf{T}}(\tau)\hat{\mathbf{T}}_{k,i}^{-1})^{\vee})$; in essence, it is this element that enforces the nonlinear kinematics during interpolation. The cost to query the trajectory at a single time of interest is $O(1)$, as it involves only the two trajectory states that are temporally adjacent to the query time.

Querying the Trajectory Covariance

In order to interpolate the posterior covariance of our trajectory estimate, we recall the form of the sparse linear prediction equations from (6.31),

$$\hat{\mathbf{P}}_k(\tau,\tau) = \check{\mathbf{P}}_k(\tau,\tau) + \begin{bmatrix} \boldsymbol{\Lambda}(\tau) & \boldsymbol{\Omega}(\tau) \end{bmatrix} \left(\hat{\mathbf{P}}_k^{2\times 2} - \check{\mathbf{P}}_k^{2\times 2}\right) \begin{bmatrix} \boldsymbol{\Lambda}(\tau)^T \\ \boldsymbol{\Omega}(\tau)^T \end{bmatrix}, \tag{6.114}$$

$$\hat{\mathbf{P}}_k^{2\times 2} := \begin{bmatrix} \hat{\mathbf{P}}_k(t_k,t_k) & \hat{\mathbf{P}}_k(t_k,t_{k+1}) \\ \hat{\mathbf{P}}_k(t_k,t_{k+1})^T & \hat{\mathbf{P}}_k(t_{k+1},t_{k+1}) \end{bmatrix}, \quad \check{\mathbf{P}}_k^{2\times 2} := \begin{bmatrix} \check{\mathbf{P}}_k(t_k,t_k) & \check{\mathbf{P}}_k(t_k,t_{k+1}) \\ \check{\mathbf{P}}_k(t_k,t_{k+1})^T & \check{\mathbf{P}}_k(t_{k+1},t_{k+1}) \end{bmatrix},$$

where the new covariance notations, $\check{\mathbf{P}}_k(t,t')$ and $\hat{\mathbf{P}}_k(t,t')$, are used to distinguish that these quantities are related to the *local* variable, $\boldsymbol{\gamma}_k(t) \sim \mathcal{GP}(\check{\boldsymbol{\gamma}}_k(t), \check{\mathbf{P}}_k(t,t'))$; consistent with our previous notation, $\mathbf{P}(t,t')$ is the covariance related to the *global* state, $\mathbf{x}(t)$.

Unlike our previous expressions that involved both posterior and prior terms, we note that the prior covariances, $\check{\mathbf{P}}_k(\tau,\tau)$ and $\check{\mathbf{P}}_k^{2\times 2}$, do not cancel (entirely). Rather than construct them recursively, via (6.24b), we make the retrospective observation that the

exactly sparse covariance interpolation scheme presented in Barfoot et al. (2014) can be simplified by substituting the following *exact* relationships into $\check{\mathbf{P}}_k^{2\times 2}$:

$$\check{\mathbf{P}}_k(t_k, t_{k+1}) = \check{\mathbf{P}}_k(t_k, t_k)\boldsymbol{\Phi}(t_{k+1}, t_k)^T, \tag{6.115a}$$

$$\check{\mathbf{P}}_k(t_{k+1}, t_{k+1}) = \boldsymbol{\Phi}(t_{k+1}, t_k)\check{\mathbf{P}}_k(t_k, t_k)\boldsymbol{\Phi}(t_{k+1}, t_k)^T + \mathbf{Q}_k(t_{k+1}), \tag{6.115b}$$

$$\check{\mathbf{P}}_k(\tau, \tau) = \boldsymbol{\Phi}(\tau, t_k)\check{\mathbf{P}}_k(t_k, t_k)\boldsymbol{\Phi}(\tau, t_k)^T + \mathbf{Q}_k(\tau). \tag{6.115c}$$

Expanding the products of $\check{\mathbf{P}}_k^{2\times 2}$, $\boldsymbol{\Lambda}(\tau)$ and $\boldsymbol{\Omega}(\tau)$, we have the *equivalent* expression,

$$\begin{aligned}\hat{\mathbf{P}}_k(\tau, \tau) &= \begin{bmatrix}\boldsymbol{\Lambda}(\tau) & \boldsymbol{\Omega}(\tau)\end{bmatrix} \hat{\mathbf{P}}_k^{2\times 2} \begin{bmatrix}\boldsymbol{\Lambda}(\tau)^T \\ \boldsymbol{\Omega}(\tau)^T\end{bmatrix} + \check{\boldsymbol{\mathcal{Q}}}_k(\tau), \\ \check{\boldsymbol{\mathcal{Q}}}_k(\tau) &:= \mathbf{Q}_k(\tau) - \mathbf{Q}_k(\tau)\boldsymbol{\Phi}(t_{k+1}, \tau)^T\mathbf{Q}_k(t_{k+1})^{-T}\boldsymbol{\Phi}(t_{k+1}, \tau)\mathbf{Q}_k(\tau)^T,\end{aligned} \tag{6.116}$$

where $\check{\mathbf{P}}_k(\tau, \tau)$ and $\check{\mathbf{P}}_k^{2\times 2}$ have been eliminated in favour of $\check{\boldsymbol{\mathcal{Q}}}_k(\tau)$. Note that this new term, $\check{\boldsymbol{\mathcal{Q}}}_k(\tau)$, captures the *relative* drift of uncertainty over the time interval, using $\mathbf{Q}_k(t)$ from (6.92) – or more generally, (6.23b).

Unfortunately, we are unable to formulate an *exact* expression for the *global* covariance at the query time, $\hat{\mathbf{P}}(\tau, \tau)$, since the posterior covariance block, $\hat{\mathbf{P}}^{2\times 2}$, that we extract from the Gauss-Newton solution, $(\mathbf{A}_{\text{pri}} + \mathbf{A}_{\text{meas}})^{-1}$, must be (approximately) translated to the *local* covariances, $\hat{\mathbf{P}}_k^{2\times 2}$, interpolated for $\hat{\mathbf{P}}_k(\tau, \tau)$, and then (approximately) translated back to $\hat{\mathbf{P}}(\tau, \tau)$. To derive this mapping between the *local* and *global* covariances, $\hat{\mathbf{P}}(t, t')$ and $\hat{\mathbf{P}}_k(t, t')$, we linearize $\boldsymbol{\gamma}_k(t)$, in (6.95), with respect to the involved *global* state variables, $\{\mathbf{x}(t), \mathbf{x}(t_k)\}$ and then take the expected value of the second-moments. This linearization is performed carefully by introducing the intermediate variable:

$$\boldsymbol{\vartheta}(t) := \{\mathbf{T}_{t,k}, \boldsymbol{\varpi}(t)\} = \{\mathbf{T}(t)\mathbf{T}(t_k)^{-1}, \boldsymbol{\varpi}(t)\}, \quad \delta\boldsymbol{\vartheta} = \begin{bmatrix}\delta\boldsymbol{\xi}_{t,k} \\ \delta\boldsymbol{\varpi}(t)\end{bmatrix}. \tag{6.117}$$

Perturbing our expression for $\boldsymbol{\gamma}_k(t)$, in (6.95), with respect to $\boldsymbol{\vartheta}(t)$, we use the relationships we derived in (6.104) and (6.105) to find

$$\boldsymbol{\gamma}_k(t) = \begin{bmatrix}\boldsymbol{\xi}_k(t) \\ \dot{\boldsymbol{\xi}}_k(t)\end{bmatrix} \approx \underbrace{\begin{bmatrix}\ln(\bar{\mathbf{T}}(t)\bar{\mathbf{T}}_{k,i}^{-1})^\vee \\ \boldsymbol{\mathcal{J}}_k(t)^{-1}\,\bar{\boldsymbol{\varpi}}(t)\end{bmatrix}}_{\bar{\boldsymbol{\gamma}}_k(t)} + \underbrace{\begin{bmatrix}\boldsymbol{\mathcal{J}}_k(t)^{-1} & \mathbf{0} \\ \frac{1}{2}\bar{\boldsymbol{\varpi}}(t)^\curlywedge\boldsymbol{\mathcal{J}}_k(t)^{-1} & \boldsymbol{\mathcal{J}}_k(t)^{-1}\end{bmatrix}}_{\boldsymbol{\Gamma}(t)} \delta\boldsymbol{\vartheta}(t), \tag{6.118}$$

where

$$\boldsymbol{\mathcal{J}}_k(t) := \boldsymbol{\mathcal{J}}(\ln(\bar{\mathbf{T}}(t)\bar{\mathbf{T}}_{k,i}^{-1})^\vee), \quad \boldsymbol{\Gamma}(t)^{-1} = \begin{bmatrix}\boldsymbol{\mathcal{J}}_k(t) & \mathbf{0} \\ -\frac{1}{2}\boldsymbol{\mathcal{J}}_k(t)\bar{\boldsymbol{\varpi}}(t)^\curlywedge & \boldsymbol{\mathcal{J}}_k(t)\end{bmatrix}. \tag{6.119}$$

Taking the expected value of the second moment, we have the relationship

$$\mathbf{P}_k(t,t') = E[(\boldsymbol{\gamma}_k(t) - \bar{\boldsymbol{\gamma}}_k(t))(\boldsymbol{\gamma}_k(t') - \bar{\boldsymbol{\gamma}}_k(t'))^T] \approx \boldsymbol{\Gamma}(t) E[\delta\boldsymbol{\vartheta}(t)\delta\boldsymbol{\vartheta}(t')^T]\boldsymbol{\Gamma}(t')^T. \quad (6.120)$$

In order to relate $\delta\boldsymbol{\vartheta}(t)$ to $\delta\mathbf{x}(t)$, we use $\delta\boldsymbol{\xi}_{b,a} \approx \delta\boldsymbol{\xi}_b - \boldsymbol{\mathcal{T}}_{b,a}\delta\boldsymbol{\xi}_a$ to find

$$\delta\boldsymbol{\vartheta}(t) = \begin{bmatrix} \delta\boldsymbol{\xi}_{t,k} \\ \delta\boldsymbol{\varpi}(t) \end{bmatrix} \approx \begin{bmatrix} \delta\boldsymbol{\xi}(t) - \bar{\boldsymbol{\mathcal{T}}}_{t,k}\delta\boldsymbol{\xi}(t_k) \\ \delta\boldsymbol{\varpi}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} \delta\boldsymbol{\xi}(t) \\ \delta\boldsymbol{\varpi}(t) \end{bmatrix}}_{\delta\mathbf{x}(t)} - \underbrace{\begin{bmatrix} \bar{\boldsymbol{\mathcal{T}}}_{t,k} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\boldsymbol{\Xi}(t)} \underbrace{\begin{bmatrix} \delta\boldsymbol{\xi}(t_k) \\ \delta\boldsymbol{\varpi}(t_k) \end{bmatrix}}_{\delta\mathbf{x}(t_k)}. \quad (6.121)$$

Rearranging this expression for $\delta\mathbf{x}(t)$, $\delta\mathbf{x}(t) \approx \delta\boldsymbol{\vartheta}(t) + \boldsymbol{\Xi}(t)\delta\mathbf{x}(t_k)$, it follows that,

$$\mathbf{P}(t,t') = E[\delta\mathbf{x}(t)\delta\mathbf{x}(t')^T] \approx E[(\delta\boldsymbol{\vartheta}(t) + \boldsymbol{\Xi}(t)\delta\mathbf{x}(t_k))(\delta\boldsymbol{\vartheta}(t') + \boldsymbol{\Xi}(t')\delta\mathbf{x}(t_k))^T]. \quad (6.122)$$

Analyzing the cross-covariance over the time interval $t_k \leq t' \leq t_{k+1}$, we note

$$\begin{aligned} E[\delta\mathbf{x}(t_k)\delta\boldsymbol{\vartheta}(t')^T] &= \begin{bmatrix} E[\delta\boldsymbol{\xi}(t_k)\delta\boldsymbol{\xi}_{t',k}^T] & E[\delta\boldsymbol{\xi}(t_k)\delta\boldsymbol{\varpi}(t')^T] \\ E[\delta\boldsymbol{\varpi}(t_k)\delta\boldsymbol{\xi}_{t',k}^T] & E[\delta\boldsymbol{\varpi}(t_k)\delta\boldsymbol{\varpi}(t')^T] \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ E[\delta\boldsymbol{\varpi}(t_k)\delta\boldsymbol{\xi}_{t',k}^T] & E[\delta\boldsymbol{\varpi}(t_k)\delta\boldsymbol{\varpi}(t')^T] \end{bmatrix}. \end{aligned} \quad (6.123)$$

Recalling the structure of $\boldsymbol{\Xi}(t)$, we find that

$$\boldsymbol{\Xi}(t) E[\delta\mathbf{x}(t_k)\delta\boldsymbol{\vartheta}(t')^T] = \mathbf{0}, \quad (6.124)$$

and therefore the expansion of (6.122) is simplified to

$$\mathbf{P}(t,t') \approx E[\delta\boldsymbol{\vartheta}(t)\delta\boldsymbol{\vartheta}(t')^T] + \boldsymbol{\Xi}(t)\mathbf{P}(t_k,t_k)\boldsymbol{\Xi}(t')^T. \quad (6.125)$$

Finally, relating (6.120) and (6.125) through $E[\delta\boldsymbol{\vartheta}(t)\delta\boldsymbol{\vartheta}(t')^T]$, we arrive at

$$\mathbf{P}_k(t,t') = \boldsymbol{\Gamma}(t)\Big(\mathbf{P}(t,t') - \boldsymbol{\Xi}(t)\mathbf{P}(t_k,t_k)\boldsymbol{\Xi}(t')^T\Big)\boldsymbol{\Gamma}(t')^T, \quad (6.126)$$

which can also be rearranged for

$$\mathbf{P}(t,t') = \boldsymbol{\Gamma}(t)^{-1}\mathbf{P}_k(t,t')\boldsymbol{\Gamma}(t')^{-T} + \boldsymbol{\Xi}(t)\mathbf{P}(t_k,t_k)\boldsymbol{\Xi}(t')^T. \quad (6.127)$$

Notably, this completes our ability to *approximately* interpolate for $\hat{\mathbf{P}}(\tau,\tau)$: we first use (6.126) to translate $\hat{\mathbf{P}}^{2\times 2}$ to $\hat{\mathbf{P}}_k^{2\times 2}$, then interpolate the *local* posterior covariance, $\hat{\mathbf{P}}_k(\tau,\tau)$, using (6.116), and finally use (6.127) to calculate $\hat{\mathbf{P}}(\tau,\tau)$.

