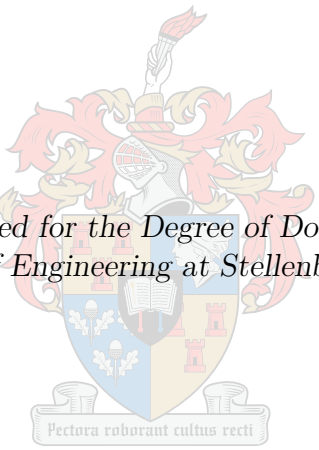
Stochastic Triangular Mesh Mapping

by

Clint Daniel Lombard



Dissertation approved for the Degree of Doctor of Engineering
in the Faculty of Engineering at Stellenbosch University

Promoter: Dr Corné E. van Daalen

April 2020

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged.
Opinions expressed and conclusions arrived at are those of the author and are not necessarily to be attributed to the NRF.

Plagiarism Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Clint Daniel Lombard

Abstract

For a mobile robot to operate autonomously in general environments, the ability of the robot to perceive its surroundings is paramount. To perform this task of perception, a robot must have the ability to incrementally construct a model of its environment—otherwise known as online dense mapping.

The existing dense mapping techniques do not account for all sources of uncertainty in the system, and also neglect to model any structure inherent in the environment. To address these shortcomings, we present the stochastic triangular mesh (STM) mapping technique: a 2.5-D representation of the surface of the environment using a continuous mesh of triangular surface elements, where each surface element models the mean plane and roughness of the underlying surface.

In contrast to existing mapping techniques, an STM map models the structure of the environment by ensuring a continuous model that can be updated incrementally and in an efficient manner—with a linear computational cost in the number of measurements. This efficiency is due to the use of approximate message-passing techniques; specifically, a combination of loopy belief propagation (LBP) and variational message passing (VMP). The uncertainty in the measurements of the environment and robot pose (position and orientation) is accounted for by the use of these Bayesian inference techniques during the map update.

We demonstrate that an STM map can be used with sensors that generate point measurements, such as light detection and ranging (LiDAR) sensors and stereo cameras. Simulated results show that, when comparing the log likelihoods of the models, an STM map is a more accurate model than the only comparable online surface-mapping technique—a standard elevation map—while also being as expressive as the offline Gaussian process (GP) mapping technique. We also provide qualitative results on a large-scale practical dataset.

In addition to presenting the STM mapping technique, we demonstrate that performing dense mapping in the relative inertial reference frame (IRF) of a triangular submap has several advantages over the traditional approach using a single global IRF, and extend the 2-D hybrid metric map (HYMM) framework to three dimensions. We demonstrate that performing dense mapping in a relative IRF addresses issues like loop closure or the kidnapped robot problem, which affect mapping in a global IRF.

Opsomming

Vir ’n mobiele robot om outonoom in algemene omgewings te werk, is die vermoë van die robot om die omgewing waar te neem van groot belang. Om hierdie taak van waarneming te verrig, moet ’n robot die vermoë hê om op ’n inkrementele wyse ’n model van sy omgewing op te stel—dit staan bekend as aanlyn digte kartering.

Die bestaande digte karteringstegnieke neem nie alle bronne van onsekerheid in die stelsel in ag nie, en modelleer ook nie enige struktuur inherent aan die omgewing nie. Om hierdie tekortkominge aan te spreek, stel ons die stochastiese-driehoekige-maas-(STM)-kaarteringstegniek voor: ’n 2.5-D-voorstelling van die oppervlak van die omgewing deur middel van ’n kontinue maas van driehoekige oppervlakelemente, waar elke oppervlakelement die gemiddelde vlak en grofheid van die onderliggende oppervlak modelleer.

In teenstelling met die bestaande karteringstegnieke, modelleer ’n STM-kaart die struktuur van die omgewing deur kontinuïteit af te dwing en dateer die kaart op ’n inkrementele en doeltreffende wyse op—met lineêre koste in die aantal metings. Hierdie doeltreffendheid is as gevolg van die gebruik van benaderde boodskap-oordrag-tegnieke; spesifiek ’n kombinasie van lusvormige kennisvoortplanting (“loopy belief propagation”) en veranderlike boodskap-oordrag (“variational message passing”). Die onsekerheid in die metings van die omgewing en robot-posisie en -oriëntasie word in ag geneem deur die gebruik van hierdie Bayesiese inferensietegnieke tydens die kaartopdatering.

Ons demonstreer dat ’n STM-kaart gebruik kan word met sensors wat puntmetings genereer, byvoorbeeld LiDAR-sensors en stereokameras. Simulasieresultate wys dat ’n STM-kaart ’n meer akkurate model is, wanneer die log-waarskynlikheid vergelyk word, as die enigste vergelykbare aanlyn-oppervlak-karteringstegniek—’n gewone hoogtekaart—terwyl dit ook net so ekspressief is as die aflyn Gaussiese-proses-karteringstegniek. Ons wys ook kwalitatiewe resultate van ’n groot praktiese datastel.

Benewens die voorstel van die STM-karteringstegniek wys ons ook dat kartering in die relatiewe inersiële koördinaatstelsel van ’n driehoekige subkaart verskeie voordele toon bo die tradisionele benadering, wat ’n enkele globale inersiële koordinaatstelsel gebruik, en brei die raamwerk vir 2-D hibriede metrieke kaart uit na drie dimensies. Ons wys dat deur digte kartering in ’n relatiewe koördinaatstelsel te doen, word probleme soos lusluiting of die ontvoerde-robot-probleem aangespreek.

Acknowledgements

During the course of this project, there have been several people who have contributed in myriad ways to both its length and success:

- Amber Bennett, I would like to thank you for your unwavering support and constant motivation, which without, none of this would have been possible.
- Dr Corné van Daalen, my mentor and supervisor, I would like to thank you for teaching me to think critically and always pushing for the very best—as well as enlightening me to the many uses of the humble comma.
- Prof. Johan du Preez, I would like to thank you for your valuable thoughts on my work and allowing the use of the EMDW PGM library.
- My colleagues and friends in the Electronic Systems Laboratory (ESL) who have enjoyed this time with me, the many adventures and coffee-fuelled discussions will always remain fond memories.
- Wessel Croukamp, Wynand van Eeden, and Johan Arendse, thank you for all your help with the numerous mechanical design and manufacturing requests.
- My family, thank you for affording me all opportunities that I have been so privileged to have, and for supporting me even when you didn't understand what I was doing.

The assistance of several organisations towards this project is greatly appreciated. Specifically, I would like to thank the National Research Foundation (NRF), and Armsor for their financial assistance, as well as Tectra Automation for sponsoring the aluminium sensor mounting rig for the RooiBot.

Several vector graphics were used from external sources. The robot icon was adapted from an original by SimpleIcon from Flaticon (Figures 1.1, 1.2 and 3.1a). The hill and tree icons were adapted from originals by sceneit from Vecteezy.com (Figure 4.1).

Contents

Acronyms and Abbreviations

BCM	Bayesian committee machine
BP	belief propagation
BRF	body reference frame
DAG	directed acyclic graph
DGPS	differential global positioning system
EKF	extended Kalman filter
EMDW	elementary, my dear Watson
EP	expectation propagation
GP	Gaussian process
HYMM	hybrid metric map
IID	independently and identically distributed
IMU	inertial measurement unit
IRF	inertial reference frame
KF	Kalman filter
KL	Kullback-Leibler
LBP	loopy belief propagation
LIBELAS	library for efficient large-scale stereo matching
LiDAR	light detection and ranging
LM	Levenberg-Marquardt
MAP	maximum a posteriori
MCMC	Markov chain Monte Carlo
MLS	multi-level surface
MRF	Markov random field
MSE	mean squared error
NDT	normal distributions transform
PGM	probabilistic graphical model
PnP	Perspective-n-Point
pose	position and orientation
ROC	receiver operating characteristic
SDF	signed distance function
SLAM	simultaneous localisation and mapping
SPLAM	simultaneous planning, localisation and mapping
STM	stochastic triangular mesh
surfel	surface element
TSDF	truncated signed distance function
UKF	unscented Kalman filter
UT	unscented transform
VMP	variational message passing
voxel	volumetric element

1 Introduction

Autonomous mobile robots grant us access to environments that would otherwise be impossible—for example disaster management [1] or interplanetary exploration [2], to name a few. They also make parts of our lives easier—a prominent example being the recent revolution in self-driving cars [3]. For a truly autonomous mobile robot to safely navigate within or interact with its environment, several interdependent subsystems are required (Figure 1.1). In particular, an autonomous mobile robot needs the ability to perceive its surroundings. This problem of perception is commonly referred to as *dense mapping* and is the focus of this dissertation.

In the general case of a robot operating in an initially unknown environment, the map needs to be incrementally built online using the measurements of the environment, such that the robot can make decisions in real time. These noisy measurements of spatial features in the environment are obtained using *exteroceptive sensors*—typically a combination of light detection and ranging (LiDAR) sensors, stereo cameras, and depth cameras. In order to combine measurements taken at different locations into a single map, the robot must determine its pose (position and orientation) within the environment; that is, to perform *localisation*. The measurements can also be used to improve localisation by performing simultaneous localisation and mapping (SLAM).

Using the dense map, a complex task like *path planning*—including collision detection and object manipulation—can be performed, whereby actions (control commands) are calculated in line with the desired goal of the robot. These control commands feed into a *robot controller* which translates them into actuator signals. In order to monitor the execution of these commands *proprioceptive sensors* measure the internal state of the robot—for example, linear accelerations and angular velocities are monitored using an inertial measurement unit (IMU).

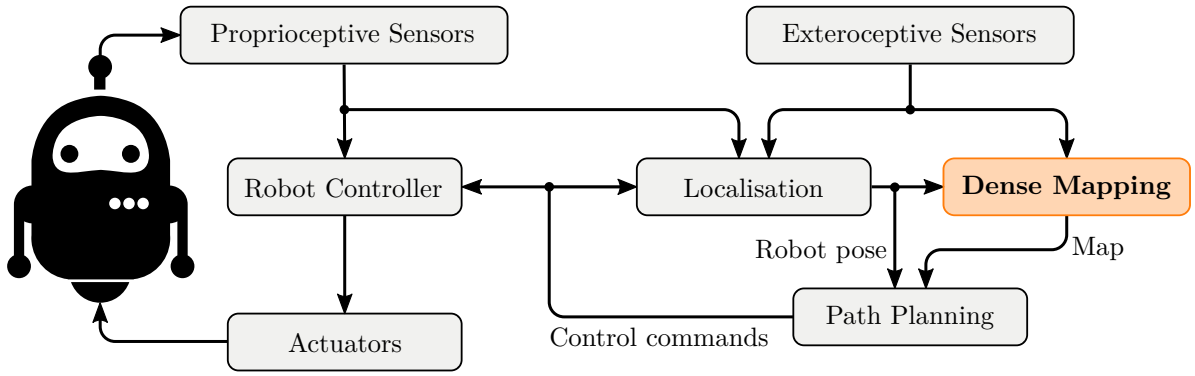


Figure 1.1: A high-level overview of the subsystems required by an autonomous robotic system, where the robot icon refers to the physical robot platform. As highlighted, we focus on dense mapping. Adapted with permission from van Daalen [4].

Dense mapping can more precisely be defined as representing a robot's belief over the environment, where "belief" refers to the knowledge about a state, given all measurements and any prior information. Calculating the dense map belief is not an easy task as both the robot pose (position and orientation) and its observations of the environment are noisy, and this information needs to be fused into a compact and expressive model. Guizilini and Ramos [5] argue that, for a dense map to be an effective representation of the environment, it should have the following attributes:

- *Reason under uncertainty.* Measurements from all robot sensors contain some degree of uncertainty, which needs to be accounted for when updating the map.
- *Incremental updates.* Due to the online nature of the problem, all the measurements are not available at once and, as the robot needs to continuously query the map, the map needs to be updated incrementally.
- *Update and query efficiently.* Exteroceptive sensors generate vast amounts of data—typically in the form of dense point clouds—that need to be incorporated quickly into the map. Additionally, the resulting map needs to be accessed quickly when required.
- *Represent the structure in the environment.* The environment is inherently structured according to spatial relationships on various length scales. By exploiting this structure, a map can interpolate occluded regions, or better estimate regions with few measurements.

In this dissertation, we present stochastic triangular mesh (STM) mapping, an online dense mapping technique designed around these four attributes. In particular, we focus on two aspects:

Robot pose belief A source of uncertainty influencing dense mapping is localisation; to build an accurate model of the environment, one must also consider the belief over the robot pose. Estimating the robot pose belief is typically performed using the process of SLAM, whereby the belief over a sparse map—consisting of re-identifiable landmarks—is estimated in conjunction with the robot pose. The sparsity of this landmark map, however, prohibits it being used for complex planning tasks, and it should therefore rather be considered as a localisation aid. When performing SLAM, the beliefs over the robot pose at any time step can change due to new information, as is also the case when a robot revisits a part of the environment—that is, when loop closure is performed. This can cause significant changes in the beliefs over these previous poses, which present a problem, as measurements obtained at previous poses would have been incorporated into the dense map using the previous pose beliefs. To account for this, measurements would need to be removed and reintegrated into the map using the retroactively updated pose beliefs—an expensive undertaking.

Statistical dependencies An aspect of representing the structure of the environment in a map is to model the continuity of surfaces. However, most dense mapping techniques are made tractable through the assumption that neighbouring map elements are statistically independent (as we will discuss in Section 1.1). This allows each element to be updated independently, but at the expense of modelling continuity. As a consequence of this assumption, the resulting models contain numerous discontinuities. This is problematic for a task like collision prediction, for which it is desirable to have a continuous representation of the environment in order for it to function accurately. The alternative approach of modelling dependencies between map elements will lead to more accurate map beliefs, but at a higher computational cost.

1.1 Existing Approaches to Dense Mapping

To motivate our proposal for a new mapping technique, we now review and evaluate the existing approaches to dense mapping, and specifically consider the desired attributes a dense map should have.

1.1.1 Polygonal Mesh Maps

Polygonal meshes are a well-known method of spatial modelling used in computer graphics. A mesh is constructed from a point cloud by linking points together with edges, forming polygons. In an early application, Thrun et al. [6] incrementally built sections of a triangular mesh of indoor environments, subsequently simplifying areas of the mesh by fitting rectangular planes using an online implementation of expectation maximisation. More recently, Wiemann et al. [7] used a k-nearest-neighbours approach to estimate the normals of a triangular mesh, creating a consistent representation of planar surfaces. The resulting mesh was simplified by fusing local polygons with similar normal vectors; however, this process was calculated in post-processing. Zienkiewicz et al. [8] modelled small-scale environments with a fixed-topology triangular mesh, and formulated the update procedure probabilistically. By performing weighted optimisation, they were able to iteratively and incrementally obtain solutions to the mesh. Although their approach is formatted probabilistically, it does not capture the uncertainty in the model. Polygonal meshes are not designed to explicitly represent any uncertainty, and therefore cannot incorporate the robot pose belief or sensor uncertainty in a principled manner. They are, however, a popular method for visualising other probabilistic techniques.

1.1.2 Occupancy Grid Maps

Occupancy grid mapping is one of the most widely used mapping techniques and is considered the de facto representation for dense mapping. Originally developed in 2-D by Moravec and Elfes [9], this technique discretises the map into a volumetric grid, where each volumetric element (voxel) stores the probability that the associated region is occupied. As with all volumetric representations in 3-D, spanning the mapping space with a grid becomes prohibitively expensive, despite the majority of the mapped space being either unoccupied or unobserved. To address this, Hornung et al. [10] developed the OctoMap framework—this is arguably the most used dense mapping framework. Instead of a fixed voxel size, they used octrees to recursively adjust the voxel resolution. Unimportant information, namely the unoccupied or unobserved space, is compressed into coarse voxels, while a finer resolution is maintained for occupied regions. Improving on this, Einhorn et al. [11] developed an adaptive online method of determining the division depth in local regions of the map using the more general k-d tree. Khan et al. [12] used rectangular voxels to compress unoccupied cubic voxels. In recent work, Droeschel et al. [13] used an allocentric (robot-centric) grid, where the resolution increases with the distance from the robot. Their approach was also implemented practically, and tested at the DARPA Robotics Challenge. In order to handle the uncertainty of the robot pose belief, Joubert et al. [14] incorporated a beam sensor model into occupancy map updates using Monte Carlo integration, although this approach was only demonstrated in 2-D. In an attempt to incorporate statistical dependency between 2-D voxels, Thrun [15] used a forward measurement model and, in an offline post-processing procedure, optimised for the maximum likelihood map.

Despite their popularity, occupancy grid maps fail to incorporate the dependencies between map elements in an online manner.

1.1.3 Elevation Maps

The most general way of representing an environment is to use a 3-D dense map; however, some environments are adequately represented as a 2.5-D map—that is, a single axis[1] is constrained to only one value per map element. These elevation maps reduce the dimensionality of the map by associating a single height to each element of a 2-D grid. Early work by Hebert et al. [16] used an elevation map to perform localisation, as well as to identify footholds for a legged robot. A drawback of elevation maps is their inability to represent overhangs, such as bridges. Triebel et al. [17] dealt with this problem in their method of multi-level surface (MLS) mapping by clustering the measurements in each map element, with the clusters across neighbouring map elements being segmented into elevation classes. Recent work by Fankhauser et al. [18] incorporated the uncertainty in robot pose belief into an allocentric elevation map. They achieved this by maintaining a distribution over the spatial uncertainty of each cell, which is updated over time based on the uncertainty in transformations between the allocentric reference frames. The resultant map is constructed from the weighted average of neighbouring cells, based on each cell's spatial uncertainty. Their approach was demonstrated for a legged robot. In gamma-SLAM, Marks et al. [19] represent the environment using a 2.5-D precision map, where measurements in each map element are modelled as samples from a Gaussian distribution over the elevation, with an unknown mean and precision. By considering only the precision—marginalising out the mean—their method generates accurate maps for localisation, while also providing a pseudo-metric for traversability. However, their model fails to account for the case where the sensor uncertainty varies at different ranges, and the model disregards height.

Elevation maps are an efficient and effective method of representing environments in which a full 3-D representation is unnecessary. However, elevation maps fail to incorporate the statistical dependencies between map elements.

1.1.4 Truncated Signed Distance Function Maps

Over the last decade, advances in depth-camera technology have created affordable and relatively accurate depth sensors. This has given rise to a non-parametric surface representation of the environment using the signed distance function (SDF), commonly used in computer graphics. The SDF calculates the Euclidean distance to the nearest surface, defining positive values to indicate free space and negative values to indicate occluded regions. Consequently, the surface is implicitly described at the zero-crossings. As it is unnecessary to compute distances that are far away from the zero-crossings, the truncated signed distance function (TSDF) truncates distances beyond chosen negative and positive distance thresholds. In KinectFusion, a technique pioneered by Newcombe et al. [20], measurements are fused into a regular 3-D grid of voxels by computing TSDF values. Due to memory constraints, this method is only suitable to medium-scale environments. Whelan et al. [21] expanded this to handle large-scale environments by using a sliding window to maintain a local section of the environment as a TSDF map, while incrementally converting the regions exiting this window into a triangular mesh. A major advantage of this method is its ability to adjust the map upon a loop closure. This is achieved through an optimisation procedure over the sensor poses and vertices of the meshed map—creating a globally consistent map. BundleFusion, a method devised by Dai et al. [22], performs bundle adjustment to optimise over the sensor poses to create accurate TSDF maps. Similarly to KinectFusion, this method suffers from memory constraints due to maintaining the complete map in a grid, and is therefore limited to small-scale environments. The standard TSDF representation is not probabilistic, and consequently neither sensor uncertainty nor pose uncertainty can be incorporated when updating the map. To address this, Dietrich et al. [23] modelled each signed distance as a

[1] Typically the vertical axis.

Gaussian distributed random variable. This approach, however, treats the distribution in each voxel independently. Additionally, this choice of distribution may be unsuitable in cases where there is large uncertainty, as a result of the truncated distances.

Although TSDF methods have been shown to generate highly detailed maps in real time, they have not been used widely in the robotics community due to several drawbacks. Firstly, the TSDF can only implicitly describe the surfaces in the environment; when an explicit surface map is required, the TSDF map needs to be converted to a mesh. Secondly, as the standard TSDF representation is not probabilistic, it cannot model the statistical dependencies between mapping elements. Finally, TSDF methods are limited to a single sensor type, namely depth cameras.

1.1.5 Normal Distribution Transform Maps

The normal distributions transform (NDT) mapping model was originally developed by Biber and Strasser [24] as a method for 2-D scan matching, and then independently expanded to 3-D by Takeuchi and Tsubouchi [25], and Magnusson et al. [26]. The resulting volumetric representation maintains a 3-D Gaussian distribution in each voxel. Consequently, measurements falling within a voxel are assumed to be independently and identically distributed (IID) samples drawn from a Gaussian distribution, where the sufficient statistics, namely the mean and covariance, can be calculated incrementally. Stoyanov et al. [27] showed that the resulting maps are accurate spatial representations when compared with the methods of occupancy grid mapping and polygonal meshes. They analysed all the methods on simulated and real-world data, comparing receiver operating characteristic (ROC) curves, accuracy, and runtime. NDT mapping was augmented by Saarinen et al. [28] to include the dimension of occupancy, thus creating the NDT occupancy map (NDT-OM). Their representation allows multi-resolution support for the NDT, while also introducing a temporal measure to handle dynamic environments.

Due to the IID assumption, a drawback of the NDT is that it cannot account for heterogeneous uncertainty, which is present in the sensor models and the robot pose belief. The map elements are also assumed to be statistically independent.

1.1.6 Gaussian Process Maps

A popular method of incorporating map element dependencies uses Gaussian process (GP) regression [29]—a method of regression using a non-parametric stochastic model. A set of input training points (measurements of the environment) are used to predict the output at some desired query points. In order to perform this regression, the correlation between points is described by a kernel (covariance), which is parameterised by a set of hyperparameters. GPs can only perform regression on a 1-D output; due to this, GPs were initially applied to 2.5-D elevation maps. Lang et al. [30] used an iterative, locally adaptive non-stationary kernel, which was able to handle sharp discontinuities without severe smoothing. Extending this, Plagemann et al. [31] used a separate GP to estimate the hyperparameters of each kernel. An ensemble of overlapping GPs was also tiled to cover the mapping region. This method was practically applied to foothold detection of a quadruped robot. To simultaneously handle multiple correlated outputs, Vasudevan [32] implemented a dependent GP to perform offline large-scale terrain modelling on scans of an open mine. Hadsell et al. [33] used a non-stationary kernel with hyperparameters that were a function of the uncertainty of the measurement range.

In order to use GP regression in full 3-D space, the environment needs to be described implicitly; one method of doing this uses occupancy. Occupancy maps in the context of GP mapping were first proposed by O'Callaghan and Ramos [34], who developed a framework for constructing 2-D GP occupancy maps (GPOMs). To determine the probability of occupancy, an additional

probabilistic least-squares classifier was used. This method was able to incorporate uncertainty in the sensor measurements and the robot pose into the map using the unscented transform and Gauss-Hermite quadrature. Jadidi et al. [35] improved on GPOMs by considering the case in which the pose uncertainty is significant, and incorporated this uncertainty into a warped GPOM representation using Gauss-Hermite quadrature and Monte Carlo integration. Both these methods, however, were offline post-processing procedures.

The standard GP formulation cannot be applied to online mapping due to two main issues. Firstly, it is required that all the training data be available at once. Secondly, performing online GP regression is intractable due the cubic computational complexity in the amount of training data. An approximate method of addressing both of these issues is to partition the training data using a Bayesian committee machine (BCM) [36], which is a method of combining estimators trained on different data by assuming conditional independence. BCMs have a linear computational complexity in the partition size and have been shown to decrease computational time—matching that of even the OctoMap framework [37, 38]. This method, however, combines multiple independently trained estimators, which is an inaccurate representation of the map belief.

Although GPs do not assume statistical independence between map elements, this comes at a prohibitive computational cost and, for this reason, GPs have primarily been limited to offline mapping. Their performance is also largely dependent on suitable hyperparameters, which can require optimisation throughout the mapping process. The resulting map is also a jointly Gaussian distribution over all the query points, which can become prohibitively expensive to maintain.

1.1.7 Hilbert Maps

A more recent method of incorporating the dependencies between map elements was introduced by Ramos and Ott [39]. Their approach, Hilbert maps, aims to generate a continuous occupancy representation of the environment. To achieve this, spatial measurements are projected onto a high-dimensional feature space, in which a simple linear classifier, namely logistic regression, is incrementally trained using stochastic gradient descent. The resulting representation is a parametric occupancy map of the environment. The robot pose belief and sensor uncertainties are also incorporated into the features using numerical integration. To improve efficiency, Doherty et al. [40] used the approach of fusing local maps, whereby multiple logistic regression classifiers could be combined incrementally. Guizilini and Ramos [5] proposed an efficient Hilbert maps approach by clustering measurements to decrease the dimensionality of the feature vectors. This efficient extension to the Hilbert maps framework was able to achieve training speeds rivalling that of the OctoMap framework. However, as a grid is not specified during training, this results in significantly poorer performance when querying the map in comparison to OctoMap.

Hilbert maps do not represent the uncertainty in the resulting map and therefore cannot truly be considered a probabilistic representation of the environment. The Hilbert maps framework has also only been demonstrated using accurate LiDAR sensors, and whether the approximations made will expand for less accurate sensors—like stereo cameras—is unclear.

1.1.8 Evaluation

The main attributes of the existing mapping techniques are summarised in Table 1.1. We also include our proposed mapping technique—stochastic triangular mesh (STM) mapping. From this we highlight a few important points related to the existing techniques:

Table 1.1: Comparison of the different dense mapping techniques across different attributes. We also show our stochastic triangular mesh (STM) mapping technique for comparison.

Mapping Technique	Incorporate Uncertainty	Incremental Updates	Update Efficiency	Statistical Dependencies	Explicit Surface
Polygonal Mesh	✗	✗	$\mathcal{O}(N)$	✗	✓
Occupancy Grid	✓	✓	$\mathcal{O}(N)$	✗	✗
Elevation	✓	✓	$\mathcal{O}(N)$	✗	✓
TSDF	✗	✓	$\mathcal{O}(N)$	✗	✗
NDT	Not in measurements	✓	$\mathcal{O}(N)$	✗	✓
GP	✓	✗	$\mathcal{O}(N^3)$	✓	✓
Hilbert	Not in the map	✓	$\mathcal{O}(N)$	✓	✗
STM	✓	✓	$\mathcal{O}(N)$	✓	✓

Robot pose belief uncertainty The majority of probabilistic mapping techniques—bar NDT mapping—have the ability to account for the uncertainty in the robot pose belief by marginalising it out using numerical integration[2]; this increases the uncertainty in the measurements, consequently increasing the map uncertainty. Although this represents the belief over the environment more accurately, if the uncertainty in the robot pose belief is significant, this could render the map essentially useless. Additionally, marginalising out the robot pose belief removes the dependencies between poses. This is again a problem when there are significant changes in the beliefs over previous poses, as the affected areas of the map need to be recalculated. A popular approach to alleviate these issues is to perform submapping, which we discuss in depth later (Chapter 3).

Statistical dependencies Most existing mapping techniques cannot incorporate statistical dependencies between map elements. Of the two main techniques that can, GP mapping comes at an intractable computational cost, and Hilbert maps do not capture any uncertainty in the resulting map, which does not accurately model the belief over the environment. In STM mapping, we enforce continuity in the model, which causes statistical dependencies between map elements; however, this only comes at a linear computational cost and still maintains a probabilistic map (Chapter 4).

Explicit surface models The existing dense mapping techniques either model the underlying surfaces in the environment explicitly or implicitly. An explicit surface representation is arguably more useful, because some methods of collision prediction [42, 43] require an explicit surface representation. An explicit surface can be extracted from an implicit surface representation; however, this requires an extra step of computation, and results in a model which does not represent any surface uncertainty. For example, an occupancy grid map could be converted to an explicit surface by thresholding the probability of occupancy. We therefore opt for an explicit surface representation of the environment (Chapter 4).

[2]Another method of considering the robot pose belief—applicable to all mapping techniques—would be using a Rao-Blackwellised particle filter [e.g. 41]. Here a set of random samples (particles) of the robot pose belief each maintain a map. However, this approach does not scale well in 3-D, as it requires maintaining multiple maps, which is already expensive for a single dense map.

Incremental updates Being able to incrementally update the map is critical for online operation. However, to produce an accurate map belief, it is just as important to use Bayesian reasoning when updating the map incrementally. Specifically, the map's current state—based on prior measurements—should be considered as context when fusing new measurements into the map; that is, we should use Bayes' theorem. The probabilistic techniques that use incremental Bayesian updates—occupancy grid, elevation, and NDT maps—do not incorporate statistical dependencies. Despite being a Bayesian technique, GP maps require the BCM approximation to perform incremental updates, and this approximation neglects the map's current state when updating. Additionally, the Hilbert maps method cannot represent the uncertainty in its estimates of the map, as it follows a frequentist approach. In STM mapping, we incorporate incremental Bayesian updates into a continuous representation of the environment (Chapter 4).

1.2 Solution Overview

Based on the shortcomings of the existing dense mapping techniques, in this dissertation, we develop the STM mapping technique: an explicit representation of the surface of the environment that models the statistical dependencies between mapping elements that can efficiently be updated incrementally. Similarly to a GP map, an STM map models the environment as a stochastic process; that is, a triangular mesh modelling the mean of the surface of the environment, and planar deviations from each surface element (surfel) of the mesh modelling the deviation of the actual surface from the modelled mean (Figure 1.2). This representation is therefore able to express both the mean topology of the terrain as well as its roughness. Unlike GP mapping, however, it is tractable to update an STM map, as the cost scales linearly with the number of measurements. To achieve this we use approximate message-passing techniques—namely, a combination of loopy belief propagation (LBP) and variational message passing (VMP)—to perform efficient inference on our model. In addition to our model, to address the issue of the uncertainty in the robot pose belief, we develop the STM mapping technique using triangular submapping regions, demarcated by landmarks extracted using SLAM—using the 2-D hybrid metric map (HYMM) framework [44, 45]. In order to use the HYMM framework with our mapping technique and make it applicable to more general environments, we also extend it to three dimensions.

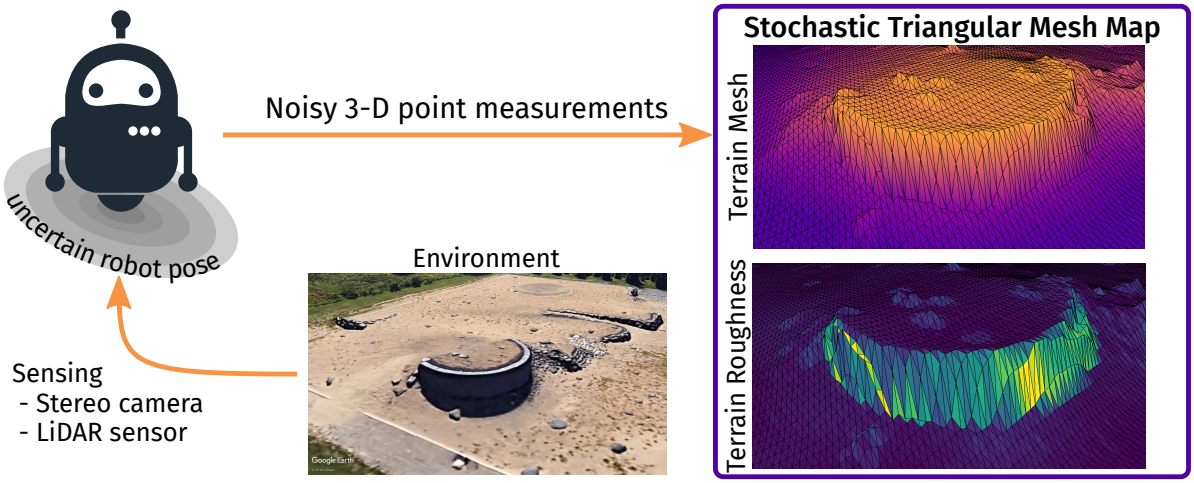


Figure 1.2: An overview of our proposed mapping technique.

1.3 Original Contributions

Given our evaluation of the existing dense mapping techniques and the comparison to our proposed technique (Table 1.1), we believe that the STM mapping technique constitutes a valuable contribution to the field of dense mapping for mobile robots. Additionally, our combination of LBP and VMP to perform efficient inference on a large-scale practical problem is a contribution to the field of statistical machine learning. This combination is also, to our best knowledge, the first of its kind.

STM mapping is able to incorporate uncertainty in a principled probabilistic manner and performs efficient incremental updates—with the updates having a linear computational complexity in the number of measurements being incorporated into the map during an update. An STM map is also an explicit representation of the surface of the environment that models statistical dependencies between the surfels in the map—this typically comes at an intractable computational cost. Individually, these attributes are not unique to the STM mapping technique. STM mapping is, however, the only technique that has all these attributes.

The majority of mapping techniques—bar gamma-SLAM [19]—build a map of the mean topology of the environment, whereas an STM map represents both the mean topology and the roughness within each surfel in the map. This provides a much richer representation of the environment, which could be particularly useful for terrain drivability analysis in terrestrial vehicles. As we also follow a Bayesian modelling approach, the uncertainty in both these aspects is captured, which is important when performing path planning—for example, regions that are uncertain due to too few measurements could be further explored.

1.3.1 Publications

The publications that have stemmed from this work are as follows:

- "Stochastic triangular mesh mapping: A terrain mapping technique for autonomous mobile robots", Robotics and Autonomous Systems, Elsevier BV [46, 47]
- "Extrinsic calibration of a push-broom LiDAR and camera using 3-D multi-planar association", 2016 IEEE Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech) [48]

1.3.2 Source Code

The source code developed during the course of this work can be found at:

`https://github.com/clintlombard/stm-mapping`

1.4 Dissertation Outline

The remainder of the dissertation is structured as follows. Next we discuss probabilistic graphical models (PGMs), which provide some background information for the rest of the dissertation (Chapter 2). Then, to motivate our decision to use the HYMM framework, we briefly explain it, and provide a 3-D extension (Chapter 3). Using this submapping framework, we then develop our STM mapping technique, representing the structure of the environment using a continuous surface (Chapter 4). Finally, we present experimental results, analysing STM maps built using both simulated and practical data (Chapter 5).

2 Probabilistic Graphical Models

As a robot navigates an environment, it obtains numerous measurements thereof. This results in a large number of interconnected variables with uncertainty—specifically between the robot's pose, the measurements of the environment, and the model of the environment (dense map). Probabilistic graphical models (PGMs) provide a compact and intuitive graphical representation over this complex, high-dimensional space, which is useful for both modelling and performing inference.

We use PGMs to model the problem of online dense mapping and, by performing inference on this model, we calculate a distribution over the dense maps that the measurements could have been generated from. Consequently, in this chapter, we discuss some aspects of PGMs that are pertinent to our proposed mapping technique.[1] We first discuss some of the different types of PGMs used for modelling (Bayesian networks) and for inference (factor and cluster graphs), in Section 2.1, and then discuss how to perform both exact and approximate inference in cluster graphs in Section 2.2.

2.1 Types of PGMs

There are many different PGMs for different types of problems (or questions); however, they all specify the same thing—the joint probability distribution over some random variables. We focus on three types of PGMs: Bayesian networks, factor graphs, and cluster graphs. In order to explain each PGM, we use the problem of simultaneous localisation and mapping (SLAM) [50, chap. 10] as an example. Localisation is typically performed using SLAM. As dense mapping is dependent on localisation, it is therefore critical to understand the core concepts of the SLAM problem. Note that we will discuss the effect of localisation in the context of mapping in Chapter 3.

The SLAM problem The SLAM problem is concerned with calculating the belief distribution over the sequence of robot poses, $X_t = (\boldsymbol{x}_0, \dots, \boldsymbol{x}_t)$ (the subscript denotes the time step), in conjunction with a (static) sparse map of re-identifiable landmarks, $L = (\boldsymbol{l}_0, \dots, \boldsymbol{l}_N)$ (the subscript denotes the landmark index), from sequences of landmark measurements, $Y_t = (\boldsymbol{y}_0^i, \dots, \boldsymbol{y}_t^j)$ (because there can be multiple observations at a time step, the superscript denotes the landmark index), and robot control commands, $U_t = (\boldsymbol{u}_0, \dots, \boldsymbol{u}_t)$. The robot control commands determine the robot's trajectory through the environment; however, it is assumed that the control commands are known.[2] In order to simplify this example, we only consider the first two time steps.

[1] For a more complete treatment of PGMs, we refer the reader to Koller and Friedman [49].

[2] The process of also solving for an optimal sequence of control commands is known as simultaneous planning, localisation and mapping (SPLAM) [51]. This planning is done to increase accuracy and information in both localisation and mapping, while simultaneously working in conjunction with a normal route planner to find a safe and optimal path to the goal.

2.1.1 Bayesian Networks

We wish to model the SLAM problem using a PGM. In order to do this, we use a *Bayesian network* [49, ch. 3], or *belief network*, which is a PGM used for modelling. Bayesian networks represent the conditional independence assumptions in the joint distribution as a directed acyclic graph (DAG) between random variable nodes, where the directed edges in a Bayesian network encode the conditional independencies; that is, a random variable (node) in the graph is conditionally independent of other variables, given its parents. An example of a Bayesian network for the SLAM problem is shown in Figure 2.1. Next we describe how this Bayesian network can be derived.

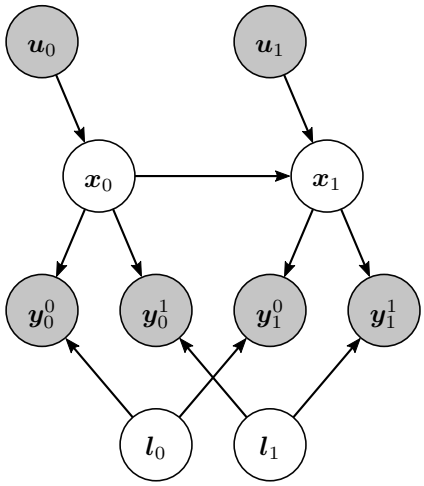


Figure 2.1: A Bayesian network representation of the joint distribution of the SLAM problem for the first two time steps. Random variables are indicated with $\odot$, conditional dependencies with arrows, and observed (or given) variables with shaded circles.

For simplicity we explicitly consider the SLAM problem for two time steps, $t = 0, 1$, with two landmarks that are both measured at each time step. Using Bayes' theorem without any conditional independence assumptions, the joint distribution can be factorised exactly as

$$p(X_1, L, U_1, Y_1) = p(Y_1|X_1, L, U_1)p(X_1|L, U_1)p(U_1|L)p(L). \tag{2.1}$$

The resulting Bayesian network for this joint distribution will be dense. However, any conditional independence assumptions will simplify the graph to a sparser representation.

For the SLAM problem, several assumptions allow us to greatly simplify the joint distribution. A measurement of the i-th landmark at the t-th time step, $\boldsymbol{y}_t^i$, is assumed to be affected only by the landmark, $\boldsymbol{l}_i$, and the robot pose from which the measurement was taken, $\boldsymbol{x}_t$. Therefore, $\boldsymbol{y}_t^i$ is conditionally independent of all other variables, given $\boldsymbol{x}_t$ and $\boldsymbol{l}_i$. This assumption simplifies the relevant factor in the joint distribution as

$$\begin{aligned} p(Y_1|X_1, L, U_1) &= p(Y_1|X_1, L) \\ &= p(\boldsymbol{y}_0^0|X_1, L, \boldsymbol{y}_0^1, \boldsymbol{y}_1^0, \boldsymbol{y}_1^1)p(\boldsymbol{y}_0^1, \boldsymbol{y}_1^0, \boldsymbol{y}_1^1|X_1, L) \\ &= p(\boldsymbol{y}_0^0|\boldsymbol{x}_0, \boldsymbol{l}_0)p(\boldsymbol{y}_0^1|\boldsymbol{x}_0, \boldsymbol{l}_1)p(\boldsymbol{y}_1^0|\boldsymbol{x}_1, \boldsymbol{l}_0)p(\boldsymbol{y}_1^1|\boldsymbol{x}_1, \boldsymbol{l}_1). \end{aligned} \tag{2.2}$$

It is also assumed that a robot pose at the t-th time step, $\boldsymbol{x}_t$, is only affected by the preceding robot pose, $\boldsymbol{x}_{t-1}$, and the control command for the t-th time step, $\boldsymbol{u}_t$. Therefore, $\boldsymbol{x}_t$ is conditionally independent of all other variables, given $\boldsymbol{x}_{t-1}$ and $\boldsymbol{u}_t$. This assumption simplifies the relevant

factor in joint distribution as

$$\begin{aligned} p(X_1|L, U_1) &= p(X_1|U_1) \\ &= p(\boldsymbol{x}_1|\boldsymbol{x}_0, U)p(\boldsymbol{x}_0|U) \\ &= p(\boldsymbol{x}_1|\boldsymbol{x}_0, \boldsymbol{u}_1)p(\boldsymbol{x}_0|\boldsymbol{u}_0). \end{aligned} \tag{2.3}$$

The conditional independence assumptions in both of these relationships are equivalently represented in a *transition factor*, $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$, and a *landmark-observation factor*, $p(\boldsymbol{y}_t^i|\boldsymbol{x}_t, \boldsymbol{l}_i)$, respectively. Note that these factors are specified based on the robot kinematics and the sensor mechanics, and are easy to construct.

The control commands are independent of the landmarks. Additionally, we assume that for both the prior distribution over the landmarks and the prior distribution over the control commands, all the variables in each prior are statistically independent. These assumptions simplify the relevant factors in joint distribution as

$$\begin{aligned} p(U_1|L)p(L) &= p(U_1)p(L) \\ &= p(\boldsymbol{u}_0)p(\boldsymbol{u}_1)p(\boldsymbol{l}_0)p(\boldsymbol{l}_1). \end{aligned} \tag{2.4}$$

By combining these simplified expressions and conditioning on observed landmark measurements, $Y_1 = \mathtt{Y}_1$, and control commands, $U_1 = \mathtt{U}_1$, the joint distribution can be simplified to

$$\begin{aligned} &p(X_1, L, U_1 = \mathtt{U}_1, Y_1 = \mathtt{Y}_1) \\ =&p(\boldsymbol{x}_0|\boldsymbol{u}_0 = \mathtt{u}_0)p(\boldsymbol{u}_0 = \mathtt{u}_0)p(\boldsymbol{y}_0^0 = \mathtt{y}_0^0|\boldsymbol{x}_0, \boldsymbol{l}_0)p(\boldsymbol{y}_0^1 = \mathtt{y}_0^1|\boldsymbol{x}_0, \boldsymbol{l}_1) \\ &p(\boldsymbol{x}_1|\boldsymbol{x}_0, \boldsymbol{u}_1 = \mathtt{u}_1)p(\boldsymbol{u}_1 = \mathtt{u}_1)p(\boldsymbol{y}_1^0 = \mathtt{y}_1^0|\boldsymbol{x}_1, \boldsymbol{l}_0)p(\boldsymbol{y}_1^1 = \mathtt{y}_1^1|\boldsymbol{x}_1, \boldsymbol{l}_1) \\ &p(\boldsymbol{l}_0)p(\boldsymbol{l}_1). \end{aligned} \tag{2.5}$$

Note that we group the factors by time step. Using these specified independence assumptions, we can now draw the Bayesian network for the SLAM problem (Figure 2.1), which equivalently represents the joint distribution in Equation 2.5.

As a Bayesian network succinctly represents the joint distribution, it allows conditional independencies to simply be read off the graph. In our example, in each of the landmark-observation factors, a landmark measurement clearly is conditionally independent of other variables given its parents—the measured landmark and the robot pose from which the measurement was taken.

Note that, in our example, the causal direction of the relationship between random variables are the same as the directions of the edges in the Bayesian network, although in general this cannot be assumed. As Bayesian networks use directed links between variables, they naturally encode causal relationships. However, in some problems there are no (obvious) causal relationships between the random variables, and therefore a PGM with undirected links might be more applicable—that is, a Markov random field (MRF) [49, ch. 4].