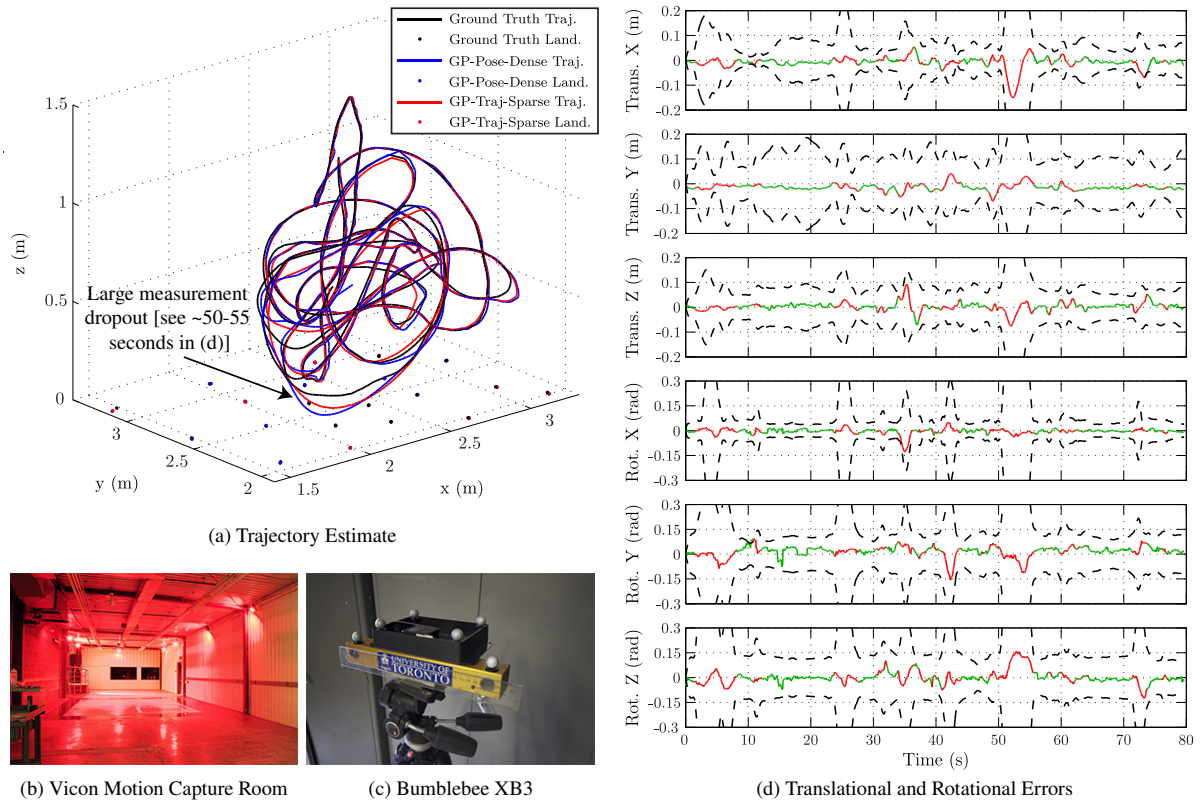
6.4.4 Stereo Camera Experiment

In this section, we experimentally validate the proposed estimator on a STEAM problem. For comparison, we also implement the (dense) Gaussian-process regression approach taken by Tong et al. (2014). The dataset that we utilize was gathered by rotating and translating a stereo camera above 20 reflective markers (that act as our point landmarks). The position and orientation of the camera, as well as the positions of the markers were tracked with a Vicon motion capture system. Stereo images were logged at roughly 15 Hz and data associations were provided by the ground truth. It is interesting to validate our approach on this dataset (despite its small scale) for two reasons: (i) ground truth is available for both the trajectory *and* the map, and (ii) the pose of the camera is often underconstrained (without a motion prior) due to the small number of landmarks observed in each frame; over 40% of the stereo images in the dataset contain less than three landmarks (the minimum number for a unique pose solution).

Since our hyperparameters, $\check{\boldsymbol{\varpi}}_0$ and $\mathbf{Q}_C$, have physical meaning for our GP prior, we chose to set $\check{\boldsymbol{\varpi}}_0 = \mathbf{0}$ (a fair assumption, given no *a priori* information) and model $\mathbf{Q}_C$ as a diagonal matrix (with its elements populated by Gaussians fit to training data).

Trajectory Estimate

The estimator described by Tong et al. (2014), *GP-Pose-Dense*, as well as our piecewise method, *GP-Traj-Sparse*, are evaluated using 1000 stereo images (roughly 80 seconds of data). The estimated trajectories are displayed in Figure 6.10. Notably, the mean trajectories produced by the algorithms are very similar, even in time segments that relied heavily on the GP prior. Although we expect that our (physically motivated) prior, *GP-Traj-Sparse*, will provide a more accurate result than *GP-Pose-Dense* on many actuated robotic systems, this is an acceptable result since our main advantage is computational efficiency. To verify quantitatively that our estimator is performing well, both translational and rotational errors are shown in Figure 6.10(d). The trajectory estimate and ground truth are aligned by minimizing the squared error between the estimated and ground truth landmark positions. The 3σ covariance envelope is plotted in each result to verify that our estimate is consistent. When a unique solution is available without using the GP prior, the error is plotted in green; when the GP prior is required (two or fewer visible landmarks), we expect some drift in the estimate and the error is plotted in red. We can see that the covariance grows in the red areas and shrinks in the green areas.



(a) Trajectory Estimate

(b) Vicon Motion Capture Room

(c) Bumblebee XB3

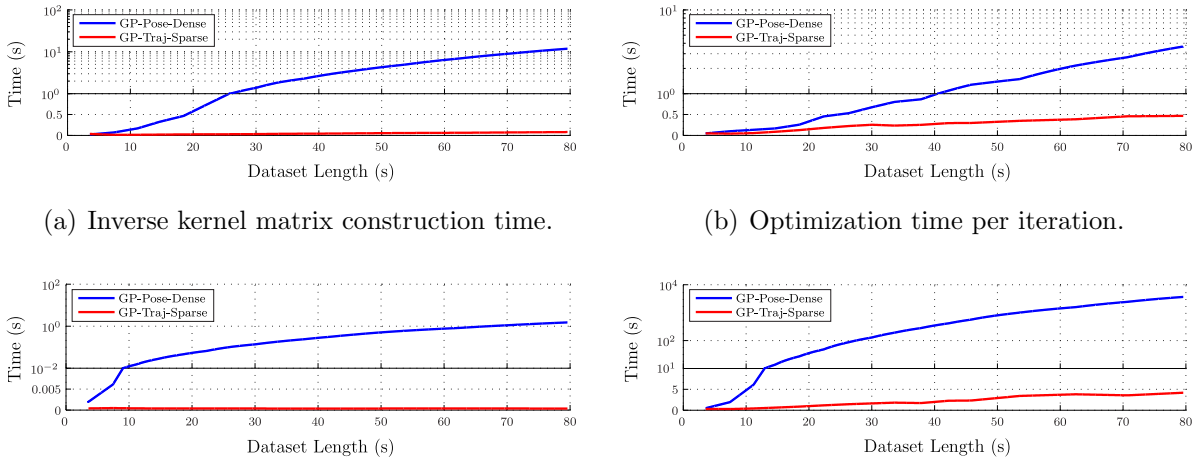
(d) Translational and Rotational Errors

Figure 6.10: This figure depicts our trajectory estimation experiment using a stereo camera. (a) The trajectory and landmark positions for the 1000 image (roughly 80 seconds) dataset. The trajectory estimates from the *GP-Pose-Dense* and *GP-Traj-Sparse* algorithms are shown in blue and red, respectively, and the ground truth (from Vicon) is shown in black. (b) Photograph of our motion capture room. (c) Photograph of the stereo camera with IR tracking balls. (d) Error plots depicting the metric accuracy of the proposed *GP-Traj-Sparse* algorithm. The error at poses that simultaneously observe three or more landmarks are plotted in green, less than three in red; note the long observation dropout around 50-55 seconds (also annotated in (a)). To show that our estimate is *consistent*, the 3σ covariance envelopes are plotted as black dashed lines.

Computational Efficiency

To verify the computational savings introduced by our *Markovian* state and prior, both estimators were implemented in Matlab and tested on a 2.4GHz i7 processor with 8GB of 1333MHz DDR3 RAM. State variables are placed at the 1000 stereo-image times (over 80 seconds) and the final solution is queried at 30Hz.

To visualize the computational complexity associated with each phase of the algorithms, the estimators were run multiple times for increasingly long subsets of the dataset; the results of this timing experiment are shown in Figure 6.11. Since we have many more measurement times, K, than landmarks, L, the *arrowhead* structure of the problem can



(a) Inverse kernel matrix construction time.

(b) Optimization time per iteration.

(c) Interpolation time per additional query time.

(d) Total computation time.

Figure 6.11: Comparison of the computation time required by both the *GP-Pose-Dense* algorithm (as described by Tong et al. (2014)) and the *GP-Traj-Sparse* algorithm. The algorithms were tested on varying dataset lengths. State variables were placed at the stereo image times (roughly 15Hz) and then interpolated at 30Hz. The plots confirm the theoretical complexities of the various algorithm phases. The total compute time required by the *GP-Traj-Sparse* algorithm is linear in trajectory length. Note the linear-to-log scale transition in each plot.

be exploited by our *GP-Traj-Sparse* algorithm to solve the system in $O(L^3 + L^2K)$ time, using sparse lower-upper Cholesky decomposition and forward-backward passes. The final solution is then queried J times (each in $O(1)$ time), making the total algorithm complexity $O(L^3 + L^2K + J)$. Solving for only pose, the *GP-Pose-Dense* algorithm has a half-size state vector, but a dense $\check{\mathbf{P}}^{-1}$; furthermore, no special form exists for $\check{\mathbf{P}}^{-1}$ and it is computed by inverting the dense covariance matrix, $\check{\mathbf{P}}$, which is a $O(K^3)$ operation.