Both Bayesian networks and MRFs are used primarily for modelling, and to perform inference, these graphs are typically converted into factor and cluster graphs. Although both factor and cluster graphs can be used for inference, we will focus on inference in cluster graphs.

2.1.2 Factor Graphs

A *factor graph* [49, ch. 4.4.1.1] represents the fully factorised functional form of the joint distribution in an undirected graph, where the random variables are linked by the factors they share. Put differently, the nodes in the graph are the random variables and factors of the joint

distribution, and the edges are created such that a random variable is linked to all the factors in which it occurs.

From the joint distribution of the SLAM problem (Equation 2.5), we can construct the factor graph representation (Figure 2.2).

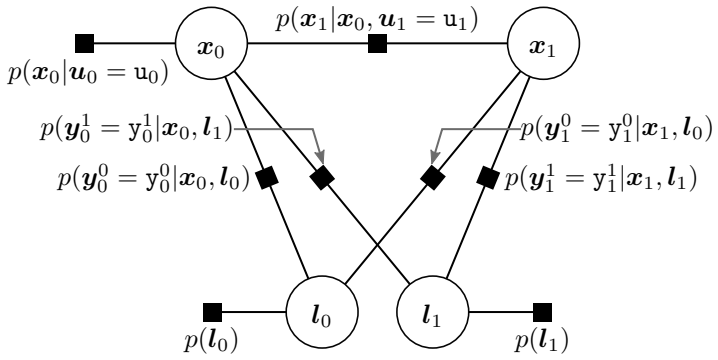


Figure 2.2: A factor graph representation of the SLAM problem. Factors are indicated with ■ and random variables with ⊙. Note that the given variables are omitted, as they are simply constants in each factor.

It should be noted that, for a particular Bayesian network, there is always a unique factor graph representation. Although we do not discuss this further, factor graphs are typically used when performing inference. We instead focus on inference in cluster graphs, which can be constructed from factor graphs.

2.1.3 Cluster Graphs

The factors of a joint distribution can be grouped into clusters, and the clusters can be connected via undirected edges, resulting in a *cluster graph* representation [49, ch. 10.1.1]. The combined factors in a cluster, i, are known as a cluster potential, $\phi_i(\mathcal{C}_i)$, and the variables in the cluster potential are known as the scope of the cluster, $\mathcal{C}_i$; note that we refer to each cluster using its scope notation. The shared links between two clusters, $\mathcal{C}_i$ and $\mathcal{C}_j$, are defined by their sepset, $\mathcal{S}_{i,j} \subseteq \mathcal{C}_i \bigcap \mathcal{C}_j$, which is a subset of the shared variables. Each cluster is a node in the graph, and the edge between any two clusters is present if their sepset is not empty.

To elucidate this notation, consider a cluster graph for the SLAM problem in Figure 2.3a (we will discuss the construction of this graph in detail shortly). Cluster $\mathcal{C}_0 = \{\boldsymbol{x}_0, L\}$ has a cluster potential, $\phi_0(\boldsymbol{x}_0, L)$, and the sepset between clusters $\mathcal{C}_0$ and $\mathcal{C}_1$ is $\mathcal{S}_{0,1} = \{\boldsymbol{x}_0, L\}$.

Although there is a unique factor graph for a Bayesian network, due to the numerous ways clusters can be grouped and the different ways they can be connected, a cluster graph, in general, is non-unique. To construct a valid cluster graph, two properties must be adhered to:

- *Family preservation property*: each factor of the joint distribution must be included in a cluster potential [49, ch. 10.1.1].
- *Running intersection property*: there can only be one path between the same variable in different clusters of the graph, where all sepsets along the path contain that variable [49, ch. 11.3.2].

Following these properties, we construct two valid cluster graphs for the SLAM problem. For

the first clustering choice (Figure 2.3a), we cluster all the factors at each time step:

$$\phi_0(\boldsymbol{x}_0, L) = p(\boldsymbol{x}_0|\boldsymbol{u}_0 = \mathtt{u}_0)p(\boldsymbol{y}_0^0 = \mathtt{y}_0^0|\boldsymbol{x}_0, \boldsymbol{l}_0)p(\boldsymbol{y}_0^1 = \mathtt{y}_0^1|\boldsymbol{x}_0, \boldsymbol{l}_1)p(\boldsymbol{l}_0)p(\boldsymbol{l}_1), \tag{2.6}$$

$$\phi_1(\boldsymbol{x}_0, \boldsymbol{x}_1, L) = p(\boldsymbol{x}_1|\boldsymbol{x}_0, \boldsymbol{u}_1 = \mathtt{u}_1)p(\boldsymbol{y}_1^0 = \mathtt{y}_1^0|\boldsymbol{x}_1, \boldsymbol{l}_0)p(\boldsymbol{y}_1^1 = \mathtt{y}_1^1|\boldsymbol{x}_1, \boldsymbol{l}_1). \tag{2.7}$$

As all the factors in the joint distribution (Equation 2.5) are present, this graph satisfies the family preservation property. The sepset between both clusters must be the intersection between both cluster scopes, $\{\boldsymbol{x}_0, L\}$, to satisfy the running intersection property.

For the second clustering choice (Figure 2.3b), we cluster each transition factor and landmark-observation factors at each time step separately:

$$\phi_0(\boldsymbol{x}_0) = p(\boldsymbol{x}_0|\boldsymbol{u}_0 = \mathtt{u}_0), \tag{2.8}$$

$$\phi_1(L) = p(\boldsymbol{l}_0)p(\boldsymbol{l}_1), \tag{2.9}$$

$$\phi_2(\boldsymbol{x}_1, L) = p(\boldsymbol{y}_0^0 = \mathtt{y}_0^0|\boldsymbol{x}_0, \boldsymbol{l}_0)p(\boldsymbol{y}_0^1 = \mathtt{y}_0^1|\boldsymbol{x}_0, \boldsymbol{l}_1), \tag{2.10}$$

$$\phi_3(\boldsymbol{x}_1, \boldsymbol{x}_0) = p(\boldsymbol{x}_1|\boldsymbol{x}_0, \boldsymbol{u}_1 = \mathtt{u}_1), \tag{2.11}$$

$$\phi_4(\boldsymbol{x}_1, \boldsymbol{y}_1, L) = p(\boldsymbol{y}_1^0 = \mathtt{y}_1^0|\boldsymbol{x}_1, \boldsymbol{l}_0)p(\boldsymbol{y}_1^1 = \mathtt{y}_1^1|\boldsymbol{x}_1, \boldsymbol{l}_1). \tag{2.12}$$

Again, as all the factors in the joint distribution (Equation 2.5) are present, this graph satisfies the family preservation property. To satisfy the running intersection property, however, the sepset between cluster $\mathcal{C}_2$ and $\mathcal{C}_3$ is chosen to be empty (could have contained $\boldsymbol{x}_0$), and similarly for the sepset between clusters $\mathcal{C}_1$ and $\mathcal{C}_4$ (could have contained L).

The first clustering choice yields a cluster graph with a chain (acyclic) structure, whereas the second has a loopy (cyclic) structure. As we will soon see, this difference will affect the choice of inference algorithm.

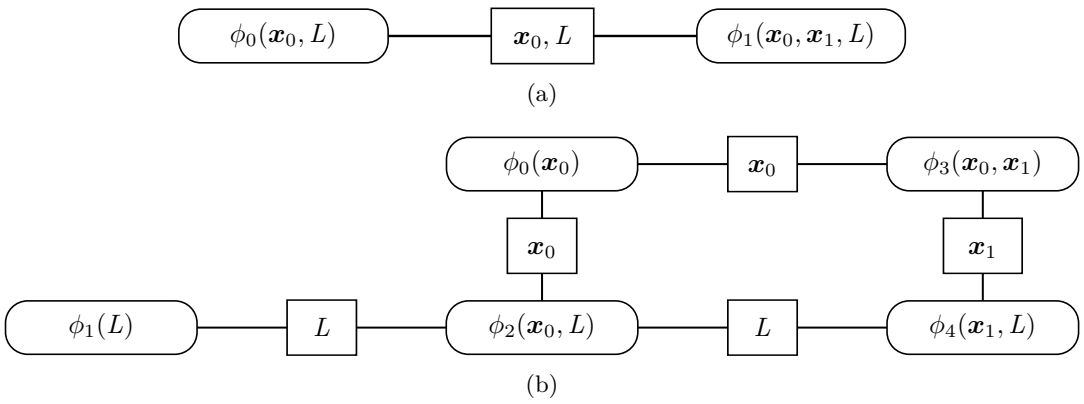


Figure 2.3: Two different cluster graph representations of the SLAM problem: (a) an acyclic graph, and (b) a cyclic graph. Cluster potentials are denoted $\phi(\cdot)$, and the sepset between clusters is indicated with $\boxdot$.

2.2 Inference in PGMs

Once a problem is modelled using a PGM, the next task is to draw meaningful conclusions from the model. Given some measurements or observations, the task of inference is to determine the belief distribution for a particular model

$$\mathcal{B}(X) \triangleq p(X|Z = \mathtt{z}), \tag{2.13}$$

where the belief distribution, denoted by $\mathcal{B}(\cdot)$, is the posterior distribution over some random variable(s), X, given all observations, $Z = \mathtt{Z}$. For our example of the SLAM problem, the given observations are the robot control commands, $U_t = \mathtt{U}_t$, and the landmark observations, $Y_t = \mathtt{Y}_t$. Note that, according to Bayes' rule, a belief distribution can be related to the joint distribution

$$\begin{aligned} \mathcal{B}(X) &= \frac{p(X, Z = \mathtt{Z})}{p(Z = \mathtt{Z})} \\ &\propto p(X, Z = \mathtt{Z}). \end{aligned} \tag{2.14}$$

In the (ideal) case where all the cluster potentials of a joint distribution are convenient closed-form distributions, calculating the belief distribution can be as simple as calculating the product of all the cluster potentials and normalising the result. However, in most cases it is intractable to calculate the joint belief distribution—there are hardly any real-world problems of interest for which one can tractably calculate the joint belief distribution. An attractive alternative to calculating the joint belief distribution is to rather perform marginal inference—that is, calculate marginal belief distributions. By integrating over the full belief distribution, a marginal belief distribution over some variables, $Y \subset X$, can be calculated as

$$\mathcal{B}(Y) = \int_{X \setminus Y} \mathcal{B}(X) \, \mathrm{d}X \setminus Y. \tag{2.15}$$

Alternatively, if it is not necessary to calculate a distribution over the latent variable of interest, the most likely value of the belief distribution can be calculated instead:

$$X_{\mathrm{MAP}} = \arg\max_{X} \mathcal{B}(X). \tag{2.16}$$

This process of calculating the most likely combination of latent variables is know as maximum a posteriori (MAP) inference. However, there are no distributions and therefore one does not have any indication of the uncertainty in the answer. The result of this inference approach also does not capture the posterior dependencies between the variables. We therefore focus instead on calculating marginal belief distributions, which captures the dependencies between subsets of the latent variables.

In some cases it is possible to calculate exact marginal belief distributions in convenient closed-form solutions—for example using belief propagation (BP), which we will discuss subsequently. However, in most cases, this is either intractable (due to numerous random variables) or impossible (due to non-linear relationships between variables), and we must therefore turn to approximate inference techniques to be able to draw conclusions from a model. Two of the main approaches for performing approximate inference are sampling-based approaches and variational inference techniques.

Markov chain Monte Carlo (MCMC) methods—such as the Metropolis–Hastings algorithm [52, ch. 11.2.2] and Gibbs sampling [52, ch. 11.3]—form part of the class of sampling-based inference techniques that can approximate a probability distribution as a sequence of samples. However, this does not scale to high-dimensional problems such as dense mapping, which would require too many samples to effectively represent a map.

Another less popular class of approximate inference techniques use functional approximations. For example, using a first-order Taylor series approximation results in the extended Kalman filter (EKF) and other similar variants to the Kalman filter (KF) [50, ch. 3]. Alternatively, the Laplace approximation fits a Gaussian distribution to the mode of a probability distribution using a second-order Taylor series approximation [52, ch. 4.4]. Both these techniques, however, are only applicable to Gaussian approximating distributions. Additionally, due to their reliance on linear

approximations, the performance of these techniques is heavily influenced by the linearisation point.

Variational inference techniques formulate inference as an optimisation problem, trying to find the optimal distribution to approximate the exact belief distribution by minimising an information divergence. We first focus on performing exact inference using BP, but discuss variational inference techniques in depth in Section 2.2.3.

2.2.1 Belief Propagation

Marginal inference can often be performed without having to explicitly compute the joint belief distribution. This is made possible using belief propagation (BP) [49, ch. 10.2.2], a type of message-passing inference technique. It is important to note that BP results in exact inference when performed on an acyclic PGM—that is, graphs with a chain or tree structure.

Belief propagation calculates messages between every cluster in a cluster graph. Messages are defined over the sepset between two connected clusters, in both directions, and can be used to calculate the marginal belief distribution. The outgoing message from cluster $\mathcal{C}_i$ to cluster $\mathcal{C}_j$ is denoted $\delta_{i\to j}(\mathcal{S}_{i,j})$. In order to calculate this message, it first is necessary to collect the incoming messages to $\mathcal{C}_i$ from all neighbouring clusters (excluding $\mathcal{C}_j$). The product of these messages, with the cluster potential $\phi_i(\mathcal{C}_i)$, is then marginalised over the scope of the sepset to calculate $\delta_{i\to j}(\mathcal{S}_{i,j})$. This process of calculating an outgoing message is described by

$$\delta_{i\to j}(\mathcal{S}_{i,j}) = \int_{\setminus \mathcal{S}_{i,j}} \phi_i(\mathcal{C}_i) \prod_{k\in\{\mathrm{Nb}_i\setminus\{j\}\}} \delta_{k\to i}(\mathcal{S}_{i,k})\,\mathrm{d}\setminus \mathcal{S}_{i,j}, \tag{2.17}$$

where Nb_i is the set of integer indices of the neighbouring clusters to cluster $\mathcal{C}_i$, and $\setminus\mathcal{S}_{i,j}$ denotes all variables in the integrand that are not in the sepset $\mathcal{S}_{i,j}$. This process of calculating a message is also known as sending a message. Due to the form of Equation 2.17, BP is commonly referred to as the integral-product or sum-product algorithm. After repeating this process of sending messages until the messages are constant, the graph is calibrated. In the exact inference case, messages are scheduled such that each message is sent only once, which is only possible if the initially selected cluster is only connected to one other cluster—that is, the selected cluster is a leaf node. Once calibrated, a cluster graph can be queried efficiently to extract marginal belief distributions. The marginal belief distribution over the variables in a cluster, $\mathcal{C}_i$, is calculated as

$$\mathcal{B}(\mathcal{C}_i) \propto \phi_i(\mathcal{C}_i) \prod_{k\in\mathrm{Nb}_i} \delta_{k\to i}(\mathcal{S}_{i,k}). \tag{2.18}$$

This belief distribution is often referred to as the cluster belief. Alternatively, the marginal belief distribution over the variables in a sepset, $\mathcal{S}_{i,j}$, can be calculated as

$$\mathcal{B}(\mathcal{S}_{i,j}) \propto \delta_{j\to i}(\mathcal{S}_{i,j})\delta_{i\to j}(\mathcal{S}_{i,j}). \tag{2.19}$$

Looking again at our example of the SLAM problem, and by applying BP to the acyclic cluster graph of the problem (Figure 2.3a), we send all the messages in the graph (Figure 2.4). We can now query this calibrated graph for marginal belief distributions. For example, the sepset belief distribution is calculated according to Equation 2.19 as

$$\mathcal{B}(\boldsymbol{x}_0, L) \propto \delta_{0\to 1}(\boldsymbol{x}_0, L)\delta_{1\to 0}(\boldsymbol{x}_0, L). \tag{2.20}$$

The online SLAM problem is only concerned with calculating the marginal belief distribution over the most recent robot pose and the landmark map, which is calculated according to Equation 2.18

with $\boldsymbol{x}_0$ marginalised out:

$$\mathcal{B}(\boldsymbol{x}_1, L) \propto \int_{\boldsymbol{x}_0} \delta_{0\to1}(\boldsymbol{x}_0, L)\phi_1(\boldsymbol{x}_0, \boldsymbol{x}_1, L)\,\mathrm{d}\boldsymbol{x}_0. \tag{2.21}$$

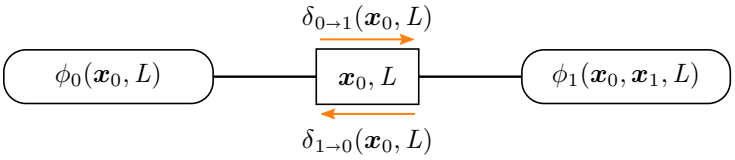


Figure 2.4: The acyclic cluster graph of the SLAM problem showing all the messages.

Note that using BP to solve for the online SLAM problem with this cluster graph structure, messages only need to be send forward. This approach is equivalent to performing recursive state estimation and, in the case of exact Gaussian distributed factors, the marginal query in Equation 2.21 results in a Kalman filter [50, chap. 3].

2.2.2 Loopy Belief Propagation

BP results in exact inference when applied to an acyclic cluster graph. However, in some cases, creating an acyclic cluster graph requires constructing clusters with large scopes; in the worst case, the scope could be all the variables in the problem. It is expensive to both store these large clusters and to calculate the messages, which is not a scalable approach in most practical problems. This can be avoided when relaxing the constraint on requiring an acyclic cluster graph when performing BP.

Loopy belief propagation (LBP) [49, ch. 11.3] is an approximate inference technique, whereby BP is iteratively applied to a cyclic graph until all the messages converge. As the calculation of an outgoing message is dependent on the incoming messages (Equation 2.17), all messages in the graph need to initialised. A typical choice for this initialisation is to make the messages uninformative.

By allowing cyclic cluster graphs, clusters with smaller scopes can be constructed, which allows for a more scalable inference algorithm. However, this comes at a price. Because LBP in an approximate inference algorithm, the solutions can be inexact. Additionally, LBP is not guaranteed to converge. Despite these problems, LBP has been shown to perform well in practice—the first famous example being the error-correction code scheme "Turbo codes" by Berrou et al. [53], which McEliece et al. [54] showed to be equivalent to BP applied to a cyclic graph.

After performing LBP on the cyclic cluster graph of the SLAM problem (Figure 2.3b)—sending all messages until convergence—we can answer marginal inference queries. Once calibrated, the graph can be queried for approximate belief distributions. For example, an approximation of the exact belief distribution for the online SLAM problem, $\tilde{\mathcal{B}}(\boldsymbol{x}_1, L)$, can be calculated according to Equation 2.18 as

$$\tilde{\mathcal{B}}(\boldsymbol{x}_1, L) \propto \delta_{3\to4}(\boldsymbol{x}_1)\delta_{2\to4}(L)\phi_4(\boldsymbol{x}_1, L). \tag{2.22}$$

A similar approach to what we have discussed was applied to the SLAM problem by Ranganathan et al. [55], but instead using a factor graph.

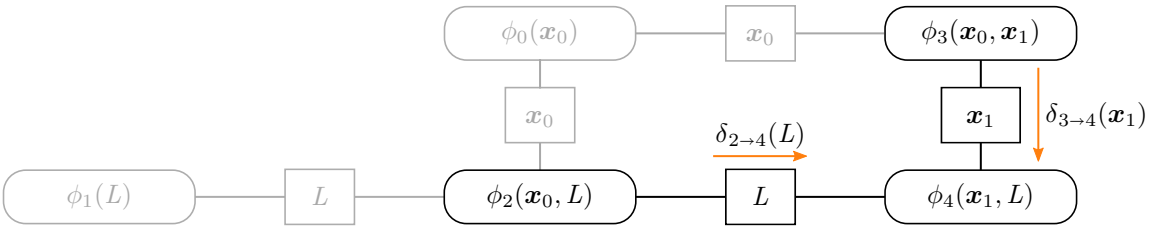


Figure 2.5: The cyclic cluster graph of the SLAM problem showing the messages required to calculate the approximate cluster belief over $\boldsymbol{x}_1, L$.

2.2.3 Variational Inference

In most problems of interest, the functional forms of the cluster potentials do not have convenient closed-form solutions—due to non-linear relationships between random variables, or a mismatch in the distribution types of the random variables. This makes it intractable to calculate the products and marginalisations required to perform inference. One method to perform approximate inference in situations like this is using variational inference.[3] The aim of variational inference is to find convenient closed-form approximations to the belief distributions of interest in a problem. This is achieved by projecting a desired exact (intractable) distribution onto an approximating family of distributions.

Variational inference defines the projection between an exact distribution, $p(X)$, and an approximate distribution, $q(X)$, in a family of approximating distributions, $\mathcal{F}$, as the minimisation of a chosen divergence measure

$$q^*(X) = \underset{q(X)\in\mathcal{F}}{\arg\min}\, D\left(q(X)\|p(X)\right), \tag{2.23}$$

where $q^*(X)$ is the optimal approximate distribution. A commonly used divergence measure is the Kullback-Leibler (KL) divergence,

$$D_{\mathrm{KL}}\left(q(X)\,\|\,p(X)\right) \triangleq \int_X q(X)\log\frac{q(X)}{p(X)}\,\mathrm{d}X, \tag{2.24}$$

which is a positive measure of relative entropy between two probability distributions $p(X)$ and $q(X)$. The Kullback-Leibler (KL) divergence is asymmetric—that is, $D_{\mathrm{KL}}\left(q(X)\,\|\,p(X)\right) \neq D_{\mathrm{KL}}\left(p(X)\,\|\,q(X)\right)$ in general. If $p(X)$ is the exact posterior and $q(X)$ is an approximation, then $D_{\mathrm{KL}}\left(q(X)\,\|\,p(X)\right)$ is referred to as the exclusive KL divergence. The reverse of this, $D_{\mathrm{KL}}\left(p(X)\,\|\,q(X)\right)$, is referred to as the inclusive KL divergence. This process of minimising the exclusive KL divergence is also referred to as *information projection*; alternatively, minimising the inclusive KL divergence is referred to as *moment projection*.

In contrast to performing a global projection of the full desired belief distribution, a group of techniques known as *approximate message passing*[4] distributes the problem of variational inference by instead incrementally performing local projections with the aim of approximately minimising the global divergence. In essence, message passing minimises the global divergence using coordinate descent. This can be explained intuitively in the context of performing inference in a cluster graph. Approximate message passing iteratively calculates each cluster potential; at each iteration, a certain cluster potential is approximated, while keeping all other cluster potentials constant. The constant cluster potentials determine the incoming message of the

[3] For a review of the principles of variational inference, see Blei et al. [56].
[4] For a review of approximate message-passing techniques, see Minka [57].

cluster potential being approximated, and are used as context to update the approximation (how the approximation is updated depends on the chosen divergence measure). This process is repeated until convergence, upon which the global divergence is assumed to be at a (local) minimum.

Performing message passing using the information projection onto the exponential family of distributions is known as variational message passing (VMP) [58]. Alternatively, using the moment projection onto the exponential family of distributions results in the expectation propagation (EP) message-passing technique [59]. Note that, in some sense, LBP is equivalent to EP, as it can be shown that LBP also uses the moment projection. However, LBP does not place an explicit constraint on the exponential family of distributions [57].

A notable property of VMP is that a fixed point in performing message passing is also a stationary point in the global divergence—in other words, minimising local divergences is equivalent to minimising the global divergence. VMP is also the only message-passing technique with this property, as a result of VMP using the exclusive KL divergence. Other approximate message-passing techniques, however, only minimise the local divergences as an approximation to minimising the global divergence [57].

In order to incorporate probabilistic information from the robot pose and the measurements of the environment into our stochastic triangular mesh (STM) map, we model the problem with PGMs and use approximate message-passing techniques (VMP and LBP) to incrementally and efficiently update the map.

3 Inertial Reference Frames in Dense Mapping

At the center there is truth. As you travel, so error creeps in.

Sir Terry Pratchett, *Small Gods*

The uncertainty in the robot pose belief is ever present when performing online dense mapping. If the robot pose belief contains significant uncertainty, it creates problems when incorporating measurements into the dense map. In this chapter, we explore submapping—an approach that reduces the effect of the uncertainty in the robot pose belief—and motivate our choice of submapping framework in the stochastic triangular mesh (STM) mapping technique.

All robotic systems are described in terms of inertial reference frames (IRFs). When discussing the issue of the uncertainty in the robot pose belief in dense mapping, it is important to understand the effect that the choice of IRF has on the resulting map. Some mapping techniques have used an allocentric IRF to ensure that the region of the map currently surrounding the robot has lower uncertainty [18] or a finer grid resolution [13]. In contrast, most mapping techniques focus on building globally consistent maps in a fixed, single-privileged IRF—often referred to as a global IRF. Due to the compounding effect of uncertain motion on the robot pose belief, regions of the map further away from the origin of the chosen IRF are increasingly uncertain; in the absence of absolute position sensing, this uncertainty is unbounded. Furthermore, when performing simultaneous localisation and mapping (SLAM) in a global IRF and then performing loop closure, large changes in beliefs over previous robot poses can be induced. Despite this, the *relative* uncertainty between successive robot poses is bounded by the—typically confident—motion dynamics. The submapping approach takes advantage of this by segmenting the environment into several submaps, each with its own IRF. This approach is used in some of the recent state-of-the-art vision-based SLAM techniques, namely LSD-SLAM by Engel et al. [60], and ORB-SLAM by Mur-Artal and Tardós [61]; both of these techniques use keyframes as intermediate submaps during SLAM. In the case of dense mapping in indoor environments, the map can be semantically segmented by room [62]. In general environments, a typical method of defining submaps uses overlapping rectangular cuboids to span the mapping space [63, 64]. However, this causes two issues: the map contains redundant regions, and it is non-trivial to accurately describe new measurements in terms of the IRFs of previously observed submaps. Hybrid metric maps (HYMMs) [44, 45] is a 2-D submapping technique that addresses both of these issues by using relative IRFs based on the sparse landmark map obtained from SLAM. In Section 3.1, we review the HYMM framework and present a 3-D extension to it. Utilising this framework, we present a principled method of decoupling measurements of the environment from the robot pose belief (Section 3.2).

Notational remark A point in the global IRF, $\boldsymbol{x}$, is expressed in the body reference frame (BRF) of the robot as $\boldsymbol{x}^B$, and in a relative IRF as $\boldsymbol{x}^R$.

3.1 Relative Inertial Reference Frames

The HYMM submapping technique segments the environment by constructing a triangular mesh between selected landmarks (Figure 3.1a). A relative IRF is associated with each triangular submap, defined in terms of the three landmarks at the vertices of the triangular region. To elucidate this definition, consider a submap demarcated by the convex hull of the landmarks $\boldsymbol{l}_0$, $\boldsymbol{l}_\alpha$ and $\boldsymbol{l}_\beta$. A 2-D point, $\boldsymbol{m}$, in the global IRF can be described by

$$\begin{aligned} \boldsymbol{m} &= \alpha(\boldsymbol{l}_\alpha - \boldsymbol{l}_0) + \beta(\boldsymbol{l}_\beta - \boldsymbol{l}_0) + \boldsymbol{l}_0 \\ &= \alpha\boldsymbol{a} + \beta\boldsymbol{b} + \boldsymbol{l}_0. \end{aligned} \tag{3.1}$$

For $\boldsymbol{m}$ to be within the submap, α and β must satisfy

$$\alpha + \beta \leq 1 \quad \text{and} \quad \alpha, \beta \in [0, 1]. \tag{3.2}$$

The 2-D relative IRF axes are defined by $\boldsymbol{a}$ and $\boldsymbol{b}$—$\boldsymbol{l}_0$ is an offset—and (α, β) are the coordinates of a 2-D point $\boldsymbol{m}$ in the relative IRF.

To extend the 2-D relative IRFs of the HYMMs to 3-D, a third axis is required. One could project the 3-D landmarks onto the horizontal plane and use the vertical axis of the global IRF. However, this approach couples the relative IRF to the global IRF, negating the true benefits of a relative IRF. Guivant et al. [44] suggested—but did not develop—two ideas of extending HYMMs to 3-D. The relative IRF axes could be constructed using four landmarks, forming tetrahedral submapping regions; however, it could be impractical to adequately span the mapping space. Alternatively, the 2-D approach can be adapted to 3-D by defining the submapping region being normal to the triangular faces of the 3-D landmark mesh. We develop the latter approach, extending the relative IRF to 3-D by introducing the third axis as

$$\boldsymbol{n} = \frac{\boldsymbol{a} \times \boldsymbol{b}}{\|\boldsymbol{a} \times \boldsymbol{b}\|}. \tag{3.3}$$

Using this definition, a 3-D point, $\boldsymbol{m}$, can be described as

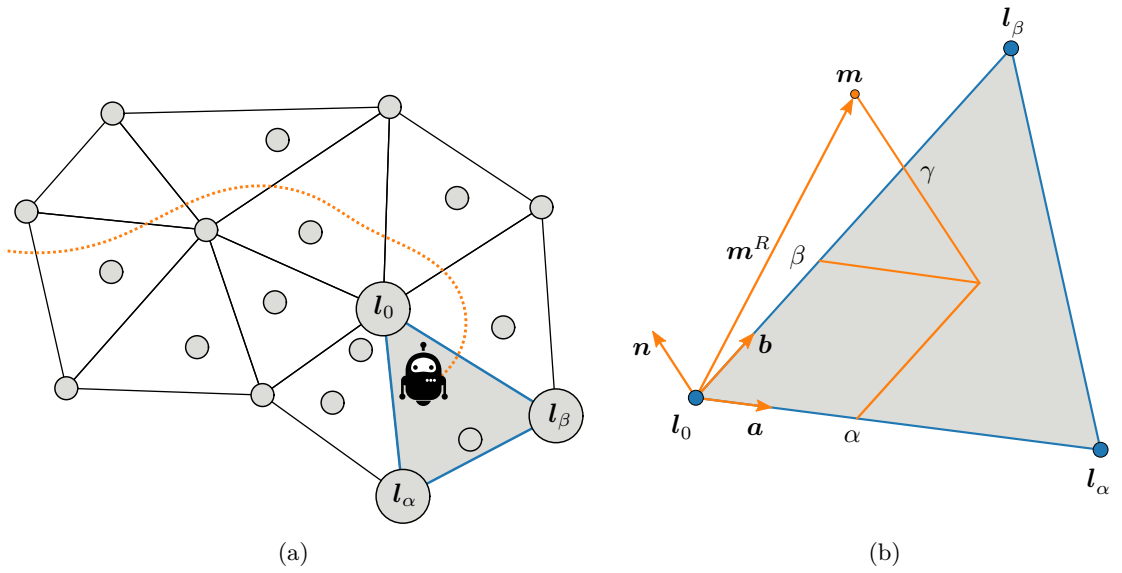
$$\begin{aligned} \boldsymbol{m} &= \alpha\boldsymbol{a} + \beta\boldsymbol{b} + \gamma\boldsymbol{n} + \boldsymbol{l}_0 \\ &= \begin{bmatrix} \boldsymbol{a} & \boldsymbol{b} & \boldsymbol{n} \end{bmatrix} \boldsymbol{m}^R + \boldsymbol{l}_0, \end{aligned} \tag{3.4}$$

where $\boldsymbol{m}^R = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^\intercal$ is a point in the 3-D relative IRF (Figure 3.1b). The constraints in Equation 3.2 also hold for $\boldsymbol{m}$ to fall within the submap, with no additional constraints placed on γ.

There is, however, a caveat to using this relative IRF description. As the definition of the relative IRF relies on landmarks, it is necessary for the chosen landmarks to be robustly and persistently identifiable. While this is still an open problem for general environments, given the progress that has been made in the past decade in feature and place recognition [65], we believe that this will be possible in the future. Despite this, there are practical applications, in more controlled environments, where easily identifiable landmarks could be placed manually in the environment—for example in land surveying, 3-D object reconstruction, or factory operation. We practically demonstrate an example of such an application in Section 5.5.

3.2 Decoupling Dense Measurements from the Robot Pose Belief

With the relative IRF definition in hand, we now present a method of decoupling measurements of the surface of the environment from their associated robot pose beliefs. Consequently, this



(a) (b)

Figure 3.1: (a) The hybrid metric map (HYMM) submapping method [44, 45] segments the environment using landmarks. (b) In our 3-D extension to the HYMM representation, a point $\boldsymbol{m}$ within a submap can be fully described relative to these landmarks.

method decouples the process of localisation from that of dense mapping. We achieve this decoupling by transforming surface measurements from the BRF of the robot to the relative IRF of the associated submap—as opposed to transforming to the global IRF.

We model the process of observing a point on the surface of the environment (in the BRF of the robot) using a sensor beam model, whereby a noisy measurement, $\boldsymbol{z}^B$, is generated by a point, $\boldsymbol{m}^B$, on the surface of the environment. Most sensors observe multiple environment surface points at a single time step. We denote the sequence of measurements as $Z^B = (\boldsymbol{z}_i^B)$, and the associated environment surface points as $M^B = (\boldsymbol{m}_i^B)$. Assuming an uninformative prior over M^B, we can calculate the belief over M^B as

$$\mathcal{B}(M^B) \triangleq p(M^B|Z^B) \propto \prod_i p(\boldsymbol{z}_i^B|\boldsymbol{m}_i^B), \tag{3.5}$$

where the belief distribution, denoted by $\mathcal{B}(\cdot)$, given all observations, is the posterior distribution over some random variables. The conditional distribution, $p(\boldsymbol{z}_i^B|\boldsymbol{m}_i^B)$, describes the measurement model, which is known beforehand for each sensor. Following from this, the belief over M^B and the SLAM states—the current robot pose, $\boldsymbol{x}$, and the sparse landmark map, L, in the global IRF—can be factorised as

$$\begin{aligned}\mathcal{B}(\boldsymbol{x}, L, M^B) &\triangleq p(\boldsymbol{x}, L, M^B|U, Y, Z^B) \\ &\approx \underbrace{p(\boldsymbol{x}, L|U, Y)}_{\mathcal{B}(\boldsymbol{x},L)} \underbrace{p(M^B|Z^B)}_{\mathcal{B}(M^B)},\end{aligned} \tag{3.6}$$

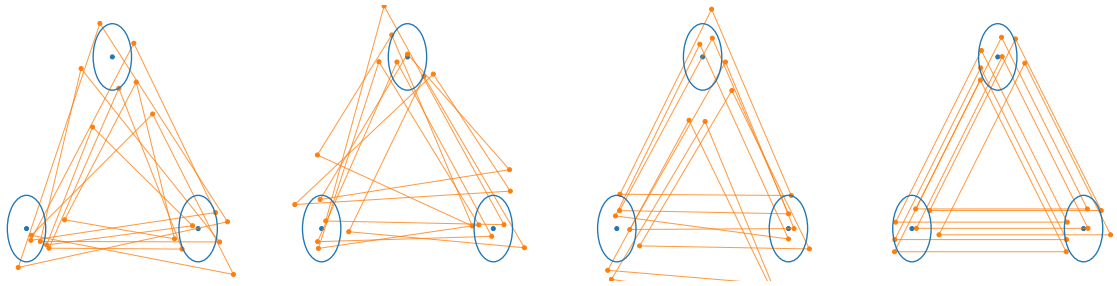
where U is the sequence of robot control commands, and Y is the sequence of landmark measurements. This factorisation makes the assumption that M^B is conditionally independent of the SLAM states given Z^B. It should also be noted that $\mathcal{B}(\boldsymbol{x}, L)$ is the result of performing SLAM. Although some SLAM algorithms only calculate the maximum a posteriori (MAP) estimate of this belief, we only consider SLAM algorithms that calculate the full belief distribution.

Going forward, we assume $\mathcal{B}(\boldsymbol{x}, L, M^B)$ to be Gaussian distributed. This does not necessarily limit the SLAM algorithm to one that makes the Gaussian assumption, but simply requires an

algorithm where the resulting belief distribution can be projected onto the Gaussian family of distributions—for example by using moment matching.

To transform M^B to the relative IRF, it must first be transformed to the same IRF as the SLAM states—the global IRF. We use the unscented transform (UT) [66, 67] to transform the environment surface points from the body RF to the global IRF, $M^B \to M$, and then from the global to the relative IRF, $M \to M^R$. The transformation $M^B \to M$ is determined by the robot pose in the global IRF, $\boldsymbol{x}$. As a consequence of this, M and $\boldsymbol{x}$ are statistically dependent, and therefore, in general, the belief distribution over all the states in the global IRF, $\mathcal{B}(\boldsymbol{x}, L, M)$, cannot be factorised. If dense mapping was performed in the global IRF, then the global IRF measurement belief, $\mathcal{B}(M)$, would be incorporated into the map. In order for this mapping process to be tractable, the environment points are assumed to be statistically independent of one another; however, this is almost never the case; because of the transformation to the global IRF, they are usually highly correlated with one another (as we will show subsequently).

In order to decouple the process of dense mapping from the robot pose, we need to remove the statistical dependency between M and $\boldsymbol{x}$. Transforming M to the relative IRF achieves this through an important characteristic of SLAM: as clusters of neighbouring landmarks are often observed from a single robot pose—or from successive robot poses—these landmarks are highly correlated.[1] Therefore, although the marginal belief over each landmark in the global IRF can contain significant uncertainty, the relative positions between neighbouring landmarks are known with little uncertainty. More importantly, the landmark correlation increases monotonically with the number of measurements [68] and, in the limit, becomes perfectly correlated for the linear Gaussian case [69]. In the case of perfect correlation, there is no uncertainty between the relative landmark positions. We illustrate the effect of various degrees of correlation on the relative landmark positions for a synthetic Gaussian belief over three 2-D landmarks (Figure 3.2). For perfect linear correlation, the triangles formed by each 6-D sample are identical in shape, whereas the shapes are inconsistent if there is no correlation. In practice, only high correlations between landmarks are achievable, although Nieto et al. [45] show that the relative uncertainty between landmarks is sufficiently low for relative IRFs to be used in mapping.



(a) $\rho = 0$ (b) $\rho = 0.5$ (c) $\rho = 0.95$ (d) $\rho = 1$

Figure 3.2: The effect of the degree of linear correlation on the relative positions of three 2-D landmarks. The correlation is quantified using the Pearson correlation coefficient, ρ, for a synthetic Gaussian belief over the landmarks. Shown in blue are the marginal distributions of each landmark—the ellipses indicate the first standard deviations. Each orange triangulation of points represents a single 6-D sample from the joint distribution over all three landmarks.

Similarly, the robot pose will be highly correlated with nearby landmarks, and consequently the relative positions of the robot pose and landmarks will also be known accurately. In the ideal case, in which the robot and landmarks' positions in the global IRF are perfectly correlated,

[1] The degree of statistical dependency can be quantified using linear correlation—specifically, using the Pearson correlation coefficient, $\rho \in [-1, 1]$. In the Gaussian case, linear correlation equates to statistical dependency; that is, $\rho = 0$ equates to statistical independence, and $\rho = \pm 1$ equates to perfect linear dependence.

their relative positions would be perfectly known. If we then describe M with respect to the landmarks (that is, transforming $M \rightarrow M^R$), M^R would be statistically independent of both the landmarks and the robot pose. This is because the relative positions of the robot and landmarks would be known perfectly, and therefore the uncertainty in the belief over M^R would only stem from measurement uncertainty. Although the relative positions are never perfectly known, they are usually accurately known and it is reasonable to approximate the belief over the SLAM states and M^R as statistically independent

$$\mathcal{B}(\boldsymbol{x}, L, M^R) \approx \mathcal{B}(\boldsymbol{x}, L)\mathcal{B}(M^R), \tag{3.7}$$

where

$$\mathcal{B}(M^R) \propto \prod_i p(\boldsymbol{z}_i^R | \boldsymbol{m}_i^R). \tag{3.8}$$

To illustrate the validity of this approximation, we visualise the absolute value of the Pearson correlation coefficient matrix for the Gaussian belief distributions $\mathcal{B}(\boldsymbol{x}, L, M)$ and $\mathcal{B}(\boldsymbol{x}, L, M^R)$ of a practical stereo vision dataset (Figures 3.3a and 3.3b). In $\mathcal{B}(\boldsymbol{x}, L, M)$, we see that there is indeed a high correlation between $\boldsymbol{x}$, L and M. Upon transforming $M \rightarrow M^R$, the resulting belief $\mathcal{B}(\boldsymbol{x}, L, M^R)$ has negligible correlations between the SLAM states and M^R. The correlations are almost zero, which equates to SLAM states and M^R being approximately statistically independent[2]. Therefore, this process of transforming to a relative IRF decouples the process of dense mapping from the robot pose.

The main advantage of this decoupling process is highlighted in the scenario in which loop closure is performed. The belief over the environment surface points in the relative IRF is decoupled from the SLAM states; hence, when the belief over the SLAM states changes due to loop closure, the environment surface points no longer need to be reintegrated into the map—as would be required in a global IRF. Instead, the burden is placed on the SLAM algorithm to maintain (or allow the extraction of) marginal distributions over the landmarks demarcating the submaps, which is a far more tractable approach.

In order to evaluate the efficacy of performing dense mapping in a relative IRF—as opposed to a global IRF—we compare the uncertainty in the beliefs over the environment surface points in a relative and global IRF. For the case of mapping in a relative IRF, we consider our approximate belief $\mathcal{B}(M^R)$ (Equation 3.8). We also consider the ideal case, in which our statistical independence approximation is exact—that is, the robot pose and landmarks are perfectly correlated or, in other words, the relative positions between $\boldsymbol{x}$ and L are perfectly known. The uncertainty in this ideal belief, $\mathcal{B}_{\text{ideal}}(M^R)$, is solely dependent on the measurement uncertainty. For the case of mapping in a global IRF, we consider the belief over the environment surface points in a global IRF, $\mathcal{B}(M)$. To compare the beliefs in the relative IRF with that of the global IRF, we need to evaluate all the beliefs using the same units. We therefore deterministically transform $\mathcal{B}(M)$ to the relative IRF using the maximum likelihood value of the landmarks; we denote the resulting belief $\mathcal{B}_{\text{global}}(M^R)$. The resulting comparison of all the beliefs is shown in Figure 3.3c. From this we can conclude that the uncertainty in belief over the environment surface point used for dense mapping can be significantly less when using a relative IRF instead of a global IRF. Additionally, the uncertainty in the belief becomes almost solely dependent on the measurement uncertainty. Nieto et al. [45] noted similar effects when representing other landmarks in a relative IRF; that is, a dramatic decrease in the correlation between landmarks in the relative and global IRFs, and a decrease in the marginal landmark belief uncertainty in the relative IRF.

In this chapter, we presented a method of decoupling the robot pose from the process of dense mapping. By transforming the information from the measured environment surface points to

[2] Jointly Gaussian-distributed variables are statistically independent if and only if they are uncorrelated.

a relative IRF, we have approximately removed their statistical dependence on the robot pose. Next we look at incorporating this information into our model of the environment.