Interpolation Quality

The last feature of our algorithm that we wish to demonstrate is its ability to predict the state (and uncertainty) at times for which we do not have any measurements. To test this, we repeated the batch estimation experiment, using the 80-second dataset, but removed a block of 30 stereo images (spanning a roughly 2.3 second duration). Post optimization, we interpolate the mean using (6.113) and the covariance using (6.127). The result, near the subsection of interest, is shown in Figure 6.12. The estimated mean and covariance (at the measurement times) are shown in blue. The interpolated mean (dashed red line) does a reasonable job of predicting the intermediate behaviour shown by the ground truth (in

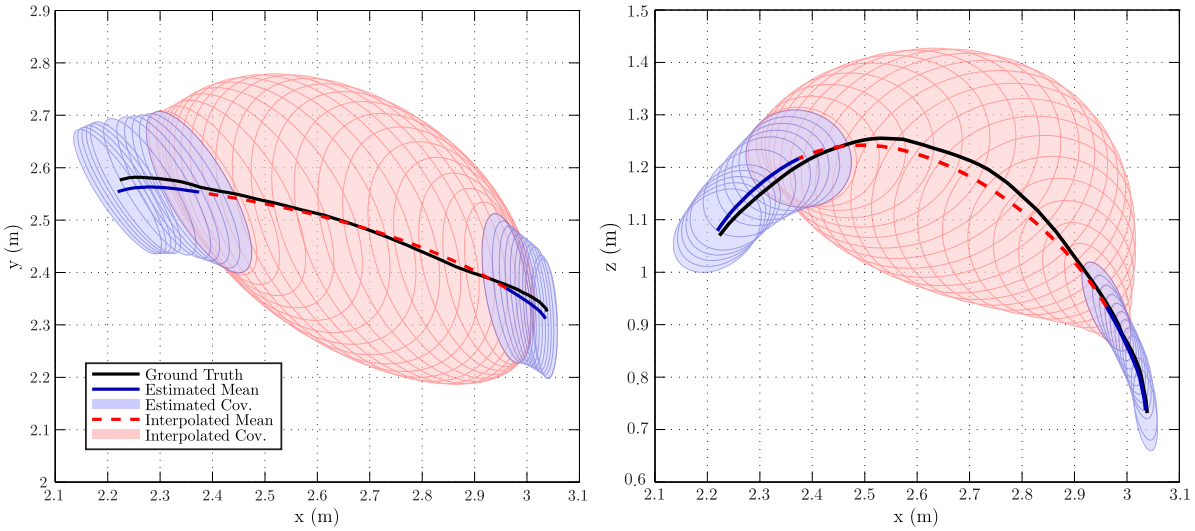


Figure 6.12: This plot shows interpolated mean and covariance estimates from the *GP-Traj-Sparse* algorithm. For this experiment, the dataset was modified by removing 30 images (spanning roughly 2.3 seconds) from the trajectory section depicted in red. The mean and covariance of state variables that bounds the region of interest are displayed in blue. The interpolated mean (dashed red line) and covariance (shaded red ellipses) depict our prediction of the unobserved behaviour between the bounding posterior estimates. Covariances are depicted using 3σ ellipses.

black). Furthermore, we note that the covariance interpolation (shaded red) is *continuous* and reflects the growth of uncertainty from the prior between the two endpoints.

6.4.5 Practical Extensions

Moving forward, we recall that our original motivation for the continuous-time problem formulation was to use the appearance-based lidar technology described in Chapter 2. Unlike the problems we have tackled with GPs thus far, the VO-style problem (with a scanning-type sensor) contains *many* measurement times and *many* landmarks parameters. Furthermore, we also wish to leverage place recognition results in order to improve our local metric accuracy and provide topometric information in our STEAM solution. To accomplish these goals, we describe two practical extensions to our GP formulation: (i) *exactly sparse* measurement interpolation, and (ii) a method to adopt the *relative* coordinate formulation of STEAM (similar to the work of Sibley et al. (2009)).

Interpolating Measurement Times

First, in order to keep computation time tractable we will leverage the heuristic *keyframing* strategy proposed by Tong et al. (2014) (previously discussed in Section 6.2.4). Rather

than discretize the state trajectory, $\mathbf{x}(t)$, at the K measurement times, this strategy advocates discretizing the state at a much coarser resolution of *keytimes*, $N \ll K$. Recalling the general form of our linearized observation model:

$$\mathbf{e}_{m_{kj}}(\bar{\mathbf{z}} + \delta\mathbf{z}) \approx \mathbf{y}_{kj} - \mathbf{g}(\bar{\mathbf{T}}(t_k),\, \bar{\boldsymbol{p}}_i^{\ell_j,i}) - \mathbf{G}_{\mathbf{x},kj}\,\delta\boldsymbol{\xi}(t_k) - \mathbf{G}_{\boldsymbol{\ell},kj}\,\delta\boldsymbol{\ell}_j, \tag{6.128}$$

we evaluate the observation model, $\mathbf{g}(\cdot)$, and Jacobians, $\mathbf{G}_{\mathbf{x},kj}$ and $\mathbf{G}_{\boldsymbol{\ell},kj}$, by exploiting the GP prediction equation, (6.113a), to calculate the *interpolated* transform, $\bar{\mathbf{T}}(t_k)$, $t_n \leq t_k < t_{n+1}$. Since our trajectory state is no longer discretized at time t_k, we must also relate the 'interpolated' perturbation, $\delta\boldsymbol{\xi}(t_k)$, back to the discretized state, $\mathbf{x}$; note that our *exactly sparse* interpolation relies only on the temporally adjacent variables, $\mathbf{x}_n = \{\mathbf{T}_{n,i}, \boldsymbol{\varpi}_n\}$, $\mathbf{x}_{n+1} = \{\mathbf{T}_{n+1,i}, \boldsymbol{\varpi}_{n+1}\}$. Sparing the laborious details, we note that by using our standard constraint-sensitive perturbations, along with the relationships we derived in (6.118) and (6.121), we find that

$$\delta\boldsymbol{\xi}(t_k) \approx \begin{bmatrix}\boldsymbol{\mathcal{A}}_n(t_k) & \boldsymbol{\mathcal{B}}_n(t_k)\end{bmatrix}\delta\mathbf{x}_{n+1} + \begin{bmatrix}\boldsymbol{\mathcal{C}}_n(t_k) & \boldsymbol{\mathcal{D}}_n(t_k)\end{bmatrix}\delta\mathbf{x}_n, \quad t_n \leq t_k < t_{n+1}, \tag{6.129}$$

with the analytical elements,

$$\begin{aligned}
\boldsymbol{\mathcal{A}}_n(t_k) &:= \boldsymbol{\mathcal{J}}_n(t_k)\big(\boldsymbol{\Omega}_{11}(t_k) + \tfrac{1}{2}\boldsymbol{\Omega}_{12}(t_k)\boldsymbol{\varpi}_{n+1}^{\curlywedge}\big)\boldsymbol{\mathcal{J}}_n(t_{n+1})^{-1},\\
\boldsymbol{\mathcal{B}}_n(t_k) &:= \boldsymbol{\mathcal{J}}_n(t_k)\boldsymbol{\Omega}_{12}(t_k)\boldsymbol{\mathcal{J}}_n(t_{n+1})^{-1}, \qquad \boldsymbol{\mathcal{C}}_n(t_k) := \bar{\boldsymbol{\mathcal{T}}}_{t_k,t_n} - \boldsymbol{\mathcal{A}}_n(t_k)\bar{\boldsymbol{\mathcal{T}}}_{t_{n+1},t_n}, \qquad (6.130)\\
\boldsymbol{\mathcal{D}}_n(t_k) &:= \boldsymbol{\mathcal{J}}_n(t_k)\boldsymbol{\Lambda}_{12}(t_k), \qquad \text{where} \qquad \boldsymbol{\mathcal{J}}_n(t) := \boldsymbol{\mathcal{J}}(\ln(\bar{\mathbf{T}}(t)\bar{\mathbf{T}}_{n,i}^{-1})^{\vee}).
\end{aligned}$$

From a factor-graph perspective, we note that this causes our (originally) binary observation factors (dependent on $\mathbf{x}_k$ and $\boldsymbol{\ell}_j$) to become ternary factors (dependent on $\mathbf{x}_n$, $\mathbf{x}_{n+1}$, and $\boldsymbol{\ell}_j$). Concerning the sparsity of our Gauss-Newton problem, the (now) ternary observation factors cause the *top-left corner* of the *arrowhead matrix*, $\mathbf{G}^T\mathbf{R}^{-1}\mathbf{G}$, to become block-tridiagonal (rather than block-diagonal). Since this new sparsity matches our (already) block-tridiagonal prior term, $\mathbf{F}^{-T}\mathbf{Q}^{-1}\mathbf{F}^{-1}$, there is no adverse effect on the total sparsity pattern. In fact, by estimating the state at N *keytimes*, $N \ll K$, and exploiting the sparsity of $\mathbf{A}_{\ell\ell}$, the batch Gauss-Newton iterations become order $O(N^3 + N^2L + K)$, which is the same complexity as a traditional discrete-time SLAM problem (with N *keyframes* and K measurements).

In this thesis we make the retrospective observation that the application of this measurement interpolation scheme (in essence) transforms our nonparametric GP model into a parametric one (using basis functions). Revisiting the general GP interpolation

equation, (6.17a), from this new perspective (where the discretized state, $\mathbf{x}$, does not necessarily coincide with measurement times), we draw a similarity to the parametric approach, discussed in Chapter 5, (5.1):

$$\mathbf{x}(\tau) = \underbrace{\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}}_{\boldsymbol{\Psi}(t)}\mathbf{x} + \underbrace{\check{\mathbf{x}}(\tau) - \check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}\check{\mathbf{x}}}_{\text{time-varying constant}}, \tag{6.131}$$

where $\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}$ can be viewed as a set of parametric basis functions, $\boldsymbol{\Psi}(t)$, *weighted* by state vector $\mathbf{x}$. Note that the additive, time-varying constant is simply *zero* for linear GP priors that do not include control inputs (such as our piecewise, constant-velocity GP prior). Unlike standard parametric approaches that tend to choose an interpolation scheme ad hoc, we reiterate that the *interpolation kernel*, $\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}$, can be derived directly from a LTV SDE. We assert that this is a more principled way to arrive at a parametric model, since the (now) physically motivated *basis*, $\boldsymbol{\Psi}(t)$, can also be viewed as a smoothness constraint that enforces the optimal trajectory prediction between the *keytimes* (using the continuous-time motion model).

To deepen this relationship, we derive the exact form of the *basis* for the white-noise-on-acceleration prior used by our piecewise GP approach in Section 6.4.3. Recalling that our LTI SDE results in a $\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}$ with the two non-zero block-column entries

$$\begin{aligned}
\boldsymbol{\Lambda}(\tau) &= \boldsymbol{\Phi}(\tau, t_k) - \mathbf{Q}_k(\tau)\boldsymbol{\Phi}(t_{k+1}, \tau)^T\mathbf{Q}_k(t_{k+1})^{-1}\boldsymbol{\Phi}(t_{k+1}, t_k), \\
\boldsymbol{\Omega}(\tau) &= \mathbf{Q}_k(\tau)\boldsymbol{\Phi}(t_{k+1}, \tau)^T\mathbf{Q}_k(t_{k+1})^{-1},
\end{aligned}$$

and analytical expressions for $\boldsymbol{\Phi}(t, t')$ and $\mathbf{Q}_k(t)$, in (6.92) and (6.89), it follows that:

$$\boldsymbol{\Lambda}(\tau) = \begin{bmatrix} \boldsymbol{\Lambda}_{11}(\tau) & \boldsymbol{\Lambda}_{12}(\tau) \\ \boldsymbol{\Lambda}_{21}(\tau) & \boldsymbol{\Lambda}_{22}(\tau) \end{bmatrix} = \begin{bmatrix} (2\lambda^3 - 3\lambda^2 + 1)\mathbf{1} & \Delta(\lambda^3 - 2\lambda^2 + \lambda)\mathbf{1} \\ \frac{6}{\Delta}(\lambda^2 - \lambda)\mathbf{1} & (3\lambda^2 - 4\lambda + 1)\mathbf{1} \end{bmatrix} \tag{6.132a}$$

$$\boldsymbol{\Omega}(\tau) = \begin{bmatrix} \boldsymbol{\Omega}_{11}(\tau) & \boldsymbol{\Omega}_{12}(\tau) \\ \boldsymbol{\Omega}_{21}(\tau) & \boldsymbol{\Omega}_{22}(\tau) \end{bmatrix} = \begin{bmatrix} (3\lambda^2 - 2\lambda^3)\mathbf{1} & \Delta(\lambda^3 - \lambda^2)\mathbf{1} \\ \frac{6}{\Delta}(\lambda - \lambda^2)\mathbf{1} & (3\lambda^2 - 2\lambda)\mathbf{1} \end{bmatrix} \tag{6.132b}$$

where $\lambda := \frac{\tau - t_n}{t_{n+1} - t_n}$, $\Delta := t_{n+1} - t_n$. Although this cubic interpolation scheme may not seem special, we note that the top rows of coefficients form the *exact* expression used by *cubic Hermite splines* and that the bottom rows are simply the derivatives. While we advocate that the nonparametric approach to STEAM is more probabilistically justified, we note that in situations where a parametric model must be used (e.g., for computational reasons), it is motivating to generate the *basis* using the GP approach presented herein[1].

[1]For the mathematically lazy, we suggest using cubic Hermite splines for parametric estimation since they are physically motivated, easy to implement, and efficient (with a local support of only *two*).

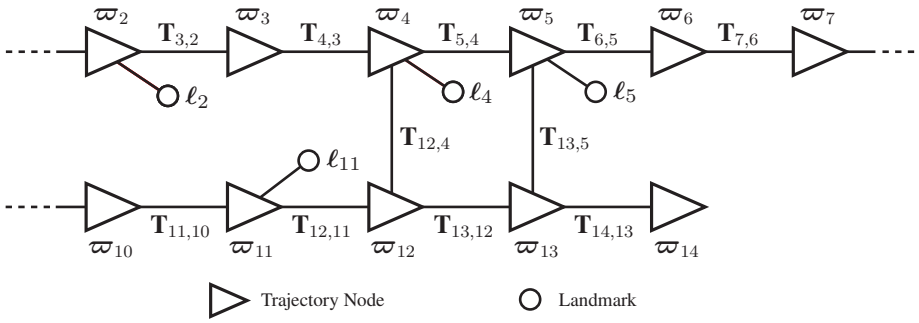
Adopting the Relative Pose-Graph

In order to process large-scale loop closures in constant time (similar to the relative, continuous-time method described in Chapter 5), we propose a hybrid approach to incremental STEAM that maintains a *relative* state representation for the trajectory at storage time (see Figure 6.13a), but converts to a *privileged* coordinate frame representation to perform incremental, window-style batch optimizations (see Figure 6.13b). Recalling that our trajectory state discretization, $\mathbf{x}$, is already in the standard discrete-time form, we note that it is trivial to convert our state parameterization to that of a relative pose-graph (where we must also store body-centric velocities at the graph nodes). However, when it comes to performing a window-style batch optimization (as described in Section 3.3.2), we must choose to either: (i) re-factor the GP prior cost terms, in (6.96), and the interpolation equations, in (6.113), to instead use the relative pose variables, $\mathbf{T}_{t_{k+1},t_k}$, or (ii) convert the *window* of relative poses that we wish to optimize into a privileged coordinate frame. Notably, the former approach can be used to augment an existing implementation of discrete-time, relative SLAM (Sibley et al., 2009) with our *exactly sparse* set of GP prior smoothing terms. However, we present two subtle advantages to taking the second approach (which also apply to the discrete-time case).

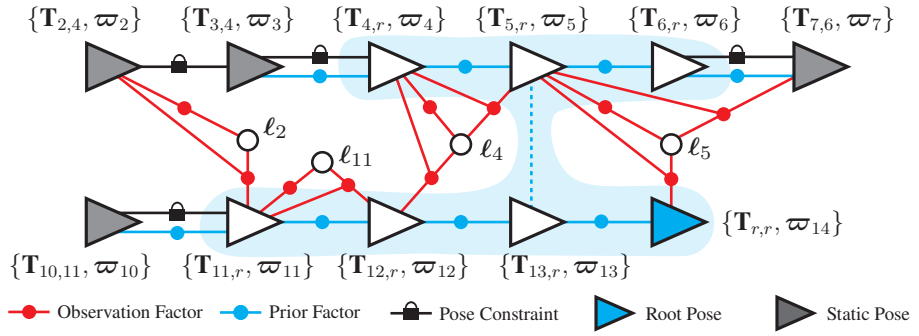
The first advantage concerns the sparsity of the incremental batch estimation problems. After selecting an *active* region of poses for the incremental batch optimization, it is trivial to compound the relative pose chains back into a privileged *root* frame, $\underrightarrow{\mathcal{F}}_{\text{rob}}(t_r)$, where the *root time*, t_r, aligns with one of the pose-graph node times, t_k. Recalling that the *relative* landmark parameters, $\boldsymbol{p}_{t_j}^{\ell_j,t_j}$, are expressed in the frame where they are first observed, $\underrightarrow{\mathcal{F}}_{\text{rob}}(t_j)$, we note that the observation cost terms have the form:

$$\mathbf{e}_{m_{kj}}(\mathbf{z}) = \mathbf{y}_{kj} - \mathbf{g}(\mathbf{T}_{t_k,t_j}\,\boldsymbol{p}_{t_j}^{\ell_j,t_j}) = \mathbf{y}_{kj} - \mathbf{g}(\mathbf{T}_{t_k,t_r}\mathbf{T}_{t_j,t_r}^{-1}\,\boldsymbol{p}_{t_j}^{\ell_j,t_j}), \tag{6.133}$$

where we have used the *root frame* to break the relative transforms, $\mathbf{T}_{t_k,t_j}$, into exactly two pose variables (rather than a chain of relative poses); by extension, this also results in a much sparser set of Jacobians (for long feature tracks). Post optimization, the pose changes can be extracted back into the relative storage format by using $\mathbf{T}_{t_{k+1},t_k} = \mathbf{T}_{t_{k+1},t_r}\mathbf{T}_{t_k,t_r}^{-1}$. Note that while even more sparsity can be obtained by temporarily expressing the landmarks in the *root* frame, we find that the relative landmark formulation reduces the number of required Gauss-Newton iterations *dramatically*. The intuition behind this is that a *global* landmark representation (with a poor initial condition) can act as a damper that causes the iterative trajectory updates to take small steps. In contrast, the *global*



(a) In order to store our *trajectory* state in a relative fashion, the standard relative pose-graph is augmented to include body-centric velocity estimates at the nodes. Also, note that the landmark variables are stored relative to the first frame in which they are observed.



(b) This figure illustrates a factor-graph view of our local batch optimization. Note that the *active* region (shaded blue) of the relative pose-graph is transitioned to use the *root* pose as a privileged coordinate frame. In contrast, *static* poses are expressed with respect to their nearest *active* leaf node; in essence, these *static regions* form fixed submaps that float with the *active* pose to which they are constrained. Ternary observation factors arise due to the landmarks being expressed in a relative fashion. Lastly, we note that prior factors in parallel with a pose constraint still affect the *active* velocity variables.

Figure 6.13: This figure depicts our state representation at storage and optimization time.

structure of *relative landmarks* deforms 'freely' with changes to the trajectory estimate.

The second advantage of our approach involves the case where a *loop* exists within the *active* sub-graph of relative poses that we are optimizing (i.e., when the *active* state is over-parameterized). Rather than *choose* a minimal parameterization of relative states to optimize, we note that converting to a set of privileged coordinates naturally enforces a *locally consistent* solution.

In summary, our incremental STEAM technique operates in five stages: (i) perform a breadth-first search from a *root* node to select an *active* region of the relative pose graph to optimize, (ii) determine the *active* landmarks and the corresponding *static* region of poses, (iii) concatenate the relative poses into a *temporary*, privileged state representation, (iv) perform a *local* batch optimization using all of the relevant observation factors, as

well as the GP prior factors between temporally adjacent trajectory states, and (v) push updated pose, velocity, and landmark estimates back into the relative storage format.

6.4.6 Appearance-Based Lidar Experiment

In this section, we validate that our piecewise GP approach, combined with measurement interpolation and our relative storage method, can be used to perform incremental STEAM with high-rate asynchronous measurements in unstructured three-dimensional terrain. Once again, we employ the use of our 1.1km appearance-based lidar dataset described in Appendix A. Comparison is performed against an open-loop, discrete-time solution and the closed-loop (parametric, relative) continuous-time method described in Chapter 5. Furthermore, all compared estimators use the same SURF feature extractions, preprocessed feature associations (i.e., outlier rejection), and loop closure proposals.

In contrast to the standard parametric approach, which focuses on proper knot spacing to model smoothness (recall the tedious training in Chapter 5), our GP approach will primarily rely on the prior model to prevent overfitting. Notably, spacing the *keytimes* too far apart will still cause underfitting (i.e., over-smoothing); however, assuming that the GP prior is well tuned, placing *keytimes* too close together will only have an adverse affect on computational performance. Therefore, we must simply choose a knot spacing large enough to enable *online* operation, but not too large as to over-smooth the solution. For convenience, we choose to place a keytime at each lidar imagestack acquisition time (a roughly half-second spacing) and found no adverse effects on accuracy. Concerning the hyperparameters, we set our initial velocity prior, $\check{\boldsymbol{\varpi}}_0$, to *zero* (with a high uncertainty), and model $\mathbf{Q}_C$ as a diagonal matrix with its elements populated by Gaussians fit to training data. Recalling that our GP prior approximates the nonlinear kinematic SDE in (6.67), we note that the diagonal elements of $\mathbf{Q}_C$ can be interpreted as the *variances* of the six body-centric acceleration (in rotation and translation).

Trajectory Estimation Performance

The open-loop trajectory estimates produced by our piecewise GP algorithm, *GP-Traj-Sparse*, the parametric continuous-time algorithm from Chapter 5, *Relative CT*, and a simple discrete-time implementation, *Discrete*, can be seen in Figure 6.14. The roughly 53 minute and 1.1km traversal contains 6880 intensity images, with 2,010,671 temporally matched SURF features associated with 688,481 unique landmarks. As expected, both

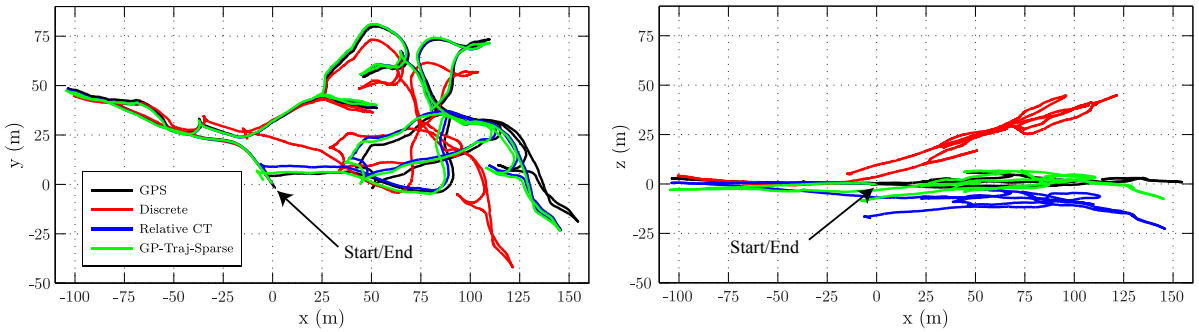


Figure 6.14: Top and side views of the open-loop trajectory estimation results for the 1.1km dataset. DGPS tracks are in black, the discrete-time estimate is in red, the parametric *Relative CT* estimate (from Chapter 5) is in blue, and the *GP-Traj-Sparse* estimate (using measurement interpolation) is in green.

continuous-time solutions perform much better than the discrete-time counterpart; note that the discrete solution does not perform motion compensation in batch estimation, but does leverage the motion-compensated RANSAC outlier rejection (described in Chapter 4). Notably, both continuous-time solutions produce very similar results in the x-y plane; we attribute the *GP-Traj-Sparse* algorithm's improved elevation estimate to the trained GP prior, which restricts large *jumps* in both the robo-centric z-axis and pitch velocity.

Since our continuous-time solutions also take advantage of loop-closures (in a topo-metric sense), we are interested in comparing their local metric accuracy. Recalling the ALARMS metric presented in Section 5.3.5, we provide a comparison in Figure 6.15. Although the open-loop integration (OLI) result of our *GP-Traj-Sparse* algorithm is much better that the OLI result of the *Relative CT* algorithm, we note that the improvement in closed-loop integration (CLI) is marginal. This suggests that *locally* accurate results can be obtained with either form of motion-compensation.

Despite the trajectory estimate of our GP approach being only marginally better that the parametric method in Chapter 5, we note two other improvements: (i) a cleaner state discretization, and (ii) faster computational performance. With respect to state discretization, we note that being able to use the (fairly) standard relative pose-graph storage (depicted in Figure 6.13) is much simpler than book-keeping overlapping basis functions. Furthermore, it allows many existing discrete-time SLAM implementations to be augmented with our GP prior smoothing terms and interpolation scheme. In contrast, we note that using cubic B-splines (or other basis functions with a *local support* greater than two) causes a tedious amount of book-keeping and *special* storage.

Although it is somewhat unfair to compare our computational results directly, since

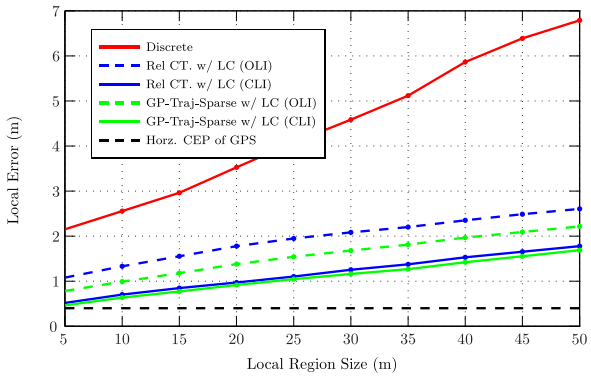
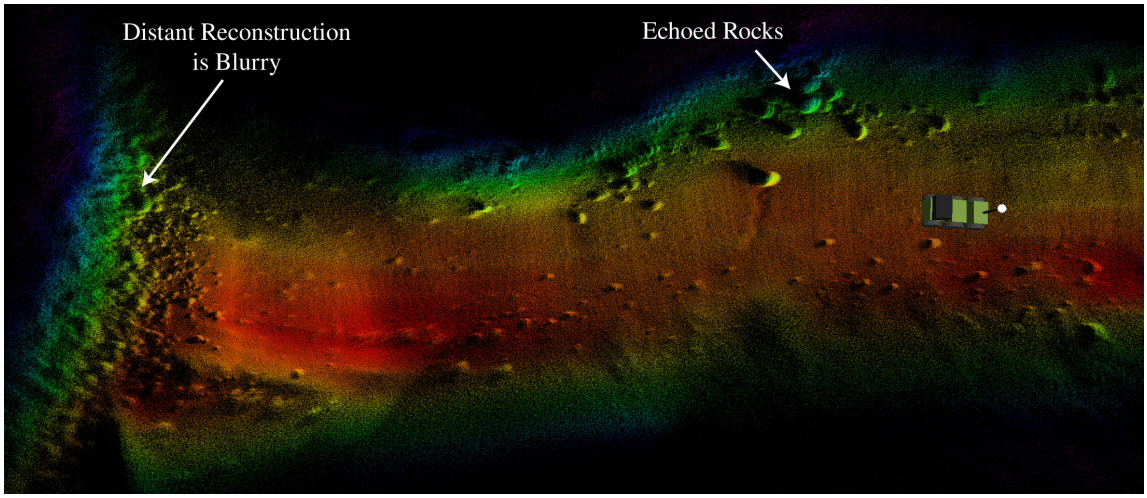


Figure 6.15: This figure shows the ALARMS error for a varying local region size. The continuous-time estimates perform much better than the discrete-time counterpart. Furthermore, utilizing the topological information to perform closed-loop integration results in the most accurate local estimates. Notably, our *GP-Traj-Sparse* algorithm performs marginally better than the parametric *Relative CT* algorithm.