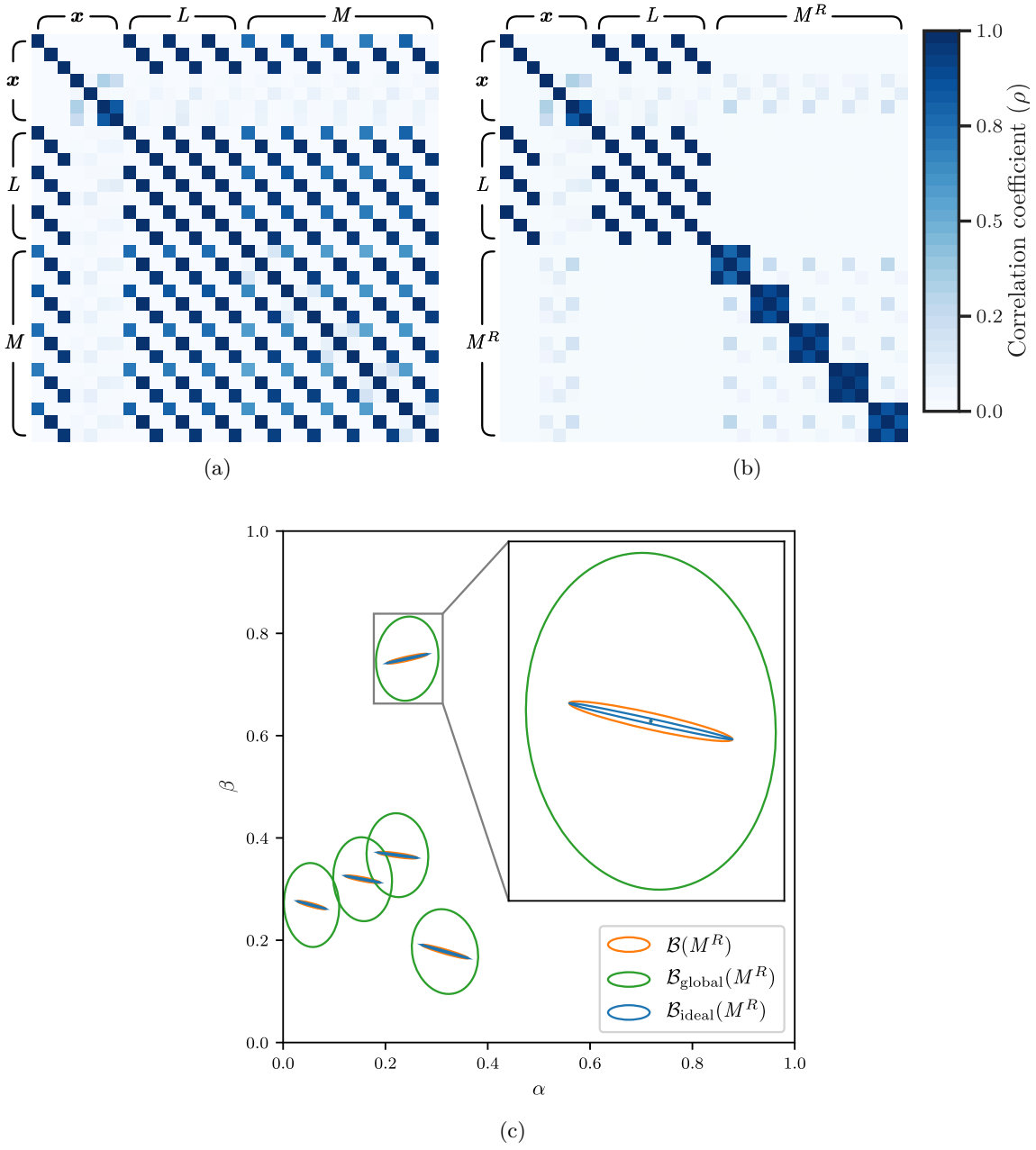


Figure 3.3: A visualisation of the absolute value of the Pearson correlation coefficient matrix (the darker, the more correlated) for the Gaussian belief distributions† $\mathcal{B}(\boldsymbol{x}, L, M)$ and (b) $\mathcal{B}(\boldsymbol{x}, L, M^R)$. In (b) the 3×3 block diagonals are from the individual 3-D environment surface points. (c) A comparison of the uncertainty in the beliefs over the environment surface measurements in a relative and global inertial reference frame (IRF). Our approximate belief, $\mathcal{B}(M^R)$, is compared to $\mathcal{B}_{\text{ideal}}(M^R)$ and $\mathcal{B}_{\text{global}}(M^R)$ (see the text for further details of the belief definitions). The ellipses outline the first standard deviations of the respective belief distributions. The data in this visualisation was taken from a practical stereo vision dataset.

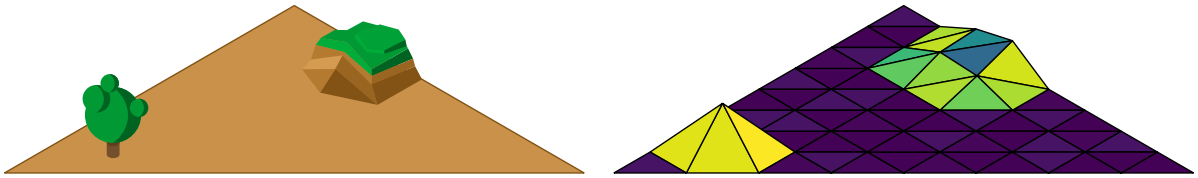
†We denote the current robot pose and landmark map in the global IRF, $\boldsymbol{x}$ and L respectively, and the environment surface points in the global IRF, M, and the relative IRF, M^R.

4 The Stochastic Triangular Mesh Map

Mere accumulation of observational evidence is not proof.

Sir Terry Pratchett, *The Hogfather*

Based on the highlighted drawbacks of the existing mapping techniques (Section 1.1.8), we propose a mapping technique that explicitly models the surface of the environment using a stochastic triangular mesh (STM); that is, a triangular mesh consisting of stochastic surface elements (surfels), forming a continuous representation of the surface of the environment (Figure 4.1).



(a) (b)

Figure 4.1: (a) A hypothetical environment within a triangular submap, and (b) a depiction of the corresponding stochastic triangular mesh (STM) map. Each triangular surface element (surfel) in the mesh is coloured according to its planar deviation—the deviation of the actual surface of the environment from the triangular planes (the lighter, the higher the deviation.)

Most of the existing explicit surface-mapping techniques operate under the approximation that the map *can* model the surface of the environment exactly. An STM map, on the other hand, accounts for the fact that the surface of the environment cannot be modelled exactly with deterministic map elements; instead, the surface of the environment is treated as a *stochastic process*. Additionally, even if it is possible to represent the environment exactly, the representation would be very wasteful; we rather summarise the surface information relevant to the application. In contrast to most mapping techniques, an STM map also captures the structure of the environment by enforcing continuity in the model. Although the principle is similar to Gaussian process mapping, we are able to update the model incrementally and in a scalable fashion.

Following the aforementioned submapping method (Chapter 3), we develop the STM map within a triangular submap. We partition the submap into a regular triangular grid by recursively subdividing the horizontal (α-β) plane into equisized triangular grid elements (Figure 4.2). The surface of the environment within each grid element—the region normal to the grid element—is modelled by the STM map using a surfel. The submap division is performed until a desired grid element size is achieved. Depending on the application, this could be simply chosen based on the dimensions and physical capability of the robot. In general, it would be more sensible to perform this partitioning in an adaptive manner, although we do not address this in the current implementation, but will discuss this in future work (see Chapter 6).

In the remainder of this chapter, we discuss how each surfel in an STM map models the surface of the environment. Subsequently, we discuss performing incremental Bayesian inference on

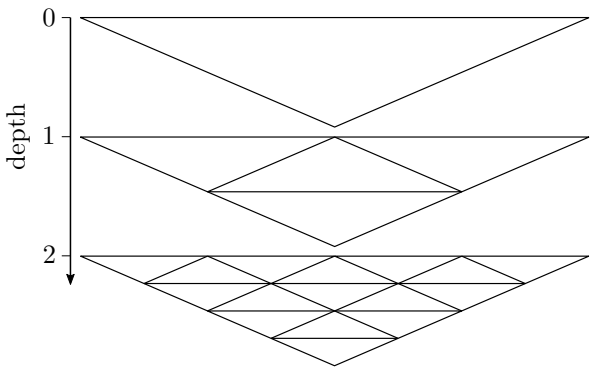


Figure 4.2: The recursive division of a triangular submap into equisized triangular grid elements.

an STM map (Section 4.2). It should also be noted that we develop the STM map within the relative inertial reference frame (IRF) of a submap, and consequently, all variables are assumed to be defined in the relative IRF. However, without loss of generality, this representation could also be applied in a global IRF framework by defining the IRF accordingly, which we demonstrate in Section 5.4.

4.1 Surface Element Model

An STM map consists of triangular surface elements (surfels), which together form a continuous representation of the surface of the environment within the submap. We assume that the surface of the environment within the submap can be represented adequately using a 2.5-D map. At any coordinate on the horizontal plane, the environment normal to this plane need only be described using a single height. Therefore, associated with each grid element is a triangular surfel that models the surface of the environment within the grid element—the region normal to the grid element.

If we initially consider a map with a single grid element, the maximum length along both the α- and β-axes within the grid element will be unity—Equation 3.2. The associated surfel models the surface of the environment within the grid element as a stochastic process—a mean plane and a homoscedastic[1] deviation from it, normal to the grid element (Figure 4.3). This simple model does not try to exactly represent the actual surface of the environment, but summarises the key aspects of the surface instead. More precisely, a point on the surface of the environment at some given coordinates (α, β) is modelled by the stochastic process

$$\begin{aligned} \gamma &= f(\alpha, \beta, \boldsymbol{h}) + \epsilon \\ &= \begin{bmatrix} 1-\alpha-\beta & \alpha & \beta \end{bmatrix} \boldsymbol{h} + \epsilon, \end{aligned} \tag{4.1}$$

where $f(\alpha, \beta, \boldsymbol{h})$ describes a point on the mean plane, and $\epsilon \sim \mathcal{N}_m(\epsilon; 0, \nu)$ models the deviation of the actual surface from the mean plane[2]. We refer to the variance of the stochastic process, ν, as the planar deviation. The mean plane is specified using the normal heights at each of the vertices of the grid element

$$\boldsymbol{h} = \begin{bmatrix} h_0 \\ h_\alpha \\ h_\beta \end{bmatrix}, \tag{4.2}$$

[1] A stochastic process is *homoscedastic* when the process variance is constant and finite.
[2] Note that for a Gaussian distribution, we denote the canonical form $\mathcal{N}_c(\cdot)$, and the moment form $\mathcal{N}_m(\cdot)$.

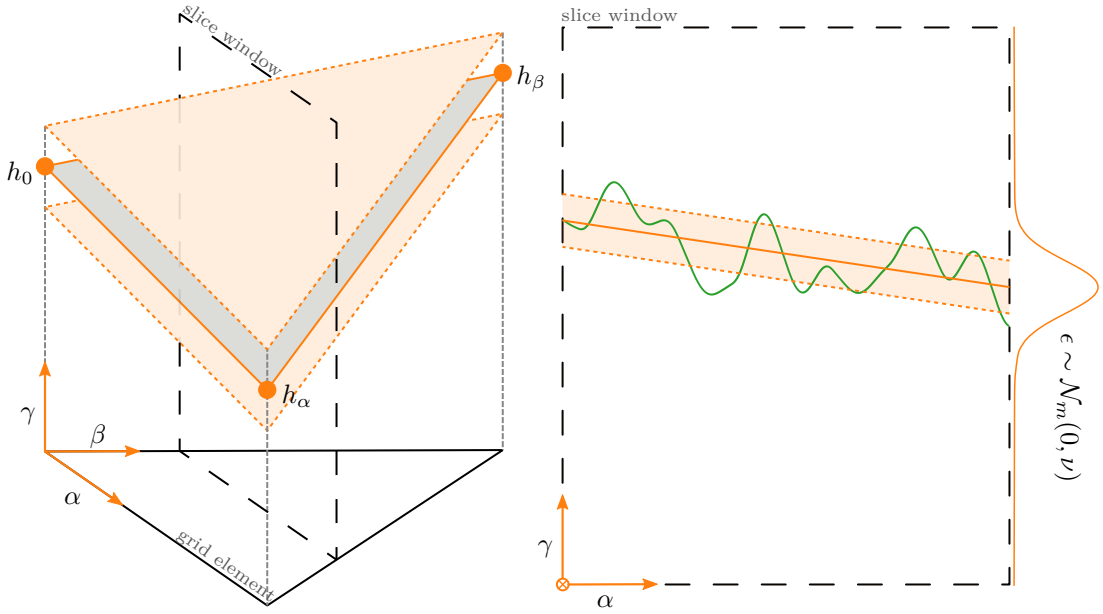


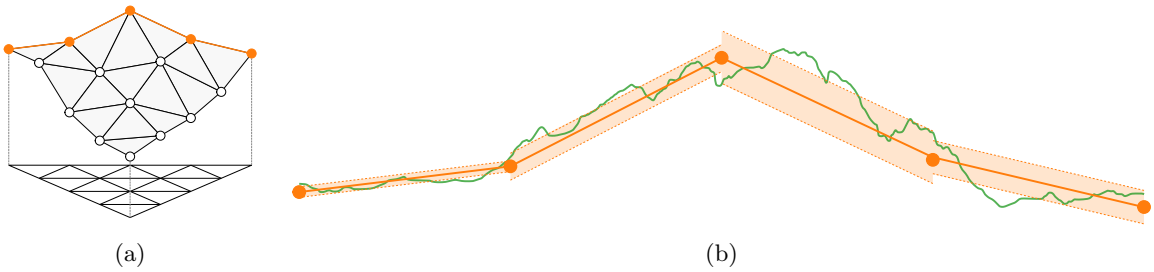
Figure 4.3: (left) The surface element (surfel) model, consisting of a mean plane—parameterised by the vertex heights $\boldsymbol{h} = [\, h_0,\, h_\alpha,\, h_\beta \,]^\intercal$—and a planar deviation ν. (right) A 2-D slice of the surfel (orange) of an underlying surface (green). The shaded region indicates the one standard deviation confidence interval. $\mathcal{N}_m(\cdot)$ denotes the moment form of a Gaussian distribution.

where h_0, h_α, and h_β are the heights at $(\alpha, \beta) = (0, 0)$, $(1, 0)$ and $(0, 1)$ respectively. Consequently, the surfel model is parameterised by $\boldsymbol{h}$ and ν, which we combine into a single parameter vector, $\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{h} & \nu \end{bmatrix}^\intercal$. This simple stochastic process allows us to account for a wide range of surfaces by essentially summarising the underlying environment within the grid element. It should be noted that we follow a Bayesian modelling approach, treating the map parameters as random variables. We specifically model the prior distributions over the surfels' heights as Gaussian distributed, and the planar deviation as inverse-gamma distributed. We expand on this when discussing performing inference on the model (Section 4.2).

In order to generalise this definition of the surfel model from a single to multiple grid elements within a submap, we normalise each grid element—scaling the maximum length of the α- and β-axes to unity, and shifting to the origin. This normalisation does not affect the values of the surfel parameters, $\boldsymbol{\theta}$, which are defined with respect to the γ-axis. Additionally, as this is a deterministic linear operation, the Gaussian measurement distributions in this normalised grid element remain exactly Gaussian distributed, which will be important when performing inference on the surfel model parameters (Section 4.2).

To create a continuous representation, the mean planes of surfels in contiguous grid elements are constrained to be continuous; that is, the heights at their shared vertices should be equal (Figure 4.4). However, we do not enforce any constraints on the planar deviations between surfels. The continuous surface in a submap formed by the mean plane in each surfel describes the mean mesh, which is parameterised by the ordered set of unique vertex heights, H. The stochastic deviation from this mean mesh is parameterised by the ordered set of planar deviations in each surfel, V. Both H and V fully parameterise an STM map, which can be combined into a single ordered set, Θ.

We have defined how the constituent surfels of an STM map model the environment; we now seek to determine the parameter values for an STM map based on measurements of the surface of the environment—in other words, to perform inference.



(a) (b)

Figure 4.4: A stochastic triangular mesh (STM) forms a continuous representation of the surface of the environment. We visualise (a) the mean mesh, and (b) a slice through the STM map (orange) of an underlying surface (green). The shaded region indicates the one standard deviation confidence interval in the respective surfels.

4.2 Model Inference

Based on the definition of an STM map, we now wish to infer the values of the map parameters, Θ, for some given noisy measurements of the surface of the environment, $Z = \mathtt{Z}$. We follow a Bayesian modelling approach, maintaining a belief distribution[3] over the map parameters. We would therefore ideally like to calculate the joint belief distribution over all the map parameters

$$\mathcal{B}(\Theta) = p(\Theta | Z = \mathtt{Z}). \tag{4.3}$$

However, for large maps, this would quickly become intractable to store, as the storage grows quadratically with the number of surfels in the map (for our choice of continuous parametric distributions). To circumvent this issue, we instead directly calculate the marginal belief for each surfel's parameters

$$\mathcal{B}(\boldsymbol{\theta}) = p(\boldsymbol{\theta} | Z = \mathtt{Z}), \tag{4.4}$$

for which storage grows linearly with each added surfel.

Next we will look at calculating each surfel's belief independently of the rest of the map (Section 4.2.1); this is equivalent to neglecting the continuity constraints between surfels. Building upon this, we expand to performing inference on the full model (Section 4.2.2). Finally, we summarise the inference algorithm for an STM map, and discuss the implementation thereof (Section 4.3).

4.2.1 Single Surfel Inference

We initially consider each surfel in isolation, and wish to calculate the belief distribution over a single surfel's parameters, $\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{h} & \nu \end{bmatrix}^\mathsf{T}$, from noisy measurements associated with the respective grid element[4]. We combine this approach to handle multiple surfels in Section 4.2.2, but it is useful to first consider this perspective before performing inference on the full model.

The Bayesian network representation of the surfel model we use is shown in Figure 4.5a, which

[3] A belief distribution is the posterior distribution over some states given all the relevant observations.

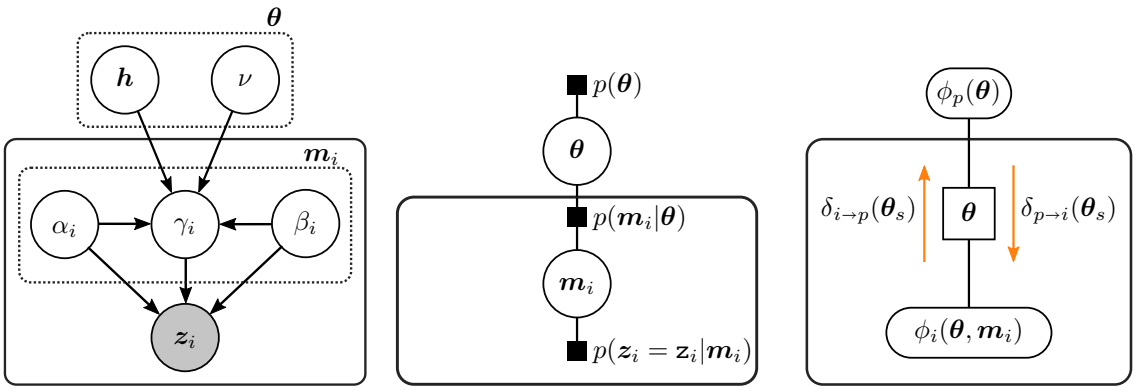
[4] We associate a measurement with a grid element based solely on whether the measurement's mean lies within the grid element.

equivalently represents the factorisation over the joint distribution

$$p(\boldsymbol{\theta}, M, Z = \mathtt{Z}) = p(\boldsymbol{\theta}) \underbrace{\prod_i p(\boldsymbol{z}_i = \mathtt{z}_i | \boldsymbol{m}_i) \overbrace{p(\gamma_i | \alpha_i, \beta_i, \boldsymbol{\theta})\, p(\alpha_i)\, p(\beta_i)}^{p(\boldsymbol{m}_i|\boldsymbol{\theta})},}_{p(M, Z=\mathtt{Z}|\boldsymbol{\theta})} \tag{4.5}$$

where $Z = (\boldsymbol{z}_i)$ is the sequence of noisy measurements corresponding to the points on the surface of the environment, $M = (\boldsymbol{m}_i)$, as mentioned in Section 3.2. This factorisation of the joint distribution models the generative process of measurements—the model parameters describe where actual points on the surface in the environment lie, and these points are observed through measurements of the environment. In the measurement mode, $p(\boldsymbol{z}_i = \mathtt{z}_i | \boldsymbol{m}_i)$, we assume that a measurement, $\boldsymbol{z}_i$, is only a priori dependent on the corresponding point on the actual surface of the environment, $\boldsymbol{m}_i$; this measurement distribution was derived in Section 3.2. In $p(\gamma_i | \alpha_i, \beta_i, \boldsymbol{\theta})$, we assume that the height of the surface point, γ_i, is only dependent on the grid element coordinates (α_i, β_i) and the surfel parameters, $\boldsymbol{\theta}$. This is based on the definition of the stochastic process in the surfel model—Equation 4.1.

From this Bayesian network we can construct a *factor graph* (Figure 4.5b). These factors can be grouped into clusters, resulting in a *cluster graph* representation (Figure 4.5c). We choose to group all the factors inside the plate into a cluster, and the factor outside the plate into a cluster. Both factor and cluster graphs can be used for inference, although we will focus on inference in cluster graphs.



(a) Bayesian network (b) Factor graph (c) Cluster graph

Figure 4.5: Different types of probabilistic graphical models representing the inference problem for our surfel model. The surfel model parameters, $\boldsymbol{\theta} = [\, \boldsymbol{h}\ \nu \,]^\intercal$, measurements, $\boldsymbol{z}_i$, and corresponding points on the surface of the environment, $\boldsymbol{m}_i = [\, \alpha_i\ \beta_i\ \gamma_i \,]^\intercal$, are all random variables. We use plate notation, where the solid rectangular plates indicate a repetition of the subgraph therein. (a) The dotted rectangle denotes an expanded vector, and the shaded circle indicates an observed (or given) variable. (b) Factors are indicated with ■. (c) Cluster potentials and messages are denoted $\phi(\cdot)$ and $\delta(\cdot)$ respectively, and the sepset between clusters is indicated with ⊡.

As a cluster graph is the result of grouping factors from the joint distribution, we can equivalently represent the joint distribution as the product over all the cluster potentials

$$p(\boldsymbol{\theta}, M, Z = \mathtt{Z}) = \phi_p(\boldsymbol{\theta}) \prod_i \phi_i(\boldsymbol{\theta}, \boldsymbol{m}_i), \tag{4.6}$$

where we refer to $\phi_p(\boldsymbol{\theta})$ as the *prior cluster potential* and each $\phi_i(\boldsymbol{\theta}, \boldsymbol{m}_i)$ as a *likelihood cluster potential*. We wish to infer the surfel belief distribution for some given measurements, $Z = \mathtt{Z}$.