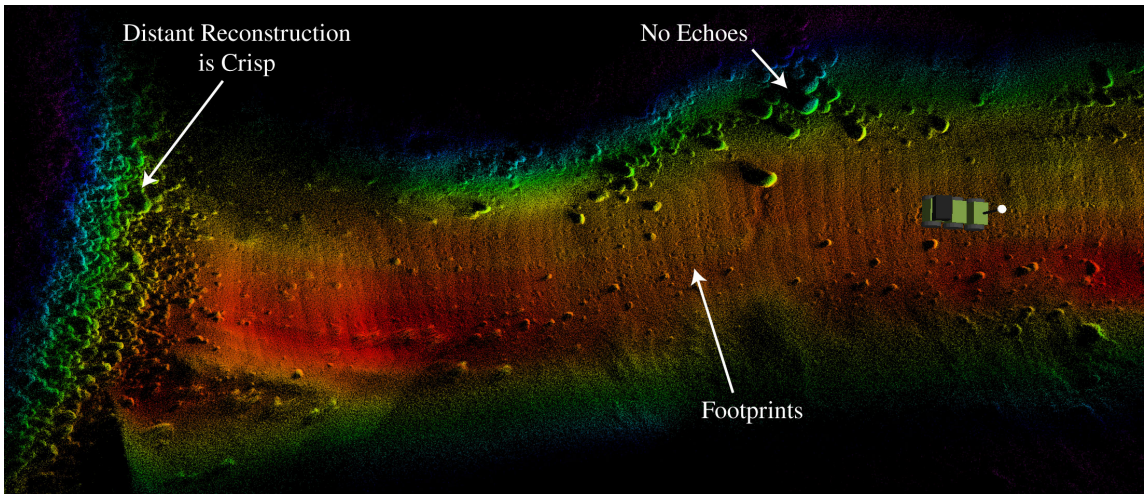
the *Relative CT* method was implemented in Matlab and the *GP-Traj-Sparse* method was implemented in C++, we note that the difference was substantial. In contrast to the *Relative CT* algorithm, which required 1.6 hours of computation to process the 53 minute dataset, our *GP-Traj-Sparse* method took a mere 15.6 minutes; coding language aside, we note that this large speed up can be primarily attributed to the increased problem sparsity and lack of burdensome numerical integration. The *active window size* used in each algorithm was roughly the same (three seconds); using the relative pose-graph, our GP approach performs a breadth-first search of depth *six* (where the keytimes are spaced roughly half-seconds apart). Breaking the timing results down, the *Relative CT* algorithm processed an average window in ~0.84 seconds, while the *GP-Traj-Sparse* method took only ~0.14 seconds (which is fast enough to use the 2 Hz lidar scans *online*). For context, we note that a typical window of *six* active frames (and another ~10 static frames) contains ~2000 'visual' measurements of ~700 active landmarks. Furthermore, the pose at each measurement time is individually interpolated and has a unique Jacobian ($\in \mathbb{R}^{6\times(12+12)}$) that relates it to the adjacent keyframe poses and velocities. In comparison, the computational cost of the GP smoothing factors is almost negligible.

Local Mapping Quality

Although it is common practice to validate SLAM algorithms using long trajectory plots and error metrics based on (rough) GPS ground-truth, we note that the resolution of these (typically) translation-only comparisons do not always indicate the usability of the



(a) Before Motion Compensatation



(b) After Motion Compensatation

Figure 6.16: This figure depicts the effect of motion compensation for a sequence of lidar scans acquired during continuous locomotion. In particular, we note that without compensating for motion, a point cloud reprojection is susceptible to close objects being 'echoed', while the distant reprojection is almost unusable. In contrast, by accounting for the temporal nature of the data we are able to produce a very clean and consistent map.

localization and mapping for navigation purposes. Realistically, successful autonomous systems do not depend on open-loop estimation for hundreds of meters at a time. In order to gain intuition about an estimator's ability to be used for obstacle detection and safe, *local* navigation, it is interesting to inspect a dense (human-readable) map.

The first qualitative result that we wish to share about our continuous-time algorithm is on the effects of motion compensation. In particular, we demonstrate the difference in point-cloud construction, with and without using the continuous-time solution to properly

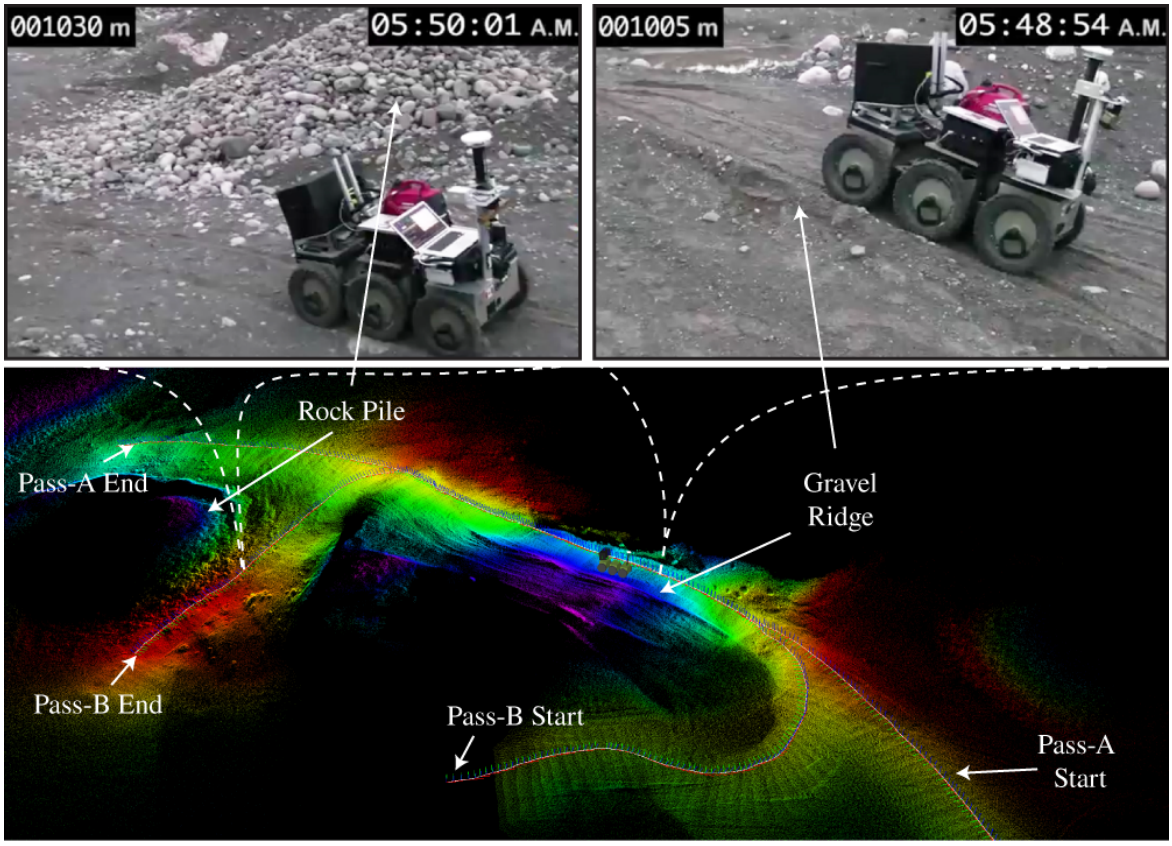


Figure 6.17: This figure shows reprojected dense lidar data, coloured by elevation, from a local region of our estimate that included loop closure. Furthermore, photographs of the experiment provide context for the types of unstructured terrain we traversed with a scanning lidar sensor.

re-project lidar measurements from the spatiotemporal pose they were acquired at (see Figure 6.16). Notably, without motion compensation our map is susceptible to 'echoed' ground planes and obstacles, as well as 'blurry' reprojections at distances greater than ~5-10 meters. In contrast, our GP approach is able to produce a very clean point cloud.

The other qualitative result that we wish to share is a locally consistent point cloud generated by reprojecting dense lidar data from an area involving loop closure (see Figure 6.17). Notably, the reprojection uses lidar data from both passes of the area and does not display any visible inconsistency.

6.5 Discussion

The GP-approach to trajectory estimation described in this chapter has provided a theoretically pleasing batch optimization framework that is probabilistically founded on continuous-time motion priors generated from stochastic models. Furthermore, it is clear

Table 6.3: This table summarizes our sparsity conclusions with respect to the parameterization of the robot state (while using a white-noise-on-acceleration prior). Concerning the standard *arrowhead matrix*, $\mathbf{G}^T\mathbf{R}^{-1}\mathbf{G}$, we report sparsity for the top-left corner (related to pose).

State Parameterization	Sparsity of $\check{\mathbf{P}}$	Sparsity of $\mathbf{G}^T\mathbf{R}^{-1}\mathbf{G}$,
pose only, $\mathbf{T}(t)$	dense	block-diagonal
pose and velocity, $\{\mathbf{T}(t), \boldsymbol{\varpi}(t)\}$	block-tridiagonal	block-diagonal
velocity only, $\boldsymbol{\varpi}(t)$	block-tridiagonal	dense (over integrated times)

(from a theoretical standpoint) that an *accurately* modelled GP prior eliminates the need for ad hoc interpolation schemes or parametric models (such as B-splines). However, while the purely nonparametric approach is both fast and effective for STEAM problems with a small number of measurement times (or a small number of landmarks), we note that current computing technology limits us from applying it to many of the interesting robotic problems that we wish to solve (e.g., sweeping, motion-distorted lidar). In order to handle high-rate or sweeping-type sensors in an *online* algorithm (with current technology), some form of parametric model is required (such as the Hermite splines that arise from our GP interpolation scheme). Nevertheless, the *exactly sparse* GP priors derived in this thesis can be applied to many existing computationally tractable algorithms, such as *keyframed* VO, and can be used to query the solution at other times of interest (post optimization).

Looking back on our GP formulation for $SE(3)$, we note that the (desirable) nonlinear SDE in (6.67) has a very simple linear form that depends on the state parameterization. By marginalizing the pose variables and estimating only velocity (i.e., $\mathbf{x}(t) = \boldsymbol{\varpi}(t)$), the second half of the nonlinear SDE, $\dot{\boldsymbol{\varpi}}(t) = \mathbf{w}(t) \in \mathbb{R}^6$, becomes a LTI SDE for the same problem. In fact, this is the *exact* prior model used by our parametric relative continuous-time method in Chapter 5. However, we note that there is a cost associated with marginalizing the pose variables: any pose-based observation (such as a visual measurement) requires numerical integration of the velocity. Furthermore, the Jacobians of the pose-based measurements also become dense over the length of velocity that must be integrated. This finding concerning sparsity is similar in nature to the observation of Barfoot et al. (2014), who found that marginalizing velocity variables caused an acceleration-based prior to become dense over the pose variables. Table 6.3 summarizes these conclusions; note that the most problem-sparsity is exploited by our GP-based approach, which estimates both the position and velocity variables.

6.6 Summary and Conclusions

In this chapter we have explored extensions to the nonparametric, Gaussian-process-regression approach to batch continuous-time trajectory estimation. The approach was originally proposed by Tong et al. (2012) and later matured by Barfoot et al. (2014) to generate *exactly sparse* GP priors from *linear* time-varying SDEs. Within this field of research, the contributions of this thesis are split into roughly two sections: (i) a linearization methodology to generate *exactly sparse* GP priors from *nonlinear* time-varying SDEs (for trajectories $\in \mathbb{R}^N$), and (ii) the application of this class of *exactly sparse* GP priors to trajectories that belong to the matrix Lie group $SE(3)$.

In the first portion of this chapter, we extended the work of Barfoot et al. (2014) by showing how *nonlinear* time-varying SDEs can be used to generate an *exactly sparse* GP prior. This is accomplished by linearizing the stochastic motion model about a *continuous-time operating point* and then leveraging the existing linear procedures. Furthermore, we perform an analysis of the complexities of this approach and explain how it can be implemented in practice. We also show how both linear and nonlinear *exactly sparse* GP priors can be exploited to improve the computational performance of data-driven hyperparameter training. Both our linearization methodology and data-driven hyperparameter training are validated on a 2D indoor experimental dataset with accurate ground-truth. These methods and results have previously appeared in Anderson et al. (2015a).