Using Bayes' theorem, we can express this belief in terms of the joint distribution

$$\begin{aligned}\mathcal{B}(\boldsymbol{\theta}) &= p(\boldsymbol{\theta}|Z=\mathtt{Z}) \\ &\propto \int_M p(\boldsymbol{\theta}, M, Z=\mathtt{Z})\,\mathrm{d}M \\ &\propto \phi_p(\boldsymbol{\theta}) \prod_i \int_{\boldsymbol{m}_i} \phi_i(\boldsymbol{\theta}, \boldsymbol{m}_i)\,\mathrm{d}\boldsymbol{m}_i.\end{aligned} \tag{4.7}$$

An equivalent method of calculating this belief is by using message passing. For inference in cluster graphs, the outgoing and incoming messages for each cluster needs to be calculated. According to the integral-product algorithm, the outgoing message from the i-th likelihood cluster is defined as

$$\delta_{i\to p}(\boldsymbol{\theta}) = \int_{\boldsymbol{m}_i} \phi_i(\boldsymbol{\theta}, \boldsymbol{m}_i)\,\mathrm{d}\boldsymbol{m}_i. \tag{4.8}$$

Similarly, the incoming message to the i-th likelihood cluster is defined as

$$\delta_{p\to i}(\boldsymbol{\theta}) = \phi_p(\boldsymbol{\theta}) \prod_{j\neq i} \delta_{j\to p}(\boldsymbol{\theta}). \tag{4.9}$$

By comparing Equation 4.7 with Equations 4.8 and 4.9, we can see that the product of these messages equivalently specifies the surfel belief

$$\mathcal{B}(\boldsymbol{\theta}) \propto \delta_{i\to p}(\boldsymbol{\theta})\delta_{p\to i}(\boldsymbol{\theta}). \tag{4.10}$$

However, there is no convenient closed-form solution to either of the messages—and consequently the surfel belief. This is due to nonlinear relationships between the random variables in the likelihood cluster potentials. In order to find a tractable solution to the surfel belief, we perform approximate inference using variational message passing (VMP) [58].

In order to calculate the approximate surfel belief,

$$\tilde{\mathcal{B}}(\boldsymbol{\theta}) \propto \tilde{\delta}_{i\to p}(\boldsymbol{\theta})\tilde{\delta}_{p\to i}(\boldsymbol{\theta}), \tag{4.11}$$

the approximate messages, $\tilde{\delta}_{i\to p}(\boldsymbol{\theta})$ and $\tilde{\delta}_{p\to i}(\boldsymbol{\theta})$, are calculated by replacing $\phi_i(\boldsymbol{\theta}, \boldsymbol{m}_i)$ by an approximating likelihood cluster potential, $\tilde{\phi}_i(\boldsymbol{\theta}, \boldsymbol{m}_i)$, in Equations 4.8 and 4.9. To calculate each approximate likelihood cluster potential, VMP performs a local information projection

$$\tilde{\phi}_i(\boldsymbol{\theta}, \boldsymbol{m}_i) = \underset{\tilde{\phi}_i\in\mathcal{F}}{\arg\min}\, D_{\mathrm{KL}}\left(\tilde{\phi}_i\tilde{\delta}_{p\to i} \,\middle\|\, \phi_i\tilde{\delta}_{p\to i}\right), \tag{4.12}$$

where $\mathcal{F}$ is a family of approximating distributions (we omit the arguments of the functions to declutter the notation). It should be noted that this is an iterative process, as we must perform this local projection for each likelihood cluster potential. To make this local projection tractable, we also make the simplifying assumption that each approximating likelihood cluster potential is of the factorised form[5]

$$\tilde{\phi}_i(\boldsymbol{\theta}, \boldsymbol{m}_i) = \tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i)\tilde{\phi}_i(\nu). \tag{4.13}$$

This factorisation over disjoint sets of the random variables is a commonly-used approximation referred as the structured mean-field approximation [70]. We choose this structured factorisation in particular, as it more accurately represents the exact distribution compared to a full factorisation. In contrast to other structured factorisation combinations, it groups variables from the same

[5]From this point on, for notational brevity we loosely use the arguments to functions as unique identifiers, namely $f(x) \triangleq f_x(x)$ and $f(y) \not\triangleq f(x)$.

distribution family (as we will soon show). We also assume the prior cluster potential to be of a similar factorised form,

$$\phi_p(\boldsymbol{\theta}) = \phi_p(\boldsymbol{h})\phi_p(\nu), \tag{4.14}$$

therefore the approximate messages and surfel belief will be similarly factorised.

In order to perform the minimisation in Equation 4.12, we first need to specify the family of approximating distributions for the factors of $\tilde{\phi}_i(\boldsymbol{\theta}, \boldsymbol{m}_i)$. For the factor over $\boldsymbol{h}$ and $\boldsymbol{m}_i$, we choose the Gaussian distribution

$$\tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i) = \mathcal{N}_c(\begin{bmatrix}\boldsymbol{h} & \boldsymbol{m}_i\end{bmatrix}^\intercal; \boldsymbol{\xi}_i, \boldsymbol{\Omega}_i), \tag{4.15}$$

where $\boldsymbol{\xi}_i$ is the information vector and $\boldsymbol{\Omega}_i$ is the information matrix of the Gaussian distribution in canonical form. For the factor over ν, we choose the inverse-gamma distribution

$$\tilde{\phi}_i(\nu) = \Gamma^{-1}(\nu; a_i, b_i), \tag{4.16}$$

where a_i is the shape parameter and b_i the scale parameter of the inverse-gamma distribution. The resulting product of these distributions—Equation 4.13—should not be confused with a normal-inverse-gamma distribution. In our case, this is simply the product of these two distributions without any conditional dependencies.

If we had already calculated the approximate likelihood cluster potential, $\tilde{\phi}_i(\boldsymbol{\theta}, \boldsymbol{m}_i)$, we could calculate the outgoing message using Equation 4.8, which would result in the distribution

$$\tilde{\delta}_{i\to p}(\boldsymbol{\theta}) = \underbrace{\mathcal{N}_c(\boldsymbol{h}; \boldsymbol{\xi}_{i\to p}, \boldsymbol{\Omega}_{i\to p})}_{\tilde{\delta}_{i\to p}(\boldsymbol{h})} \underbrace{\Gamma^{-1}(\nu; a_{i\to p} - 1, b_{i\to p})}_{\tilde{\delta}_{i\to p}(\nu)}. \tag{4.17}$$

We assume that the prior cluster potential is also similarly distributed, as

$$\phi_p(\boldsymbol{\theta}) = \underbrace{\mathcal{N}_c(\boldsymbol{h}; \boldsymbol{\xi}_p, \boldsymbol{\Omega}_p)}_{\phi_p(\boldsymbol{h})} \underbrace{\Gamma^{-1}(\nu; a_p, b_p)}_{\phi_p(\nu)}. \tag{4.18}$$

Both the Gaussian and inverse-gamma families of distributions are part of the exponential family [52]. These have the convenient property that the product and division of distributions in the family also results in a distribution in the family [57] (up to a proportionality constant), which amounts to the addition and subtraction of their natural parameters respectively. We therefore can calculate the approximate incoming message to a cluster, which will be distributed similarly to its constituents,

$$\tilde{\delta}_{p\to i}(\boldsymbol{\theta}) = \underbrace{\mathcal{N}_c(\boldsymbol{h}; \boldsymbol{\xi}_{p\to i}, \boldsymbol{\Omega}_{p\to i})}_{\tilde{\delta}_{p\to i}(\boldsymbol{h})} \underbrace{\Gamma^{-1}(\nu; a_{p\to i}, b_{p\to i})}_{\tilde{\delta}_{p\to i}(\nu)}. \tag{4.19}$$

According to Equation 4.9, the natural parameters of $\tilde{\delta}_{p\to i}(\boldsymbol{h})$ are calculated as

$$\boldsymbol{\Omega}_{p\to i} = \boldsymbol{\Omega}_p + \sum_{j\neq i} \boldsymbol{\Omega}_{j\to p} \quad \text{and} \quad \boldsymbol{\xi}_{p\to i} = \boldsymbol{\xi}_p + \sum_{j\neq i} \boldsymbol{\xi}_{j\to p}, \tag{4.20}$$

and the natural parameters of $\tilde{\delta}_{p\to i}(\nu)$ are calculated as

$$a_{p\to i} = a_p + \sum_{j\neq i} a_{j\to p} \quad \text{and} \quad b_{p\to i} = b_p + \sum_{j\neq i} b_{j\to p}. \tag{4.21}$$

The same is true for the approximate surfel belief,

$$\tilde{\mathcal{B}}(\boldsymbol{\theta}) = \underbrace{\mathcal{N}_c(\boldsymbol{h}; \boldsymbol{\xi}_h, \boldsymbol{\Omega}_h)}_{\tilde{\mathcal{B}}(\boldsymbol{h})} \underbrace{\Gamma^{-1}(\nu; a_\nu, b_\nu)}_{\tilde{\mathcal{B}}(\nu)}. \tag{4.22}$$

According to Equation 4.11, the natural parameters of $\tilde{\mathcal{B}}(\boldsymbol{h})$ are calculated as

$$\boldsymbol{\Omega}_h = \boldsymbol{\Omega}_{i\to p} + \boldsymbol{\Omega}_{p\to i} \quad \text{and} \quad \boldsymbol{\xi}_h = \boldsymbol{\xi}_{i\to p} + \boldsymbol{\xi}_{p\to i}, \tag{4.23}$$

and the natural parameters of $\tilde{\mathcal{B}}(\nu)$ are calculated as

$$a_\nu = a_{i\to p} + a_{p\to i} \quad \text{and} \quad b_\nu = b_{i\to p} + b_{p\to i}. \tag{4.24}$$

Performing VMP with the structured mean-field approximation therefore allows us to perform tractable inference to obtain an approximation of the surfel belief. Up to this point, we have assumed that we have the approximate likelihood cluster potentials, $\tilde{\phi}_i(\boldsymbol{\theta}, \boldsymbol{m}_i)$. We now discuss exactly how to calculate each of the factors of approximate likelihood cluster potentials and, consequently, the outgoing message, $\tilde{\delta}_{i\to p}(\boldsymbol{\theta})$.

Updating $\tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i)$

The iterative update of $\tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i)$ that would perform the local information projection of Equation 4.12 can be calculated according to the VMP algorithm [58] as

$$\begin{aligned} &\log \tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i) + \log \tilde{\delta}_{p\to i}(\boldsymbol{h}) \\ &\propto \left\langle \log \phi_i(\boldsymbol{\theta}, \boldsymbol{m}_i, \boldsymbol{z}_i = \mathbf{z}_i) \tilde{\delta}_{p\to i}(\boldsymbol{\theta}) \right\rangle_{\tilde{\mathcal{B}}(\nu)}, \end{aligned} \tag{4.25}$$

where $\langle \cdot \rangle$ denotes the expectation operation. In order to arrive at a solution to $\tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i)$, it is convenient to first describe the process of calculating the approximate belief,

$$\tilde{\mathcal{B}}(\boldsymbol{h}, \boldsymbol{m}_i) \propto \tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i) \tilde{\delta}_{p\to i}(\boldsymbol{h}), \tag{4.26}$$

and then to divide out the message $\tilde{\delta}_{p\to i}(\boldsymbol{h})$ to recover the desired approximation to the cluster potential $\tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i)$. Rearranging Equation 4.25 in terms of $\tilde{\mathcal{B}}(\boldsymbol{h}, \boldsymbol{m}_i)$ we can calculate the expectation

$$\begin{aligned} \log \tilde{\mathcal{B}}(\boldsymbol{h}, \boldsymbol{m}_i) &\propto \left\langle \log \left(\phi_i(\boldsymbol{\theta}, \boldsymbol{m}_i) \tilde{\delta}_{p\to i}(\boldsymbol{\theta}) \right) \right\rangle_{\tilde{\mathcal{B}}(\nu)} \\ &= \log p(\gamma_i | \alpha_i, \beta_i, \boldsymbol{h}, \nu = \nu_{\mathcal{B}}) \\ &\quad + \log p(\boldsymbol{z}_i = \mathbf{z}_i | \boldsymbol{m}_i) \\ &\quad + \log \left(p(\alpha_i) p(\beta_i) \tilde{\delta}_{p\to i}(\boldsymbol{h}) \right), \end{aligned} \tag{4.27}$$

where

$$\nu_{\mathcal{B}} = \left\langle \nu^{-1} \right\rangle^{-1}_{\tilde{\mathcal{B}}(\nu)} = \frac{b_\nu}{a_\nu}. \tag{4.28}$$

In order to find an analytical expression for $\tilde{\mathcal{B}}(\boldsymbol{h}, \boldsymbol{m}_i)$, we consider each of the constituent distributions in Equation 4.27. As mentioned in Section 3.2, the *measurement distribution* is Gaussian distributed, which can be rearranged such that the mean of the distribution is the observation

$$\begin{aligned} p(\boldsymbol{z}_i = \mathbf{z}_i | \boldsymbol{m}_i) &= \mathcal{N}_m(\boldsymbol{z}_i = \mathbf{z}_i; \boldsymbol{m}_i, \boldsymbol{\Sigma}_{z_i}) \\ &= \mathcal{N}_m(\boldsymbol{m}_i; \mathbf{z}_i, \boldsymbol{\Sigma}_{z_i}). \end{aligned} \tag{4.29}$$

According to the surfel model in Section 4.1, the *likelihood distribution* is Gaussian distributed

$$p(\gamma_i | \alpha_i, \beta_i, \boldsymbol{h}, \nu_{\mathcal{B}}) = \mathcal{N}_m(\gamma_i; f(\boldsymbol{h}, \alpha_i, \beta_i), \nu_{\mathcal{B}}), \tag{4.30}$$

where $f(\cdot)$ is the mean of the stochastic process as defined in Equation 4.1. We place an uninformative and independently distributed Gaussian prior over α_i and β_i, and combine it with $\tilde{\delta}_{p\to i}(\boldsymbol{h})$ to form the jointly Gaussian distributed *context distribution*

$$p(\alpha_i)p(\beta_i)\tilde{\delta}_{p\to i}(\boldsymbol{h}) = \mathcal{N}_m(\begin{bmatrix}\boldsymbol{h} & \alpha_i & \beta_i\end{bmatrix}^\intercal; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \tag{4.31}$$

with moments calculated as

$$\boldsymbol{\mu}_c = \begin{bmatrix}\boldsymbol{\mu}_{p\to i} \\ \mu_{\alpha_i} \\ \mu_{\beta_i}\end{bmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_c = \text{diag}(\boldsymbol{\Sigma}_{p\to i}, \sigma^2_{\alpha_i}, \sigma^2_{\beta_i}), \tag{4.32}$$

where $\boldsymbol{\Sigma}_{p\to i} = \boldsymbol{\Omega}^{-1}_{p\to i}$, $\boldsymbol{\mu}_{p\to i} = \boldsymbol{\Sigma}_{p\to i}\boldsymbol{\xi}_{p\to i}$, and the $\text{diag}(\cdot)$ operator creates a block diagonal matrix of its arguments. Most notably, all distributions except the likelihood distribution are exactly Gaussian distributed over $\boldsymbol{h}$ and $\boldsymbol{m}_i$. This dissimilarity in likelihood distribution is due to the nonlinear relationships introduced by $f(\cdot)$. To ensure that the product of all the distributions is tractable, we also wish to approximate the likelihood distribution as a Gaussian distribution. A common approach to achieving this is by linearising; we therefore use a linear Taylor series approximation,

$$f(\boldsymbol{x}) \approx f(\bar{\boldsymbol{x}}) + \boldsymbol{F}(\boldsymbol{x} - \bar{\boldsymbol{x}}), \tag{4.33}$$

where $\boldsymbol{x} = \begin{bmatrix}\boldsymbol{h} & \alpha_i & \beta_i\end{bmatrix}^\intercal$, and $\boldsymbol{F}$ is the Jacobian matrix of $f(\boldsymbol{x})$ evaluated at the linearisation point $\boldsymbol{\mu}_c$:

$$\boldsymbol{F} = \left.\frac{\delta f}{\delta \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{\mu}_c} = \left.\begin{bmatrix}1 - \alpha_i - \beta_i & \alpha_i & \beta_i & h_\alpha - h_0 & h_\beta - h_0\end{bmatrix}\right|_{\boldsymbol{\mu}_c}. \tag{4.34}$$

Using this we can calculate the product of the likelihood and context distributions,

$$\begin{aligned} &p(\gamma_i|\alpha_i, \beta_i, \boldsymbol{h}, \nu_{\mathcal{B}})p(\alpha_i)p(\beta_i)\tilde{\delta}_{p\to i}(\boldsymbol{h}) \\ &\propto \mathcal{N}_c(\begin{bmatrix}\boldsymbol{h} & \boldsymbol{m}_i\end{bmatrix}^\intercal; \bar{\boldsymbol{\xi}}, \bar{\boldsymbol{\Omega}}) \end{aligned}, \tag{4.35}$$

with natural parameters calculated as

$$\bar{\boldsymbol{\xi}} = \bar{\boldsymbol{\Omega}}\begin{bmatrix}\boldsymbol{\mu}_c \\ f(\boldsymbol{\mu}_c)\end{bmatrix} \quad \text{and} \quad \bar{\boldsymbol{\Omega}} = \begin{bmatrix}\boldsymbol{\Sigma}_c & (\boldsymbol{F}\boldsymbol{\Sigma}_c)^\intercal \\ \boldsymbol{F}\boldsymbol{\Sigma}_c & \boldsymbol{F}\boldsymbol{\Sigma}_c\boldsymbol{F}^\intercal + \nu_{\mathcal{B}}\end{bmatrix}^{-1}. \tag{4.36}$$

The normalised product of the prediction distribution and the measurement distribution results in the desired belief

$$\tilde{\mathcal{B}}(\boldsymbol{h}, \boldsymbol{m}_i) = \mathcal{N}_c(\begin{bmatrix}\boldsymbol{h} & \boldsymbol{m}_i\end{bmatrix}^\intercal; \boldsymbol{\xi}, \boldsymbol{\Omega}), \tag{4.37}$$

with natural parameters calculated as

$$\begin{aligned} \boldsymbol{\xi} &= \bar{\boldsymbol{\xi}} + \boldsymbol{S}^\intercal\boldsymbol{\Sigma}^{-1}_{z_i}\mathbf{z}_i \\ &= \begin{bmatrix}\boldsymbol{\xi}_h \\ \boldsymbol{\xi}_m\end{bmatrix} \end{aligned} \quad \text{and} \quad \begin{aligned} \boldsymbol{\Omega} &= \bar{\boldsymbol{\Omega}} + \boldsymbol{S}^\intercal\boldsymbol{\Sigma}^{-1}_{z_i}\boldsymbol{S} \\ &= \begin{bmatrix}\boldsymbol{\Omega}_{hh} & \boldsymbol{\Omega}_{hm} \\ \boldsymbol{\Omega}_{mh} & \boldsymbol{\Omega}_{mm}\end{bmatrix}, \end{aligned} \tag{4.38}$$

where $\boldsymbol{S} = \begin{bmatrix}\mathbf{0}_{3\text{x}3} & \boldsymbol{I}_{3\text{x}3}\end{bmatrix}$ is a selection matrix. Notably, the above method is equivalent to an extended information filter (EIF) with a nonlinear state transition model and a linear measurement model [50, ch. 3.5.4]. Finally, we can calculate $\tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i)$ by dividing out $\tilde{\delta}_{p\to i}(\boldsymbol{h})$ from $\tilde{\mathcal{B}}(\boldsymbol{h}, \boldsymbol{m}_i)$ (Equation 4.26), which results in

$$\tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i) = \mathcal{N}_c(\begin{bmatrix}\boldsymbol{h} & \boldsymbol{m}_i\end{bmatrix}^\intercal; \boldsymbol{\xi}_i, \boldsymbol{\Omega}_i), \tag{4.39}$$

with natural parameters calculated as

$$\boldsymbol{\xi}_i = \begin{bmatrix} \boldsymbol{\xi}_h - \boldsymbol{\xi}_{p\to i} \\ \boldsymbol{\xi}_m \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Omega}_i = \begin{bmatrix} \boldsymbol{\Omega}_{hh} - \boldsymbol{\Omega}_{p\to i} & \boldsymbol{\Omega}_{hm} \\ \boldsymbol{\Omega}_{mh} & \boldsymbol{\Omega}_{mm} \end{bmatrix}. \tag{4.40}$$

From this result, we can now calculate a part of the outgoing message, $\tilde{\delta}_{i\to p}(\boldsymbol{h})$, by marginalising out $\boldsymbol{m}_i$. Therefore, using Equation 4.8, we can calculate the natural parameters of $\tilde{\delta}_{i\to p}(\boldsymbol{h})$ (Equation 4.17) as

$$\boldsymbol{\Omega}_{i\to p} = \boldsymbol{\Omega}_{hh} - \boldsymbol{\Omega}_{p\to j} - \boldsymbol{\Omega}_{hm}\boldsymbol{\Omega}_{mm}^{-1}\boldsymbol{\Omega}_{mh} \tag{4.41}$$

and

$$\boldsymbol{\xi}_{i\to p} = \boldsymbol{\xi}_h - \boldsymbol{\xi}_{p\to j} - \boldsymbol{\Omega}_{hm}\boldsymbol{\Omega}_{mm}^{-1}\boldsymbol{\xi}_m. \tag{4.42}$$

We have now discussed how to calculate the likelihood cluster potential $\tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i)$ and the outgoing message $\tilde{\delta}_{i\to p}(\boldsymbol{h})$. Next, we focus on the likelihood cluster potential and outgoing message over the planar deviation.

Updating $\tilde{\phi}_i(\nu)$

Similarly to calculating $\tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i)$, to perform the local information projection of Equation 4.12 with respect to $\tilde{\phi}_i(\nu)$, the VMP algorithm [58] defines the iterative update as

$$\log \tilde{\phi}_i(\nu) + \log \tilde{\delta}_{p\to i}(\nu) \propto \left\langle \log \phi_i(\boldsymbol{\theta}, \boldsymbol{m}_i)\tilde{\delta}_{p\to i}(\boldsymbol{\theta}) \right\rangle_{\tilde{\mathcal{B}}(\boldsymbol{h},\boldsymbol{m}_i)}. \tag{4.43}$$

The marginalisation to calculate the outgoing message, $\tilde{\delta}_{i\to p}(\boldsymbol{\theta})$—Equation 4.8—will have no effect on the outgoing message, $\tilde{\delta}_{i\to p}(\nu)$, due to our choice of factorisation. Therefore, solving for the cluster potential is equivalent to solving for the outgoing message—$\tilde{\delta}_{i\to p}(\nu) = \tilde{\phi}_i(\nu)$—which we can directly calculate as

$$\begin{aligned} \log \tilde{\delta}_{i\to p}(\nu) &\propto \left\langle \log p(\gamma_i|\alpha_i, \beta_i, \boldsymbol{h}, \nu) \right\rangle_{\tilde{\mathcal{B}}(\boldsymbol{h},\boldsymbol{m}_i)} \\ &\propto -a_{i\to p}\log\nu - \frac{b_{i\to p}}{\nu}, \end{aligned} \tag{4.44}$$

where $\tilde{\mathcal{B}}(\boldsymbol{h}, \boldsymbol{m}_i)$ is calculated from Equation 4.27. The resulting approximate potential matches the form of an inverse-gamma distribution, with shape and scale parameters

$$a_{i\to p} = \frac{1}{2} \quad \text{and} \quad b_{i\to p} = \frac{1}{2}\left\langle (\gamma_i - f(\alpha_i, \beta_i, \boldsymbol{h}))^2 \right\rangle_{\tilde{\mathcal{B}}(\boldsymbol{h},\boldsymbol{m}_i)}, \tag{4.45}$$

where $f(\cdot)$ is defined by Equation 4.1. We again use the linear Taylor series approximation to calculate the expectation[6], where we linearise around the mean of $\tilde{\mathcal{B}}(\boldsymbol{h}, \boldsymbol{m}_i)$. This results in the expectation

$$b_{i\to p} = \frac{1}{2}\boldsymbol{F}_{\text{aug}}\boldsymbol{\Sigma}\boldsymbol{F}_{\text{aug}}^{\intercal} + \frac{1}{2}(\mu_\gamma - f(\mu_\alpha, \mu_\beta, \boldsymbol{\mu}_h))^2, \tag{4.46}$$

where $\boldsymbol{F}_{\text{aug}} = \begin{bmatrix} \boldsymbol{F} & 1 \end{bmatrix}$—with $\boldsymbol{F}$ defined by Equation 4.34—and $\boldsymbol{\Sigma} = \boldsymbol{\Omega}^{-1}$ and $\boldsymbol{\mu} = \boldsymbol{\Sigma}\boldsymbol{\xi}$ are the moments of the distribution in Equation 4.37.

We have now discussed how to perform inference on each surfel in isolation. Before discussing performing inference on the full model, we briefly look at the performance of the proposed inference algorithm.

[6]This expectation could be calculated analytically, because $f(\cdot)$ is a multilinear function and $\tilde{\mathcal{B}}(\boldsymbol{h}, \boldsymbol{m}_i)$ is Gaussian distributed. However, we found that, in rare cases, it was problematic for the initially uncertain stages of message passing, and consequently hindered convergence. In contrast, using the Taylor series approximation was found to be a more robust approach. Additionally, the Taylor series is computationally cheaper to compute.

Inference Performance

To illustrate the performance of the proposed inference algorithm, we compare our approximation of the surfel model belief against Markov chain Monte Carlo (MCMC) samples of the exact belief distribution for a 2-D simulation—using the Metropolis-Hastings algorithm to generate samples. Figure 4.6 shows representative cases for a stereo camera and a LiDAR sensor. We see that our algorithm accurately matches the marginal distributions from MCMC. In cases where the measurement uncertainty is unreasonably high[7], our algorithm results differ noticeably from the exact belief; this is due to errors induced through linearisation. However, when more measurements are added, this effect is mitigated. We also did not encounter any issues with this when using either simulation or practical data—as we will see in Chapter 5.

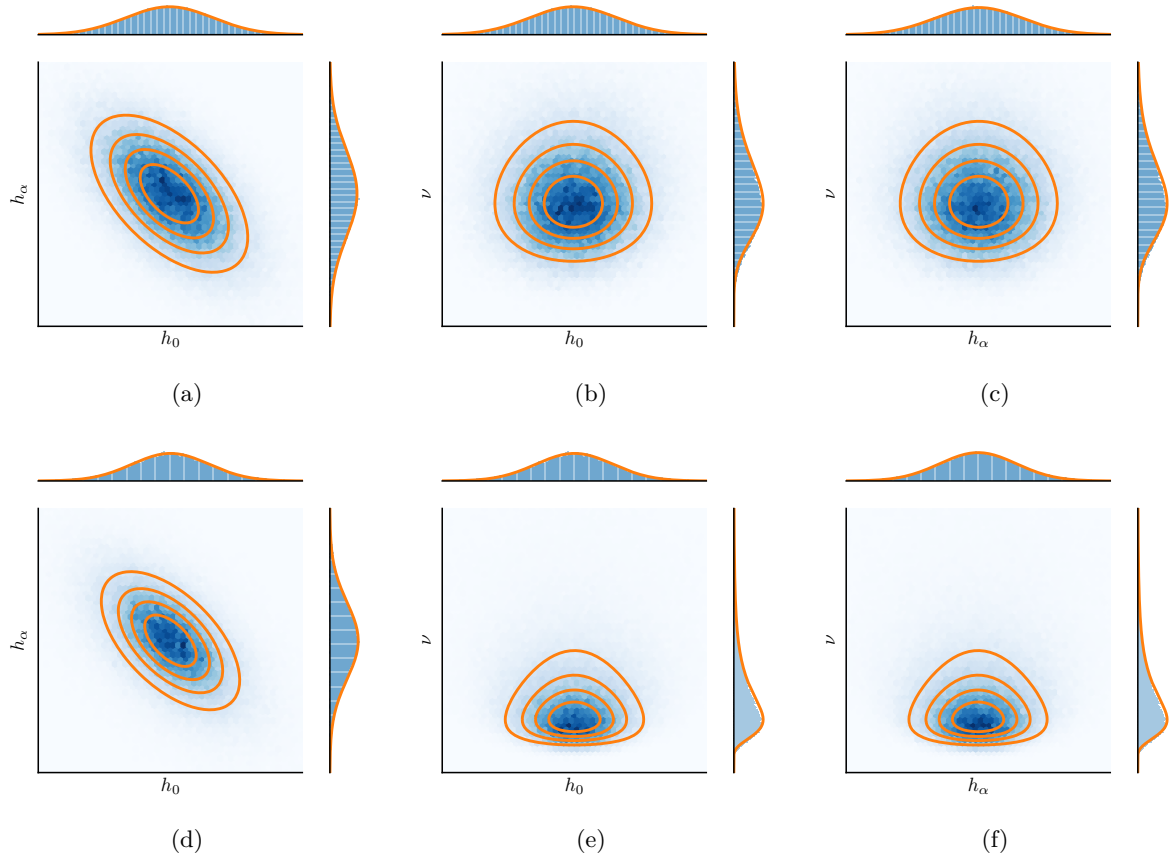


Figure 4.6: The accuracy of our inference algorithm in calculating an approximate surfel model belief (orange), compared against MCMC samples of the exact belief distribution (blue) for a 2-D simulation. We emulate two different sensor modalities: (a-c) a stereo camera by using 100 measurements with high range and low bearing uncertainty, and (d-f) a LiDAR sensor by using 10 measurements with low measurement uncertainty. Shown for these two cases are the three 2-D projections of the 3-D belief distribution over h_0, h_α, and ν, and the marginal distributions for each axis. Note that we do not show the axis ticks, as we are only concerned with the *relative* difference between the exact belief and our approximation.

We have shown that this method of approximation can accurately represent the surfel belief; however, this was only for the case where each surfel is updated in isolation. We will now consider inference on the full model, utilising the approach we have developed here.

[7]The measurement uncertainty relative to the size of the grid element increases with the grid division depth. Therefore, too fine a grid division could result in a measurement distribution that spans several grid elements.

4.2.2 Inference on the Full Model

As opposed to calculating the joint belief distribution over all the surfel parameters in an STM map, we instead wish to calculate the marginal belief distributions over each surfel's parameters. However, as there are shared height parameters between contiguous surfels, we cannot naively use exactly the same approach as we did previously to perform inference on the full model. It should be noted that, although we will consider the joint distribution over all the map parameters, we do not calculate this expressly, as the marginal surfel beliefs are our desired end result.

We can reformulate the inference problem over all the map parameters by utilising our previous approach of performing independent surfel inference. For a given surfel, s, we assume that measurements within the associated grid element, Z_s, and the points on the surface of the environment, M_s, are conditionally independent of all other measurements, $Z \setminus Z_s$, and all other surface points, $M \setminus M_s$, given the parameters of the surfel, $\boldsymbol{\theta}_s$. Following this assumption, and using Bayes' theorem, we can factorise the full joint distribution

$$\begin{aligned} p(\Theta, M, Z = \mathtt{Z}) &= p(\Theta)p(M, Z = \mathtt{Z}|\Theta) \\ &= p(\Theta) \prod_{s \in S} p(M_s, Z_s = \mathtt{Z}_s|\boldsymbol{\theta}_s), \end{aligned} \tag{4.47}$$

where S is the set of all surfels in an STM map. Each conditional distribution, $p(M_s, Z_s = \mathtt{Z}_s|\boldsymbol{\theta}_s)$, is described by the same model we used previously, namely

$$p(M_s, Z_s = \mathtt{Z}_s|\boldsymbol{\theta}_s) = \prod_i \phi_i(\boldsymbol{\theta}_s, \boldsymbol{m}_i), \tag{4.48}$$

where we again refer to $\phi_i(\boldsymbol{\theta}_s, \boldsymbol{m}_i)$ as the likelihood cluster potential—as defined in Equations 4.5 and 4.6. We factorise the prior distribution into a product of prior cluster potentials,

$$p(\Theta) \propto \prod_{s \in S} \phi_p(\boldsymbol{\theta}_s). \tag{4.49}$$

As with our previous approach, each prior cluster potential is distributed according to Equation 4.18—namely a Gaussian distribution over the vertex heights and an inverse-gamma distribution over the planar deviation. We will examine the effect that this factorisation has on the resulting surfel beliefs, and how the parameter choices (specifically for the height prior distribution) could encode additional structure in the environment (Section 5.1). Combining the prior factorisation with Equation 4.48, we obtain

$$p(\Theta, M, Z = \mathtt{Z}) = \prod_{s \in S} \phi_s(\boldsymbol{\theta}_s, M_s), \tag{4.50}$$

where the surfel cluster potential, $\phi_s(\boldsymbol{\theta}_s, M_s, Z_s)$, is calculated according to

$$\begin{aligned} \phi_s(\boldsymbol{\theta}_s, M_s) &= \phi_p(\boldsymbol{\theta}_s)p(M_s, Z_s = \mathtt{Z}_s|\boldsymbol{\theta}_s) \\ &= \phi_p(\boldsymbol{\theta}_s) \prod_i \phi_i(\boldsymbol{\theta}_s, \boldsymbol{m}_i). \end{aligned} \tag{4.51}$$

To calculate a surfel belief for some given measurements, $Z = \mathtt{Z}$, we must marginalise out $\setminus \boldsymbol{\theta}_s$—all variables except for $\boldsymbol{\theta}_s$—from the full joint distribution,

$$\begin{aligned} \mathcal{B}(\boldsymbol{\theta}_s) &\propto \int_{\setminus \boldsymbol{\theta}_s} p(\Theta, M, Z = \mathtt{Z}) \, \mathrm{d} \setminus \boldsymbol{\theta}_s \\ &= \int_{\setminus \boldsymbol{\theta}_s} \prod_{s \in S} \phi_s(\boldsymbol{\theta}_s, M_s) \, \mathrm{d} \setminus \boldsymbol{\theta}_s. \end{aligned} \tag{4.52}$$

The surfel belief can be separated into two terms: one containing the surfel cluster potential of a surfel, $s \in S$, and the other containing the surfel cluster potentials of the rest of surfels, $R = S \setminus s$, namely

$$\mathcal{B}(\theta_s) \propto \underbrace{\int_{M_s} \phi_s(\boldsymbol{\theta}_s, M_s)\,\mathrm{d}M_s}_{\phi_p(\boldsymbol{\theta}_s)\prod_i \delta_{i\to p}(\boldsymbol{\theta}_s)} \underbrace{\int_{\setminus \boldsymbol{\theta}_s} \prod_{r\in R} \phi_r(\boldsymbol{\theta}_r, M_r)\,\mathrm{d}\setminus\boldsymbol{\theta}_s}_{\delta_{R\to s}(\boldsymbol{h}_s)}. \tag{4.53}$$

The first factor is identical to what was calculated previously when considering each surfel in isolation, with the messages from each likelihood cluster calculated according to Equation 4.8. We therefore assume that we can calculate this term. The second term is the message from the rest of the surfels in the map, which is due to the shared heights between surfels. This message essentially captures what the rest of the surfels in the map surmise about the heights of the current surfel. Because of the highly coupled nature of the problem, the marginalisation to calculate $\delta_{R\to s}(\boldsymbol{h}_s)$ is expensive to compute. To keep things tractable, we approximate this message using loopy belief propagation (LBP) [71] (see Section 2.2.2).

In order to use LBP, we must first construct a cluster graph for the full inference problem. We previously performed inference by only considering each surfel cluster potential when calculating each surfel belief in isolation. We therefore can construct the cluster graph of the full joint distribution, $p(\Theta, M, Z = \mathtt{z})$, by repeating the previous cluster graph (Figure 4.5c) for each surfel in the map, but also creating connections between surfels that share an edge. The resulting cluster graph is illustrated in Figure 4.7. The scope of the sepset between surfels is generally over the two heights from the shared edge between the surfels—for example $\mathcal{S}_{s,a} = \boldsymbol{h}_s \cap \boldsymbol{h}_a$. Although it is not shown explicitly, to ensure a valid cluster graph of the joint distribution it is also necessary to reduce the scope of some sepsets so that the graph obeys the *running intersection property*, which states that there can only be a single path between the same variable in different parts of the graph. To ensure this property is obeyed, we use the LTRIP algorithm of Streicher and du Preez [72] to construct a cluster graph for the fully-connected model.

From this cluster graph, we can instead calculate an approximation of the message from the rest of the surfels in the map, $\tilde{\delta}_{R\to s}(\boldsymbol{h}_s)$, by only considering the messages from the subset of surfels that are contiguous to the current surfel, $C \subseteq R$. This results in a much simpler and tractable calculation,

$$\tilde{\delta}_{R\to s}(\boldsymbol{h}_s) \propto \prod_{c\in C} \delta_{c\to s}(\mathcal{S}_{s,c}), \tag{4.54}$$

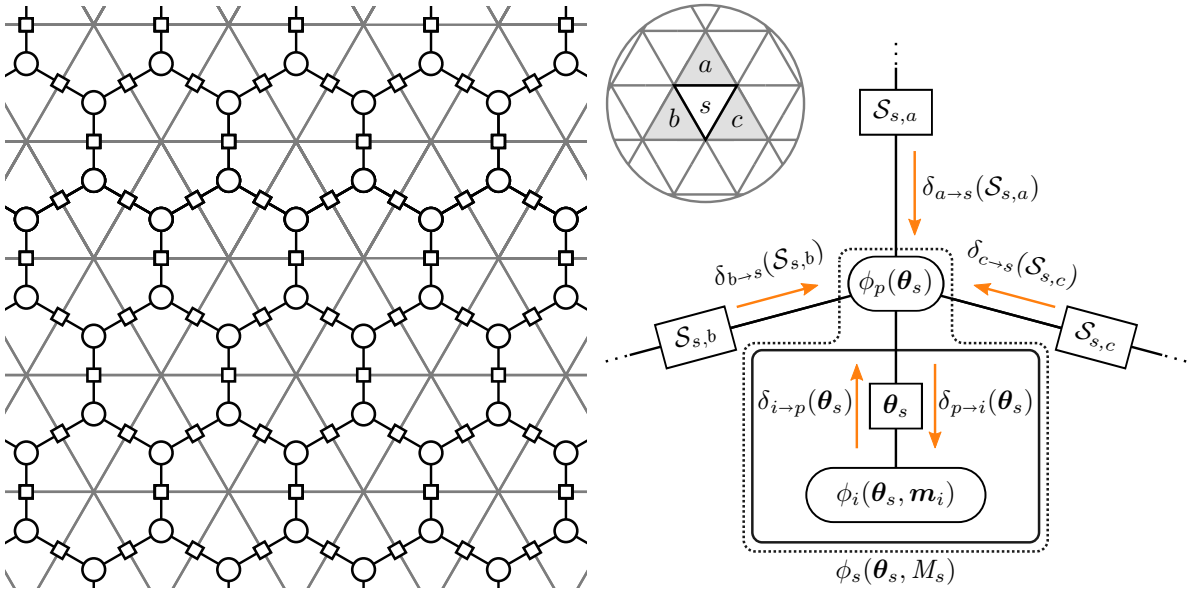
where $\mathcal{S}$ denotes the sepset between surfels. It should be noted that the scope of the resulting message might not always contain all the heights in the surfel (as would be the case for surfels on the border of the submap, or if the sepset scope has been reduced to satisfy the running intersection property). Using the integral-product algorithm, LBP defines the iterative update of a message from one surfel, s, to another surfel, a, according to

$$\delta_{s\to a}(\mathcal{S}_{a,s}) \propto \int_{\setminus \mathcal{S}_{a,s}} \phi_p(\boldsymbol{\theta}_s) \prod_i \delta_{i\to p}(\boldsymbol{\theta}_s) \prod_{c\in C\setminus a} \delta_{c\to s}(\mathcal{S}_{s,c})\,\mathrm{d}\setminus\mathcal{S}_{a,s}. \tag{4.55}$$

Using Equation 4.53, this can be simplified to

$$\delta_{s\to a}(\mathcal{S}_{a,s}) \propto \int_{\setminus \mathcal{S}_{a,s}} \frac{\mathcal{B}(\boldsymbol{\theta}_s)}{\delta_{a\to s}(\mathcal{S}_{a,s})}\,\mathrm{d}\setminus\mathcal{S}_{a,s}. \tag{4.56}$$

By iteratively passing these messages between surfels, LBP approximately calculates the marginal belief of each surfel. In order to perform this marginalisation and calculate each surfel belief, the



(a) (b)

Figure 4.7: (a) The cluster graph for the full inference problem shown for a zoomed-in section of a stochastic triangular mesh (STM) map. We indicate the sepsets between clusters with □, and each surfel cluster potential with ◯. The grey lines indicate the grid elements of the map. (b) We zoom in to the cluster graph, focusing on a specific surfel, s, which is contiguous with surfels a, b and c. The resulting graph is similar to the cluster graph in Figure 4.5c, but there are now connections between contiguous surfels. The expanded surfel cluster potential $\phi_s(\boldsymbol{\theta}_s, M_s)$ is shown using the dotted outline. The sepset between the surfels is denoted $\mathcal{S}$ (see text for more details on the scope of the sepsets).

likelihood cluster potentials in each surfel potential need to have a closed-form solution, but this is not the case due to the nonlinearities between the variables in the likelihood cluster potentials. In order to find a closed-form solution to each surfel belief (and consequently the map belief), we again turn to variational message passing (VMP) to approximate each likelihood cluster potential; that is, using the local information projection according to Equation 4.12, repeated here for clarity:

$$\tilde{\phi}_i^*(\boldsymbol{\theta}_s, \boldsymbol{m}_i) = \underset{\tilde{\phi}_i \in \mathcal{F}}{\arg\min}\, D_{\mathrm{KL}}\left(\tilde{\phi}_i \tilde{\delta}_{p\to i} \,\middle\|\, \phi_i \tilde{\delta}_{p\to i}\right).$$

However, the approximate incoming message, $\tilde{\delta}_{p\to i}(\boldsymbol{\theta}_s)$, is now calculated according to

$$\tilde{\delta}_{p\to i}(\boldsymbol{\theta}_s) = \phi_p(\boldsymbol{\theta}_s)\tilde{\delta}_{R\to s}(\boldsymbol{h}_s)\prod_{j\neq i}\tilde{\delta}_{j\to p}(\boldsymbol{\theta}_s). \tag{4.57}$$

Similarly to previously—Equation 4.9—this message is calculated by combining the prior cluster potential, $\phi_p(\boldsymbol{\theta}_s)$, with the messages from all other likelihood clusters, $\prod_{j\neq i}\tilde{\delta}_{j\to p}(\boldsymbol{\theta}_s)$, but now it additionally combines the message from neighbouring surfels, $\tilde{\delta}_{R\to s}(\boldsymbol{h}_s)$. Using Equation 4.53, the approximate incoming message can be rewritten in terms of the surfel belief,

$$\tilde{\delta}_{p\to i}(\boldsymbol{\theta}_s) \propto \frac{\tilde{\mathcal{B}}(\boldsymbol{\theta}_s)}{\tilde{\delta}_{i\to p}(\boldsymbol{\theta}_s)}, \tag{4.58}$$

which, in comparison to Equation 4.57, is cheaper to calculate when iterating over all the measurements. If we combine the prior potential, $\phi_p(\boldsymbol{\theta}_s)$, with the incoming message, $\tilde{\delta}_{R\to s}(\boldsymbol{h}_s)$, into a pseudo-prior, the same approach we described previously (Section 4.2.1) can be used to perform the local information projection. This specifically includes using the Gaussian family of

distributions to model the height distributions, which results in LBP performing approximate inference on only Gaussian distributions—a variant of LBP known as Gaussian LBP. Gaussian LBP is a standard problem formulation for which implementations are available, and we therefore use EMDW—an in-house PGM library[8]. LBP is a tractable approximate inference technique; however, due to its iterative nature, its convergence has not yet been proven. In Gaussian LBP, however, it is known that, if convergence is reached, the posterior means are correct [73, 74]. Although the surfel belief distributions in our problem are not guaranteed to be exactly Gaussian this is a fair assumption, as the number of measurements becomes large enough. In our implementation we have also not empirically experienced any issues with convergence.

The approach presented here performs inference on an STM map using a hybrid message-passing technique using loopy belief propagation (LBP) and variational message passing (VMP). Next we provide an overview of the proposed inference algorithm, and discuss some aspects of its implementation.