In summary, the contributions of this early work are:

1. A methodology to use NTV SDEs, driven by white noise, to generate an *exactly sparse* GP prior for continuous-time trajectories that belong to a vectorspace.
2. An analysis of the complexities that arise from needing a continuous-time linearization point. Specifically, we identify a circular dependence in the linearization, propose a remedy that exploits the iterative nature of the estimation scheme, and determine the minimal storage requirements.
3. An analysis of how data-driven hyperparameter training can also exploit the *exactly sparse* nature of our GP priors.
4. An evaluation of both the nonlinear GP prior, formed from a NTV SDE, and the data-driven hyperparameter training for a 2D mobile robot dataset.

In the later portion of the chapter, we investigate how the Markov property can be exploited to make the GP-regression approach to STEAM efficient for bodies that are

rotating and translating in 3D space. We first analyze how our linearization methodology for vectorspaces can be extended to $SE(3)$, for a (ideal) nonlinear prior, and then demonstrated how a sequence of *local*, LTI SDEs can be used to form a close approximation. Using this piecewise, linear GP prior, we propose a STEAM estimator for $SE(3)$ that is fast, singularity-free, and physically motivated. Furthermore, we demonstrate practical extensions that allow us to use this technique with high-rate, motion-distorted lidar and perform constant-time loop closure on a *relative* pose-graph. The methods are verified experimentally for two different STEAM problems. Note that our exactly sparse GP prior for $SE(3)$, along with the stereo camera results, has previously appeared in Anderson and Barfoot (2015); however, the *relative* extension and appearance-based lidar results have not been previously published.

In summary, the contributions of this later work are:

1. A theoretical extension of our previous linearization methodology to generate an exactly sparse GP prior from a NTV SDE that belongs to $SE(3)$.
2. A piecewise GP prior that uses a sequence of *local* LTI SDEs to closely approximate the desired NTV SDE (when velocity is near constant); the resultant GP-based estimator is fast, singularity-free, and can be used to query the mean and covariance of the posterior estimate at any time.
3. A description of the methodology we use to extend our GP estimator to the *relative* coordinate formulation of STEAM.
4. An evaluation of our estimator's accuracy, consistency, computational performance, and ability to *continuously* interpolate both the mean and covariance functions, using a stereo camera dataset.
5. An evaluation using lidar-style VO and place recognition over a 1.1 km traverse of unstructured, three-dimensional terrain.

Notably, owing to the interesting parallel between the GP-regression approach to STEAM and the standard discrete-time approach to SLAM, it is simple to apply our work to most existing discrete-time batch SLAM implementations. Minimally, we note that it is trivial to incorporate our three-dimensional GP prior factors into an existing batch SLAM solution. Furthermore, this allows for the discrete-time posterior to be queried in a continuous-time fashion; this may be useful for estimating a trajectory with a camera and then reprojecting lidar data post optimization.

Chapter 7

Conclusion

In this chapter, we provide a short summary of the contributions and publications that arose from the work presented in this thesis. Furthermore, we provide some narrative on interesting directions for future work in the area of continuous-time trajectory estimation.

7.1 Summary of Contributions

Our motivation to work on batch continuous-time trajectory estimation originated from the desire to use appearance-based lidar as a passive camera substitute (as described in Chapter 2). Specifically, the original aim of this thesis was to extend the *relative* coordinate formulation of batch discrete-time SLAM (as described in Chapter 3) to estimate a continuous-time trajectory; we postulated that this would allow us to perform incremental SLAM with high-rate asynchronous sensors, such as lidar – even at loop closure. Along this journey we have learned much about batch continuous-time trajectory estimation, made several contributions and publications towards the use of appearance-based lidar for *online* STEAM, and what we hope are both theoretical and practical contributions to the field of trajectory estimation for robotics.

The methods and experiments presented in Chapter 4, where we proposed a novel RANSAC variant for motion-distorted sensors, first appeared in the publication:

- Anderson, S. and Barfoot, T. D. (2013a). RANSAC for motion-distorted 3D visual sensors. In Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan.

The results were then combined with those of Chapter 5 and published in the article:

- Anderson, S., MacTavish, K., and Barfoot, T. D. (2015b). Relative continuous-time SLAM. International Journal of Robotics Research, 34(12):1453-1479.

In summary, the contributions of Chapter 4 are:

1. A temporally-sensitive adaptation of the vision-based RANSAC algorithm, using a constant-velocity model to account for motion-distortion in 3D data.
2. A heuristic extension of the motion-compensated algorithm that improves computational performance without any large sacrifice to quality.
3. Derivation of the conditions for which our constant-velocity model can be solved.
4. An experimental validation of the algorithm, using SURF features extracted from a 6880 frame sequence of lidar intensity imagery.

The methods and experiments presented in Chapter 5, where we explored *relative*, parametric continuous-time estimation, have appeared in two previous publications:

- Anderson, S. and Barfoot, T. D. (2013). Towards relative continuous-time SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany.
- Anderson, S., MacTavish, K., and Barfoot, T. D. (2015b). Relative continuous-time SLAM. International Journal of Robotics Research, 34(12):1453-1479.

In summary, the contributions of Chapter 5 are:

1. A continuous-time adaptation of the relative-coordinate formulation of SLAM. We demonstrate that by estimating the velocity profile of the robot, we can devise an incremental SLAM solution that runs in constant time – even at loop closure.
2. An explanation of how to perform sliding-window-style estimation with basis functions that have a local support greater than two.
3. An evaluation of the algorithm using lidar-style VO and appearance-based place recognition over a 1.1 kilometer traverse of unstructured, three-dimensional terrain.

Most of the methods and experiments presented in Chapter 6, where we explored both nonlinear and 3D extensions to a class of *exactly sparse* GP priors, have appeared in two previous publications. First, the work of Barfoot et al. (2014) was extended by our work on NTV SDEs and hyperparameter training for publication in a journal article:

- Anderson, S., Barfoot, T. D., Tong, C. H., and Sarkka, S. (2015a). Batch nonlinear continuous-time trajectory estimation as exactly sparse gaussian process regression. Autonomous Robots, special issue on Robotics Science and Systems, 39(3):221-238.

Second, our exactly sparse GP prior for $SE(3)$ was published in the proceedings of a full-paper refereed conference, along with the stereo camera results:

- Anderson, S. and Barfoot, T. D. (2015). Full STEAM Ahead: Exactly sparse gaussian process regression for batch continuous-time trajectory estimation on SE(3). In Proceedings of the International Conference on Intelligent Robots and Systems (IROS), pages 157-164, Hamburg, Germany.

Note that the *relative* extension and appearance-based lidar results have not been previously published. In summary, the contributions of Chapter 6 are:

1. A methodology to use NTV SDEs, driven by white noise, to generate an *exactly sparse* GP prior for continuous-time trajectories that belong to a vectorspace.
2. An analysis of the complexities that arise from needing a continuous-time linearization point and a description of the methods that allow us to implement a STEAM estimator based on the nonlinear GP priors.
3. An analysis of how data-driven hyperparameter training can also exploit the *exactly sparse* nature of our GP priors.
4. An evaluation of both the nonlinear GP prior, formed from a NTV SDE, and the data-driven hyperparameter training for a 2D mobile robot dataset.
5. A theoretical extension of our linearization methodology to generate an exactly sparse GP prior from a NTV SDE that belongs to $SE(3)$.
6. A piecewise GP prior that uses a sequence of *local* LTI SDEs to closely approximate the desired NTV SDE (when velocity is near constant).
7. A description of the methodology we use to extend our GP estimator to the *relative* coordinate formulation of STEAM.
8. An evaluation of our estimator's accuracy, consistency, computational performance, and ability to *continuously* interpolate both the mean and covariance functions, using a stereo camera dataset.
9. An evaluation of our estimator's ability to perform lidar-style VO using a 1.1 km traverse of unstructured, three-dimensional terrain.

7.2 Future Work

In this section we discuss possible avenues of future work. The suggestions are partitioned into those that concern the parametric and nonparametric approaches to STEAM.

The biggest challenge for the parametric approach to trajectory estimation is choosing a *knot* spacing that will reliably fit the true motion of the vehicle. Although we propose

that a more realistic motion prior can help alleviate the chance of overfitting, we note that *knot* spacing is a more intuitive and predictable method for an engineer to tune the smoothness of a trajectory curve. Along this line of thought, we make the following suggestions for possible extensions to the parametric approach:

1. The work of Oth et al. (2013) proposes using a *normalized innovation squared* (NIS) test to determine when higher knot resolution is required. This avenue is very promising, but it is not clear that a naive implementation of this approach will be robust to outliers. Further work is required in this area.
2. An alternative approach is to try and use *a priori* information to choose a knot spacing. For example, one could characterize the behaviour of a specific vehicle and then choose the knot spacing based on the commanded control inputs.

Our suggestions for the nonparametric approach are primarily comprised of improvements and extensions to the GP prior model:

1. The most straightforward extension of our 3D approach would be to consider other kinematic models that might be more suitable for non-ground vehicles; for example, one might expect the primary disturbances on quadrotor control to be more prevalent in an inertial frame, rather than the body-centric frame.
2. Reintroducing the control inputs into our piecewise GP prior could be beneficial. However, we note that wheel slip and other unanticipated forces prevent us from trusting the control inputs too heavily. An interesting exercise would be to model the smoothing term, $\mathbf{Q}_C$, as a time-varying function that takes the control inputs as a parameter (e.g., when the control inputs are zero, we expect almost no acceleration).
3. In general, it would be interesting to exploit as much a priori knowledge in the GP prior as possible. For example, dynamical models can be used by including vehicle mass, the gravity vector, motor torques, and even terrain interactions. The surface upon which a ground vehicle drives is a relatively unexploited prior in three-dimensions. Given a model of the terrain (e.g., using lidar scans) one could include cost terms that penalize the pose estimate if it tries to 'float away'.

These avenues of future work represent some of the next steps in maturing the use of continuous-time trajectory representations for robotic localization. It is my belief that this field of research is very promising and it is my hope that these types of techniques become more widely adopted in the many fields of robotic state estimation.

Appendix A

Appearance-Based Lidar Dataset

In order to generate repeatable experimental results for this thesis, the appearance-based lidar experiments conducted by McManus et al. (2012) were packaged into the publicly available[1] *Gravel Pit Lidar-Intensity Imagery Dataset* (Anderson et al., 2012). The general pipeline used to process lidar data in these experiments is reviewed in Chapter 2. The experiments used a ROC6 field robot, equipped with an Autonosys LVC0702 lidar, to capture intensity, range, azimuth, elevation, and timestamp images, with a resolution of 480×360, at 2Hz; the hardware configuration is presented in Figure 2.1(a) and typical intensity/range images are shown in Figure 2.2. The purpose of this appendix is to provide additional detail on the data products, models, and post-processed results used by the experimental results in this thesis.

[1]The dataset is available at `http://asrl.utias.utoronto.ca/datasets/abl-sudbury/`, along with detailed descriptions of the various post-processed data products and coordinate frames.

Figure A.1: The Ethier Sand and Gravel pit in Sudbury, Ontario, Canada.

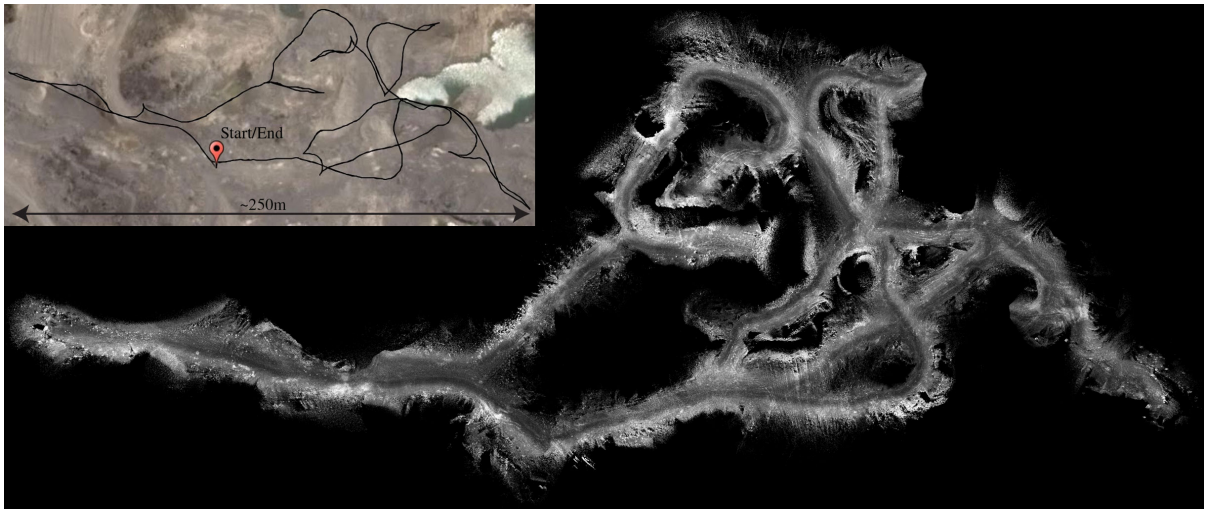


Figure A.2: Point cloud coloured by lidar intensity data for the entire 1.1km traversal. Lidar measurements were reprojected using the trajectory estimation results from Chapter 6. The inset shows satellite imagery and ground-truth DGPS tracks for the traversal.

A.1 Gravel Pit Traversal

The appearance-based lidar experiments presented in this thesis make use of data acquired during a one hour and roughly 1.1km traversal from a VT&R experiment in Sudbury, Ontario, Canada (see Figures A.1 and A.2). During the traversal, the robot traveled between 0 and 0.5 m/s and captured 6880 lidar *imagestacks*. Ground truth is provided by a Thales DG-16 Differential GPS unit, with a Circular Error Probability (CEP) of 0.4m (with 95% of measurements occurring within 0.9m).

All intermediate data products in our VO pipeline have been stored to make our estimation results completely repeatable and enable fair comparisons between all algorithms; arguably, the most important being the data associations provided by RANSAC (since it is *not* a deterministic algorithm). Specifically, we store: (i) the raw lidar *imagestacks*, (ii) post-processed 8-bit intensity images, (iii) DGPS measurements from the Thales DG-16, (iv) SURF extracted from the 8-bit intensity imagery, (v) azimuth, elevation, range, and time measurements interpolated for each SURF feature, (vi) initial temporal matches determined using SURF feature descriptor similarity, (vii) outlier rejection on the temporal data associations (provided by the RANSAC algorithm described in Chapter 4), and (viii) place-recognition results provided by MacTavish and Barfoot (2014).

A.2 Measurement Model

Owing to the nature of the ray-like measurements (i.e. azimuth, elevation, and range), a spherical observation model was used for the vision-based algorithms in Chapters 4, 5, and 6. Noting the observation-model linearization performed in Chapter 3 – equation (3.63) – we define the following spherical model for our lidar SURF measurements:

$$\mathbf{k}(\boldsymbol{p}_s^{\ell,s}) := \begin{bmatrix} \operatorname{atan2}(\varepsilon_2, \sqrt{\varepsilon_1^2+\varepsilon_3^2}) \\ \operatorname{atan2}(\varepsilon_3, \varepsilon_1) \\ \frac{\sqrt{\varepsilon_1^2+\varepsilon_2^2+\varepsilon_3^2}}{\eta} \end{bmatrix}, \quad \boldsymbol{p}_s^{\ell,s} = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}, \tag{A.1}$$

where $\boldsymbol{p}_s^{\ell,s}$ is the homogeneous coordinate for the position of point ℓ, expressed in sensor frame $\underrightarrow{\mathcal{F}}_s$, and $\mathbf{k}(\cdot)$ is the nonlinear sensor model that transforms the point into the spherical sensor space. The observation model Jacobian, $\mathbf{K}_{kj}$, needed to complete the linearization is therefore

$$\mathbf{K}_{kj} := \begin{bmatrix} -\frac{\varepsilon_1\,\varepsilon_2}{\left(\varepsilon_1^2+\varepsilon_2^2+\varepsilon_3^2\right)\sqrt{\varepsilon_1^2+\varepsilon_3^2}} & \frac{\sqrt{\varepsilon_1^2+\varepsilon_3^2}}{\varepsilon_1^2+\varepsilon_2^2+\varepsilon_3^2} & -\frac{\varepsilon_2\,\varepsilon_3}{\left(\varepsilon_1^2+\varepsilon_2^2+\varepsilon_3^2\right)\sqrt{\varepsilon_1^2+\varepsilon_3^2}} & 0 \\ -\frac{\varepsilon_3}{\varepsilon_1^2+\varepsilon_2^2} & 0 & \frac{\varepsilon_1}{\varepsilon_1^2+\varepsilon_2^2} & 0 \\ \frac{\varepsilon_1}{\eta\sqrt{\varepsilon_1^2+\varepsilon_2^2+\varepsilon_3^2}} & \frac{\varepsilon_2}{\eta\sqrt{\varepsilon_1^2+\varepsilon_2^2+\varepsilon_3^2}} & \frac{\varepsilon_3}{\eta\sqrt{\varepsilon_1^2+\varepsilon_2^2+\varepsilon_3^2}} & -\frac{\sqrt{\varepsilon_1^2+\varepsilon_2^2+\varepsilon_3^2}}{\eta^2} \end{bmatrix}. \tag{A.2}$$

Lastly, we note that in order to use a simple spherical model for our measurements, the raw azimuth, elevation, and range measurements, $\mathbf{y}'_{kj}$, require an intrinsic calibration due to the true nature of the sensor (see Figure 2.1(b)). This is accomplished by using the generalized distortion model presented by Dong et al. (2013). The augmented (i.e. undistorted) spherical measurement, $\mathbf{y}_{kj}$, is found using the model:

$$\mathbf{y}_{kj} = \mathbf{y}'_{kj} + \boldsymbol{\Psi}(\alpha', \beta')\mathbf{c}, \quad \mathbf{y}'_{kj} = \begin{bmatrix} \alpha' & \beta' & r' \end{bmatrix}^T, \tag{A.3}$$

where $\boldsymbol{\Psi}(\alpha', \beta')$ is a set of basis functions (e.g. grid-based bilinear interpolation) that takes the raw azimuth and elevation as inputs, and $\mathbf{c}$ is a set of pre-optimized weighting coefficients that determine the additive correction.

A.3 Place Recognition

Loop-closure proposals were generated for the Chapter 5 and 6 experiments by using the place-recognition algorithm proposed by MacTavish and Barfoot (2014). Based on FAB-MAP (Cummins and Newman, 2008), the augmented algorithm describes and

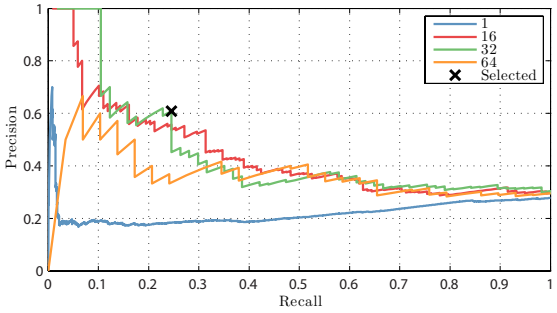


Figure A.3: This figure shows the precision-recall curve of place recognition for different image-group sizes. Note the significant improvement of group matching over the single-image representation. The selected group size of 32 and threshold of $p \geq 0.85$ resulted in 61% precision and 25% recall.

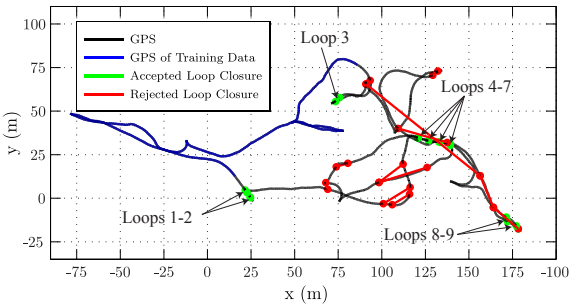


Figure A.4: This figure shows the loop closure proposals overlaid onto the GPS data. The nine accepted loop closures are shown in green, while the rejected loop closures are shown in red. Note that the blue trajectory segment was used for training, and was not evaluated for loop closures.

compares groups of images; loop closures are proposed by computing the likelihood that a query intensity image (or group of images) came from each previously visited location. Due to the (poor) quality of features extracted from our motion-distorted lidar intensity imagery we found that this modification was necessary to get reasonable place-recognition performance (see Figure A.3).

In order to geometrically validate the proposed loop closures, the SURF feature measurements from the blocks of 32 images are undistorted into a privileged coordinate frame (using the continuous-time trajectory solutions from Chapters 5 and 6). A standard RANSAC algorithm is then run to find a transformation between the two temporal segments that validates the loop closure; of the proposed 21 matches, seen in Figure A.4, the nine shown in green were accepted (in both STEAM experiments).

Bibliography

Absil, P.-A., Mahony, R., and Sepulchre, R. (2009). *Optimization algorithms on matrix manifolds*. Princeton University Press.

Ait-Aider, O., Andreff, N., Lavest, J., and Martinet, P. (2006). Simultaneous object pose and velocity computation using a single view from a rolling shutter camera. *Computer Vision–ECCV 2006*, pages 56–68.

Ait-Aider, O. and Berry, F. (2009). Structure and kinematics triangulation with a rolling shutter stereo rig. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1835–1840. IEEE.

Anderson, S. and Barfoot, T. D. (2013a). Ransac for motion-distorted 3d visual sensors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan.

Anderson, S. and Barfoot, T. D. (2013b). Towards relative continuous-time slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany.

Anderson, S. and Barfoot, T. D. (2015). Full steam ahead: Exactly sparse gaussian process regression for batch continuous-time trajectory estimation on se(3). In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 157–164, Hamburg, Germany.

Anderson, S., Barfoot, T. D., Tong, C. H., and Särkkä, S. (2015a). Batch nonlinear continuous-time trajectory estimation as exactly sparse gaussian process regression. *Autonomous Robots,* special issue on "Robotics Science and Systems", 39(3):221–238.

Anderson, S., Dellaert, F., and Barfoot, T. D. (2014). A hierarchical wavelet decomposition for continuous-time slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China.

Anderson, S., MacTavish, K., and Barfoot, T. D. (2015b). Relative continuous-time slam. *International Journal of Robotics Research*, 34(12):1453–1479.

Anderson, S., McManus, C., Dong, H., Beerepoot, E., and Barfoot, T. D. (2012). The gravel pit lidar-intensity imagery dataset. Technical Report ASRL-2012-ABL001, UTIAS.

Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (5):698–700.

Barfoot, T. D. (2016). *State Estimation for Robotics: A Matrix-Lie-Group Approach*. Draft in preparation for publication by Cambridge University Press.

Barfoot, T. D. and Furgale, P. T. (2014). Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Transactions on Robotics*.

Barfoot, T. D., McManus, C., Anderson, S., Dong, H., Beerepoot, E., Tong, C. H., Furgale, P., Gammell, J. D., and Enright, J. (2013). Into darkness: Visual navigation based on a lidar-intensity-image pipeline. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, Singapore.

Barfoot, T. D., Tong, C. H., and Särkkä, S. (2014). Batch continuous-time trajectory estimation as exactly sparse gaussian process regression. In *Proceedings of Robotics: Science and Systems (RSS)*, Berkeley, USA.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. *Computer Vision–ECCV 2006*, pages 404–417.

Bell, B. M. (1994). The iterated kalman smoother as a Gauss-Newton method. *SIAM Journal on Optimization*, 4(3):626–636.

Berczi, L.-P., Posner, I., and Barfoot, T. D. (2015). Learning to assess terrain from human demonstration using an introspective gaussian-process classifier. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3178–3185, Seattle, Washington.

Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics.

Bibby, C. and Reid, I. (2010). A hybrid slam representation for dynamic marine environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 257–264. IEEE.

Bosse, M. and Zlot, R. (2009). Continuous 3d scan-matching with a spinning 2d laser. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4312–4319. IEEE.

Bosse, M., Zlot, R., and Flick, P. (2012). Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping. *IEEE Transactions on Robotics*, 28(5):1104–1119.

Brown, D. C. (1958). A solution to the general problem of multiple station analytical stereotriangulation. RCA-MTP data reduction tech. report no. 43, Patrick Airforce Base.

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. *Computer Vision–ECCV 2010*, pages 778–792.

Chirikjian, G. S. (2009). *Stochastic Models, Information Theory, and Lie Groups: Classical Results and Geometric Methods*, volume 1-2. Birkhauser, New York.

Chirikjian, G. S. and Kyatkin, A. B. (2016). *Harmonic Analysis for Engineers and Applied Scientists: Updated and Expanded Edition*. Courier Dover Publications.

Chli, M. and Davison, A. J. (2008). Active matching. In *Computer Vision–ECCV 2008*, pages 72–85. Springer.

Churchill, W. and Newman, P. (2012). Practice makes perfect? managing and leveraging visual experiences for lifelong navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4525–4532. IEEE.

Churchill, W. and Newman, P. (2013). Experience-based Navigation for Long-term Localisation. *The International Journal of Robotics Research (IJRR)*.

Cummins, M. and Newman, P. (2008). Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665.

Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *IEEE T. PAMI*, 29(6):1052–1067.

Deisenroth, M. P., Turner, R., Huber, M., Hanebeck, U. D., and Rasmussen, C. E. (2012). Robust filtering and smoothing with Gaussian processes. *IEEE T. Automatic Control*, 57:1865–1871.

Dellaert, F. and Kaess, M. (2006). Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203.

Dong, H., Anderson, S., and Barfoot, T. D. (2013). Two-axis scanning lidar geometric calibration using intensity imagery and distortion mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany.

Dong, H. and Barfoot, T. (2011). Pose interpolation for bundle adjustment.

Dong, H. and Barfoot, T. D. (2012). Lighting-invariant visual odometry using lidar intensity imagery and pose interpolation. In *Proceedings of the International Conference on Field and Service Robotics (FSR)*, Matsushima, Japan.

Ferris, B., Fox, D., and Lawrence, N. (2007). Wifi-SLAM using Gaussian process latent variable models. In *Proc. IJCAI*.

Fischler, M. and Bolles, R. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Fleps, M., Mair, E., Ruepp, O., Suppa, M., and Burschka, D. (2011). Optimization based imu camera calibration. In *In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3297–3304.

Furgale, P. T. and Barfoot, T. D. (2010). Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics,* special issue on "Visual mapping and navigation outdoors", 27(5):534–560.

Furgale, P. T., Barfoot, T. D., and Sibley, G. (2012). Continuous-time batch estimation using temporal basis functions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2088–2095, St. Paul, USA.

Furgale, P. T., Tong, C. H., Barfoot, T. D., and Sibley, G. (2015). Continuous-time batch trajectory estimation using temporal basis functions. *International Journal of Robotics Research.*

Google Inc. (2015). Google self-driving car project monthly report - October 2015. Technical report, Google Inc.

Grisetti, G., Stachniss, C., Grzonka, S., and Burgard, W. (2007). A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA.

Guizilini, V. and Ramos, F. (2012). Semi-parametric models for visual odometry. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3482–3489. IEEE.

Haralick, B. M., Lee, C.-N., Ottenberg, K., and Nölle, M. (1994). Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13(3):331–356.

Hartley, R. and Zisserman, A. (2000). *Multiple view geometry in computer vision.* Cambridge University Press.

Hedborg, J., Forssén, P., Felsberg, M., and Ringaby, E. (2012). Rolling shutter bundle adjustment. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1434–1441.

Hedborg, J., Ringaby, E., Forssén, P.-E., and Felsberg, M. (2011). Structure and motion estimation from rolling shutter video. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 17–23. IEEE.

Horn, B. K. (1987). Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642.

Huber, P. J. (1981). Robust statistics.

Husmann, K. and Pedersen, L. (2008). Strobe lit high dynamic range stereo imagery for dark navigation. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS).*

Jazwinski, A. H. (1970). *Stochastic Processes and Filtering Theory*. Academic, New York.

Jia, C. and Evans, B. L. (2012). Probabilistic 3-d motion estimation for rolling shutter video rectification from visual and inertial measurements. In *Proc. IEEE Int. Workshop on Multimedia Signal Processing*.

Johnson, A. E., Goldberg, S. B., Cheng, Y., and Matthies, L. H. (2008). Robust and efficient stereo feature tracking for visual odometry. In *IEEE International Conference on Robotics and Automation*, pages 39–46. IEEE.

Julier, S. J. and Uhlmann, J. K. (1997). New extension of the kalman filter to nonlinear systems. In *AeroSense'97*, pages 182–193. International Society for Optics and Photonics.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. *IJRR*, 31(2):217–236.

Kaess, M., Ranganathan, A., and Dellaert, R. (2008). iSAM: Incremental smoothing and mapping. *IEEE TRO*, 24(6):1365–1378.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45.

Kalman, R. E. and Bucy, R. S. (1961). New results in linear filtering and prediction theory. *Transactions of the ASME–Journal of Basic Engineering*, 83(3):95–108.

Kim, M.-J., Kim, M.-S., and Shin, S. Y. (1995). 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*, pages 369–376. ACM.

Ko, J. and Fox, D. (2009). GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90.

Ko, J. and Fox, D. (2011). Learning GP-BayesFilters via Gaussian process latent variable models. *Auton. Robots*, 30(1):3–23.

Konolige, K. (2010). Sparse sparse bundle adjustment. In *Proceedings of the British Machine Vision Conference*, pages 1–11.

Konolige, K., Agrawal, M., and Sola, J. (2007). Large-scale visual odometry for rough terrain. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, pages 201–212.

Konolige, K., Bowman, J., Chen, J., Mihelich, P., Calonder, M., Lepetit, V., and Fua, P. (2010). View-based maps. *The International Journal of Robotics Research*, 29(8):941.

Lawrence, N. (2003). Gaussian process latent variable models for visualization of high dimensional data. In *Proc. NIPS*.

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

Levenberg, K. (1944). A method for the solution of certain non–linear problems in least squares.

Levinson, J. S. (2011). *Automatic laser calibration, mapping, and localization for autonomous vehicles*. Stanford University.

Longuet-Higgins, H. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135.

Lovegrove, S., Patron-Perez, A., and Sibley, G. (2013). Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. *British Machine Vision Conference (BMVC13)*.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Lu, F. and Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Auton. Robots*, 4(4):333–349.

Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence*.

MacTavish, K. and Barfoot, T. D. (2014). Towards hierarchical place recognition for long-term autonomy. In *Proceedings of the IEEE International Conference on Robotics and Automation workshop on "Visual Place Recognition in Changing Environments"*, Hong Kong, China.

Magnusson, M., Lilienthal, A., and Duckett, T. (2007). Scan registration for autonomous mining vehicles using 3d-ndt. *Journal of Field Robotics*, 24(10):803–827.

Maimone, M., Cheng, Y., and Matthies, L. (2007). Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186.

Maimone, M., Johnson, A., Cheng, Y., Willson, R., and Matthies, L. (2006). Autonomous navigation results from the mars exploration rover (mer) mission. *Experimental Robotics IX*, pages 3–13.

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441.

Matthies, L., Maimone, M., Johnson, A., Cheng, Y., Willson, R., Villalpando, C., Goldberg, S., Huertas, A., Stein, A., and Angelova, A. (2007). Computer vision on mars. *International Journal of Computer Vision*.

Matthies, L. and Shafer, S. (1987). Error modeling in stereo navigation. *IEEE Journal of Robotics and Automation, 3(3)*.

Matthies, L. H. (1989). *Dynamic Stereo Vision*. PhD thesis, Pittsburgh, PA, USA. AAI9023429.

McManus, C., Furgale, P. T., and Barfoot, T. D. (2011). Towards appearance-based methods for lidar sensors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1930–1935, Shanghai, China.

McManus, C., Furgale, P. T., and Barfoot, T. D. (2013). Towards lighting-invariant visual navigation: An appearance-based approach using scanning laser-rangefinders. *Robotics and Autonomous Systems*, 61:836–852.

McManus, C., Furgale, P. T., Stenning, B. E., and Barfoot, T. D. (2012). Visual teach and repeat using appearance-based lidar. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 389–396, St. Paul, USA.

Moravec, H. (1980). Obstacle avoidance and navigation in the real world by a seeing robot rover. In *tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University and doctoral dissertation, Stanford University*, number CMU-RI-TR-80-03. Available as Stanford AIM-340, CS-80-813 and republished as a Carnegie Mellon University Robotics Institue Technical Report to increase availability.

Murray, R. M., Li, Z., and Sastry, S. (1994). A mathematical introduction to robotic manipulation. *Boca Raton, FL: CRC*.

Neira, J., Tardos, J. D., Horn, J., and Schmidt, G. (1999). Fusing range and intensity images for mobile robot localization. *IEEE Transactions on Robotics and Automation*, 15:76–84.

Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(6):756–770.

Nüchter, A., Lingemann, K., Hertzberg, J., and Surmann, H. (2007). 6d slam-3d mapping outdoor environments. *Journal of Field Robotics*, 24(8-9):699–722.

Olson, E., Leonard, J., and Teller, S. (2006). Fast iterative alignment of pose graphs with poor initial estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2262–2269. IEEE.

Oth, L., Furgale, P. T., Kneip, L., and Siegwart, R. (2013). Rolling shutter camera calibration. In *Proc. of The IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Portland, USA.

Pathak, K., Birk, A., Vaskevicius, N., Pfingsthorn, M., Schwertfeger, S., and Poppinga, J. (2010). Online three-dimensional slam by registration of large planar surface segments and closed-form pose-graph relaxation. *Journal of Field Robotics*, 27(1):52–84.

Pomerleau, F., Colas, F., and Siegwart, R. (2015). A Review of Point Cloud Registration Algorithms for Mobile Robotics. *Foundations and Trends in Robotics (FnTROB)*, 4(1):1–104.

Powell, M. J. (1970). A hybrid method for nonlinear equations. *Numerical methods for nonlinear algebraic equations*, 7:87–114.

Ranganathan, P. and Olson, E. (2012). Gaussian process for lens distortion modeling. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3620–3625. IEEE.

Rankin, A. L., Huertas, A., and Matthies, L. H. (2007). Night-time negative obstacle detection for off-road autonomous navigation. In *Defense and Security Symposium*, pages 656103–656103. International Society for Optics and Photonics.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA.

Rosen, D., Huang, G., and Leonard, J. J. (2014). Inference over heterogeneous finite-/infinite-dimensional systems using factor graphs and Gaussian processes. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1261–1268.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006*, pages 430–443. Springer.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *International Conference on Computer Vision*, Barcelona.

Särkkä, S. (2006). *Recursive Bayesian Inference on Stochastic Differential Equations*. PhD thesis, Helsinki Uni. of Technology.

Sibley, G., Matthies, L., and Sukhatme, G. (2008). A sliding window filter for incremental slam. In *Unifying perspectives in computational and robot vision*, pages 103–112. Springer.

Sibley, G., Mei, C., Reid, I., and Newman, P. (2009). Adaptive relative bundle adjustment. In *Proceedings of Robotics: Science and Systems*, Seattle, USA.

Sibley, G., Mei, C., Reid, I., and Newman, P. (2010). Vast-scale outdoor navigation using adaptive relative bundle adjustment. *The International Journal of Robotics Research*, 29(8):958.

Smith, R. C., Self, M., and Cheeseman, P. (1990). Estimating uncertain spatial relationships in robotics. In Cox, I. J. and Wilfong, G. T., editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, New York.

Sommer, H., Forbes, J. R., Siegwart, R., and Furgale, P. (2015). Continuous-time estimation of attitude using b-splines on lie groups. *Journal of Guidance, Control, and Dynamics*, pages 1–20.

Stenning, B. E., McManus, C., and Barfoot, T. D. (2013). Planning using a network of reusable paths: A physical embodiment of a rapidly exploring random tree. *Journal of Field Robotics,* special issue on "Space Robotics", 30(6):916–950.

Strasdat, H., Montiel, J. M. M., and Davison, A. J. (2010). Real-time monocular SLAM: Why filter? In *Proc. ICRA*.

Sünderhauf, N. and Protzel, P. (2012). Switchable constraints for robust pose graph slam. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1879–1884. IEEE.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2001). Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1):99–141.

Thrun, S. and Montemerlo, M. (2006). The graph SLAM algorithm with applications to large-scale mapping of urban structures. *IJRR*, 25(5-6):403–429.

Tong, C., Anderson, S., Dong, H., and Barfoot, T. D. (2014). Pose interpolation for laser-based visual odometry. *Journal of Field Robotics,* special issue on "Field and Service Robotics", 31(5):787–813.

Tong, C. H., Furgale, P., and Barfoot, T. D. (2012). Gaussian process Gauss-Newton: Non-parametric state estimation. In *Proc. of the 9th Conf. on Computer and Robot Vision*, pages 206–213.

Tong, C. H., Furgale, P. T., and Barfoot, T. D. (2013). Gaussian process gauss-newton for non-parametric simultaneous localization and mapping. *International Journal of Robotics Research*, 32(5):507–525.

Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (2000). Bundle adjustment — A modern synthesis. In Triggs, B., Zisserman, A., and Szeliski, R., editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer Berlin Heidelberg.

Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 13(4):376–380.

Vasudevan, S., Ramos, F., Nettleton, E., and Durrant-Whyte, H. (2009). Gaussian process modeling of large-scale terrain. *Journal of Field Robotics*, 26(10):812–840.

Wang, Y. and Chirikjian, G. S. (2006). Error propagation on the euclidean group with applications to manipulator kinematics. *Robotics, IEEE Transactions on*, 22(4):591–602.

Wang, Y. and Chirikjian, G. S. (2008). Nonparametric second-order theory of error propagation on motion groups. *IJRR*, 27(11-12):1258–1273.

Wolfe, K., Mashner, M., and Chirikjian, G. (2011). Bayesian fusion on lie groups. *Journal of Algebraic Statistics*, 2(1):75–97.

Yan, X., Indelman, V., and Boots, B. (2014). Incremental sparse GP regression for continuous-time trajectory estimation and mapping. In *Proceedings of the NIPS Workshop on Autonomously Learning Robots*.

Zlot, R. and Bosse, M. (2012). Efficient large-scale 3d mobile mapping and surface reconstruction of an underground mine. In *Proceedings of the International Conference on Field and Service Robotics (FSR)*, Matsushima, Japan.

Zlot, R. and Bosse, M. (2014). Efficient large-scale three-dimensional mobile mapping for underground mines. *Journal of Field Robotics*.