4.3 Algorithm Overview and Implementation

In Algorithm 1 we describe the process of performing inference given a single batch of measurements. Note that, this could be a batch of measurements accumulated over time, or from a single time step. Notably, lines 7 to 9 and 16 to 18 correspond to the loopy belief propagation (LBP) update steps, and lines 10 to 15 to the variational message passing (VMP) update steps. We also adjust the method of updating each surfel belief to be more efficient: if only the incoming message from neighbouring surfels has changed during the iterative update process, then it is unnecessary to calculate the updated surfel belief using Equation 4.53. We instead (equivalently) simplify the update to

$$\tilde{\mathcal{B}}(\boldsymbol{\theta}_s) \leftarrow \tilde{\mathcal{B}}(\boldsymbol{\theta}_s) \frac{\tilde{\delta}^*_{R\to s}(\boldsymbol{h}_s)}{\tilde{\delta}_{R\to s}(\boldsymbol{h}_s)}, \tag{4.59}$$

where we denote the updated and previous incoming messages $\tilde{\delta}^*_{R\to s}$ and $\tilde{\delta}_{R\to s}$ respectively. We next discuss some of the aspects of the algorithm.

Message initialisation We first need to initialise all of the messages before we can begin message passing. In general, the initialisation for iterative approximate inference algorithms can influence the performance, affecting the local optimum to which the algorithm converges, and the convergence speed. We initialise the messages between surfels to be vacuous. For the outgoing messages from each likelihood cluster, we use a simple empirical method of initialisation. Specifically, we initialise each outgoing message, $\tilde{\delta}_{i\to p}(\boldsymbol{h}_s)$, to have a mean at the γ component of the measurement $\mathbf{z}_i$, and a (practically) uninformative covariance. This ensures that $\tilde{\mathcal{B}}(\boldsymbol{h})$ is initialised to have a mean at the average γ component of the measurements. We initialise each outgoing message, $\tilde{\delta}_{i\to p}(\nu_s)$, such that the expected value calculated in Equation 4.28 is the variance of the γ component of the measurements.

Incremental updating During online operation, measurements are received incrementally, and we wish to calculate the surfel beliefs based on all currently available measurements. Ideally, the surfel beliefs should be recalculated as new measurement batches arrive, while also considering all previous measurements. However, as more measurement batches are received, this approach would quickly become intractable. A viable approximation is to only update the outgoing messages from a window of the W latest measurements [75]. Fixing the messages from clusters containing

[8] This is currently in the process of being made open source.

Algorithm 1 STM Map Inference

Inputs:
Prior cluster potentials for each surfel: $\phi_p(\boldsymbol{\theta}_s)\ \forall\ s \in S$
Measurements in each surfel: $\mathtt{Z}_s\ \forall\ s \in S$
Output:
Surfel beliefs: $\tilde{\mathcal{B}}(\boldsymbol{\theta}_s)\ \forall\, s \in S$

Initialise all messages

▷ *Initialise all surfel beliefs* ..... *Equation* 4.53
for all $s \in S$ **do**
  $\tilde{\mathcal{B}}(\boldsymbol{\theta}_s) \leftarrow \phi_p(\boldsymbol{\theta}_s)\tilde{\delta}_{R\to s}(\boldsymbol{h}_s)\prod_i \tilde{\delta}_{i\to p}(\boldsymbol{\theta}_s)$
end for

repeat
  ▷ *Cycle through all the surfels in the map*
  for all $s \in S$ **do**
    ▷ *Calculate the updated incoming message from neighbouring surfels* .. *Equation* 4.54
    $\tilde{\delta}^*_{R\to s}(\boldsymbol{h}_s) \leftarrow \prod_{c\in C} \delta_{c\to s}(\mathcal{S}_{s,c})$

    ▷ *Update the surfel belief* ..... *Equation* 4.59
    $\tilde{\mathcal{B}}(\boldsymbol{\theta}_s) \leftarrow \tilde{\mathcal{B}}(\boldsymbol{\theta}_s)\frac{\tilde{\delta}^*_{R\to s}(\boldsymbol{h}_s)}{\tilde{\delta}_{R\to s}(\boldsymbol{h}_s)}$
    $\tilde{\delta}_{R\to s}(\boldsymbol{h}_s) \leftarrow \tilde{\delta}^*_{R\to s}(\boldsymbol{h}_s)$

    ▷ *Update the likelihood cluster potential for each measurement*
    for all $\mathtt{z}_i \in \mathtt{Z}_s$ **do**
      ▷ *Calculate incoming message to the likelihood cluster* ..... *Equation* 4.58
      $\tilde{\delta}_{p\to i}(\boldsymbol{\theta}_s) \leftarrow \frac{\tilde{\mathcal{B}}(\boldsymbol{\theta}_s)}{\tilde{\delta}_{i\to p}(\boldsymbol{\theta}_s)}$

      ▷ *Update the likelihood cluster potential* ..... *Section 4.2.1*
      $\tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i)\tilde{\phi}_i(\nu) \leftarrow \arg\min_{\tilde{\phi}_i} D_{\mathrm{KL}}\left(\tilde{\phi}_i\tilde{\delta}_{p\to i}\,\middle\|\,\phi_i\tilde{\delta}_{p\to i}\right)$

      ▷ *Calculate the outgoing message from the likelihood cluster* ..... *Equation* 4.8
      $\tilde{\delta}_{i\to p}(\boldsymbol{\theta}_s) \leftarrow \tilde{\phi}_i(\nu)\int \tilde{\phi}_i(\boldsymbol{h}, \boldsymbol{m}_i)\,\mathrm{d}\boldsymbol{m}_i$

      ▷ *Update the surfel belief* ..... *Equation* 4.11
      $\tilde{\mathcal{B}}(\boldsymbol{\theta}_s) \leftarrow \tilde{\delta}_{i\to p}(\boldsymbol{\theta}_s)\tilde{\delta}_{p\to i}(\boldsymbol{\theta}_s)$
    end for

    ▷ *Calculate the outgoing messages to neighbouring surfels* ..... *Equation* 4.56
    for all $c \in C$ **do**
      $\delta_{s\to c}(\mathcal{S}_{c,s}) = \int \frac{\mathcal{B}(\boldsymbol{\theta}_s)}{\delta_{c\to s}(\mathcal{S}_{c,s})}\,\mathrm{d} \setminus \mathcal{S}_{c,s}$
    end for
  end for
until all messages converge

measurements outside this window ensures that updating the surfel belief is linear in the window size W. In practice, we combine these constant messages with the prior, discarding the old measurements so that the storage will not grow unbounded. This new prior now summarises

all the previous measurements. Following this approach, we can then use the same procedure outlined in Algorithm 1. It should also be noted that previously calculated messages between surfels should not be discarded, but now used as the initialisation for the next batch update.

Convergence To monitor the state of convergence, we use the fact that all messages will be constant at convergence—that is, convergence has been reached if all messages do not change between iterations. In practice, a message is deemed to be constant if the difference between the successive iterations of the message is negligible. To evaluate this difference, we use the exclusive Kullback-Leibler (KL) divergence[9] between a message at the current and previous iterations—$D_{\mathrm{KL}}\left(\text{current iteration} \,\|\, \text{previous iteration}\right)$—and the message has converged if the resulting divergence is below a threshold. If all messages related to a surfel have converged, then it is no longer necessary to iterate over it. It should also be noted that, although convergence is not theoretically guaranteed (due to the use of loopy belief propagation (LBP)), we have not empirically experienced any issues with convergence in our implementation.

Computational complexity We can see that the asymptotic computational complexity of our algorithm is $\mathcal{O}(KN)$, where K is the number of iterations in the outermost loop before convergence, and N is the total number of measurements being incorporated into the map. In practice, K is constant with respect to N, and therefore the algorithm can be considered approximately linear in N. We support this claim with experimental results (Section 5.2).

4.4 Discussion

We also investigated using expectation propagation (EP) to perform inference as an alternative to VMP. However, this required performing expensive high-dimensional numerical integration [76, 77], and suffered severely from numerical instability, making it unsuitable for our application.

We have now described performing inference on an STM map. Next we look at a series of experiments evaluating the use an STM map to represent environments with both simulated and practical data.

[9] However, other divergences could be used: the symmetric KL divergence, inclusive KL divergence, the Mahalanobis distance, or the Bhattacharyya distance, to name a few.

5 Experimental Results

Up to this point we have discussed how a stochastic triangular mesh (STM) map models the environment, as well as the process of performing inference in the map. We now investigate how the STM mapping technique performs under different experimental conditions. Note that, although we only test STM mapping using data from ground vehicles, it is applicable to all types of autonomous robots: terrestrial, underwater, and aerial.

For the first set of experiments, we use simulated measurement data (Sections 5.1 to 5.3). We investigate how the prior distribution parameter choices affect the resulting map (Section 5.1), analyse the cost of performing inference (Section 5.2), and compare the accuracy of an STM mapping to elevation mapping and Gaussian process (GP) mapping (Section 5.3).

We then consider practical datasets collected using either stereo cameras or a light detection and ranging (LiDAR) sensor (Sections 5.4 and 5.5). We demonstrate the modelling performance of the STM mapping technique in relative and global inertial reference frames (IRFs), and perform a qualitative analysis of the resulting STM maps (Sections 5.4 and 5.5 respectively). Using practical data, we also perform a quantitative comparison between elevation mapping and STM mapping (Section 5.4).

5.1 Parameters of the Prior Distribution

In Section 4.2.1, Equation 4.18, we gave the form of the prior distribution over the surfel parameters, but did not specify the parameters of the prior distribution. For surfels in the map that contain many measurements, the choice of prior parameters has a negligible effect on the resulting map belief. However, if there are clusters of neighbouring surfels with no measurements, then these surfels' beliefs are largely determined by the prior distributions.

We want to capture knowledge of the structure of the environment in the prior distribution. The structure we expect is that neighbouring regions tend to have similar heights, which means that we expect neighbouring heights to be positively correlated. If a completely uninformative height prior distribution is used, then only the heights of shared vertices will be affected when performing inference in unobserved regions. In order to achieve this, we use a height prior distribution with covariance,

$$\mathbf{\Sigma}_p = \sigma^2 \begin{bmatrix} 1 & \rho & \rho \\ \rho & 1 & \rho \\ \rho & \rho & 1 \end{bmatrix}, \tag{5.1}$$

where ρ is the Pearson correlation coefficient and σ^2 the variance. Without any other information, we expect the heights in a submap to be distributed around the submap plane. We therefore choose the mean of the height prior distribution to represent this initial belief; that is, we set the mean to the zero vector. We choose σ^2 to be uninformative and focus on ρ, which should be chosen

in the range $\rho \in [0, 1)$ to ensure a positively correlated prior distribution. It should be noted that, as the planar deviations in neighbouring surfels are modelled as being statistically independent, we model the prior of the planar deviation in a surfel as an uninformative inverse-gamma distribution.

In Figure 5.1, we show the result of varying the correlation coefficient on the map belief for a 2-D simulation. From this we can see that, by increasing the correlation coefficient, the mean heights of unobserved surfels are affected at an increasing distance from the observed region, which is consistent with smoothly varying environments. We empirically chose $\rho = 0.5$, although this choice may vary depending on the nature of the environment. This parameter could conceivably be learnt from data.

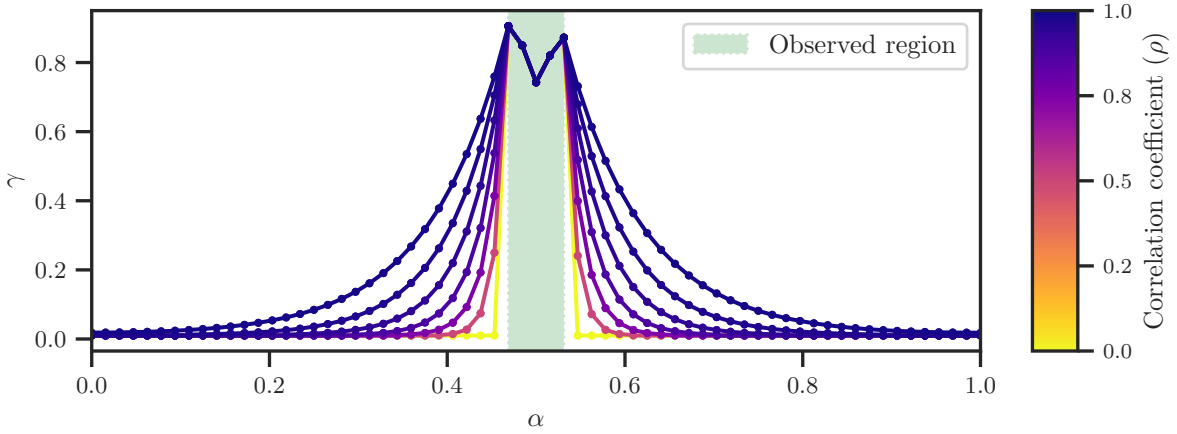


Figure 5.1: The effect of using a correlated prior distribution when a section of the environment is observed. This is shown for a series of correlation coefficients, spaced using a geometric series: $\{0, 2^{-1}, 2^{-1}+2^{-2}, \ldots\}$. Only the mean of the mean mesh belief is shown, and it is coloured according to the correlation coefficient (the darker, the more correlated).

5.2 Cost of Inference

From a tractability perspective, for the STM mapping technique to be viable for online dense mapping, the cost of incorporating one measurement into the map should be constant, irrespective of the size of the map or the number of measurements in the map. This computational requirement is equivalent to the computational complexity of updating the map being linear in the number of measurements, which we previously stated to be true for an STM map (Section 4.3). To motivate this statement, we analyse the cost of inference by simulating two scenarios we expect a robot to encounter frequently in practice: incrementally observing new regions of the environment, and re-observing previously observed regions. It should be noted that, to quantify the cost of inference, we count the number of outgoing messages calculated when updating the map (Algorithm 1, lines 13 and 16). This is an appropriate metric, as it measures the number of iterations for the innermost loops of our inference algorithm.

For both scenarios, we represented the surface of the environment using a 2.5-D surface generated using Perlin noise[1]—a different surface was generated for each scenario. A single submap was used containing 1024 surfels. Simulated measurements were generated within the relative IRF of the submap by randomly sampling the surface in the observed region (we discuss the sampling

[1] A type of gradient noise originally developed by Perlin [78], which is used to procedurally generate virtual environments and textures in computer graphics.

strategies further shortly), and then adding zero-mean non-identically distributed Gaussian noise to each sample. The noise covariance was determined by randomly rotating a diagonal covariance matrix. We also ensured that the measurement density in the observed region was approximately uniform and constant across all time steps, at 10 measurements per surfel. Finally, after updating the map at each time step, all the new measurements are windowed.

In the first scenario, we simulated a newly observed region of the environment that shifts incrementally forward–this is typically the case for a robot driving with a 2-D LiDAR sensor mounted in a push-broom configuration. At each time step, simulated measurements are generated from a sampling region (Figure 5.2a), and the map is updated with the new measurements. The cost of inference for this experiment is shown in Figure 5.2b. We see that the total number of messages passed at each time step has a linearly decreasing trend, which is expected due to the linearly decreasing sampling region. When we normalise the message counts in each update by the number of new measurements in the update, then this normalised message count is relatively constant. Note that with each incremental update, only local regions of the map were affected significantly (Figure 5.2c).

In the second scenario, we simulated a region of the environment that is observed repeatedly. This emulates a robot revisiting a previously explored region, or a static robot repeatedly observing the same region. At each time step, simulated measurements are generated by sampling from the entire mapping region. The number of messages passed follows an exponentially decreasing trend (Figure 5.3); that is, subsequent observations of a previously observed region become computationally cheaper. This is because fewer iterations are required to reach convergence.

From these two scenarios, we see that updating an STM map does indeed have a linear computational cost in the number of measurements, which makes STM mapping a viable online dense mapping technique for the scenarios we expect a robot to encounter in a practical application. We have, however, only analysed the relative computational cost of performing inference on an STM map. When considering the absolute computational cost, one of most important aspects is the message and factor dimensionality. In our algorithm, a message passed between neighbouring surfels has at most two dimensions, a message passed within a surfel has three dimensions, and the largest factor has six dimensions. These dimensions are all fixed and relatively low, therefore dimensionality is not a problem for our algorithm. Additionally, although our unoptimised Python implementation takes ± 3 ms per measurement to update an STM map[2], we expect significantly faster results with an efficient implementation.

5.3 Comparisons of Model Accuracy

An STM map is an explicit surface representation that also captures model uncertainty. Therefore, to evaluate the modelling accuracy of an STM map, we compare it against a similar class of mapping techniques, namely ones that also use an explicit surface representation and capture model uncertainty. From our summary of the key attributes of the existing mapping techniques (Table 1.1), the suitable candidate mapping techniques are normal distributions transform (NDT) mapping, Gaussian process (GP) mapping, and elevation mapping. NDT mapping, however, does not model any measurement uncertainty. We therefore only compare STM mapping against elevation mapping and GP mapping.

[2]This is based on a later experiment (Section 5.4) using 19 million measurements.

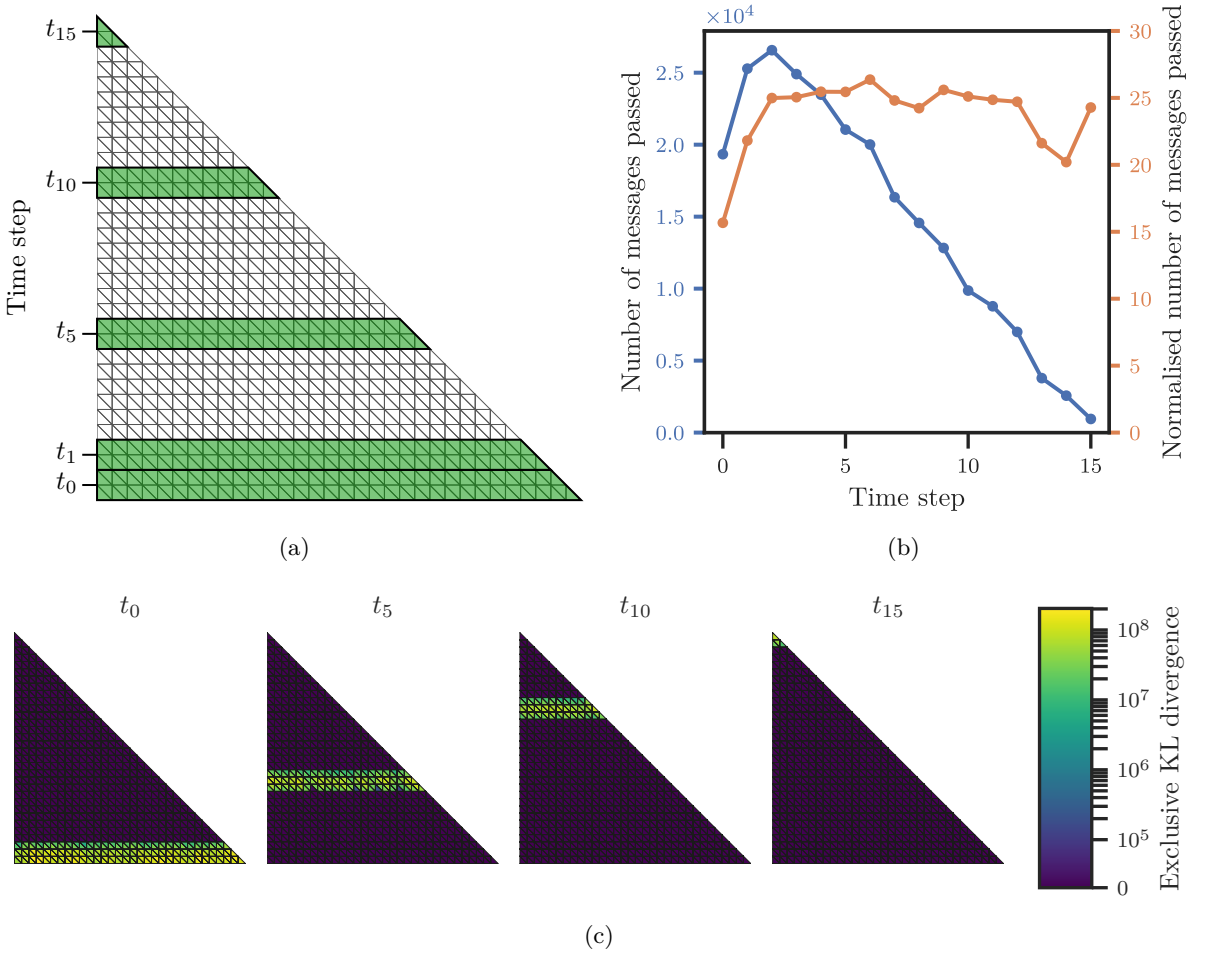


Figure 5.2: An experiment showing the cost of inference for the scenario in which new regions of the environment are incrementally observed. (a) The experimental setup showing the sampling regions at selected time steps (green, shaded). Note that the size of the sampling region at each time step decreases linearly. (b) The cost of inference is quantified by the number of messages passed (blue, left axis). This message count is also normalised by the number of new measurements in each update (orange, right axis). (c) To visualise the incremental changes in the surfel beliefs, we calculate the exclusive Kullback-Leibler (KL) divergence between the surfel beliefs before and after the measurements at a specific time step were incorporated. Note, the colour mapping is scaled using a symmetrical logarithm, which is linear on $[0, 10^5]$ and logarithmic everywhere else.

5.3.1 Comparison of Elevation Mapping

We first compare STM mapping with standard elevation mapping—that is, each height is updated using a one-dimensional Kalman filter [17, 18]. We perform this comparison using simulated measurement data by quantifying each model's accuracy against the ground truth using the mean squared error (MSE) of each model, and the log-likelihood ratio between both models. Using the same set of measurements, models were built for increasingly finer grid divisions. We first perform this comparison on a 2-D environment[3], as it is more intuitive to visualise, and we then extend this to a 3-D environment, which is generated using Perlin noise. We also ensured that there were ± 10 measurements per surfel at the finest grid division (for both the 2-D and 3-D experiments).

[3]The 2-D simulation environment was created using the height profile of a mountain extracted using manual photogrammetry.

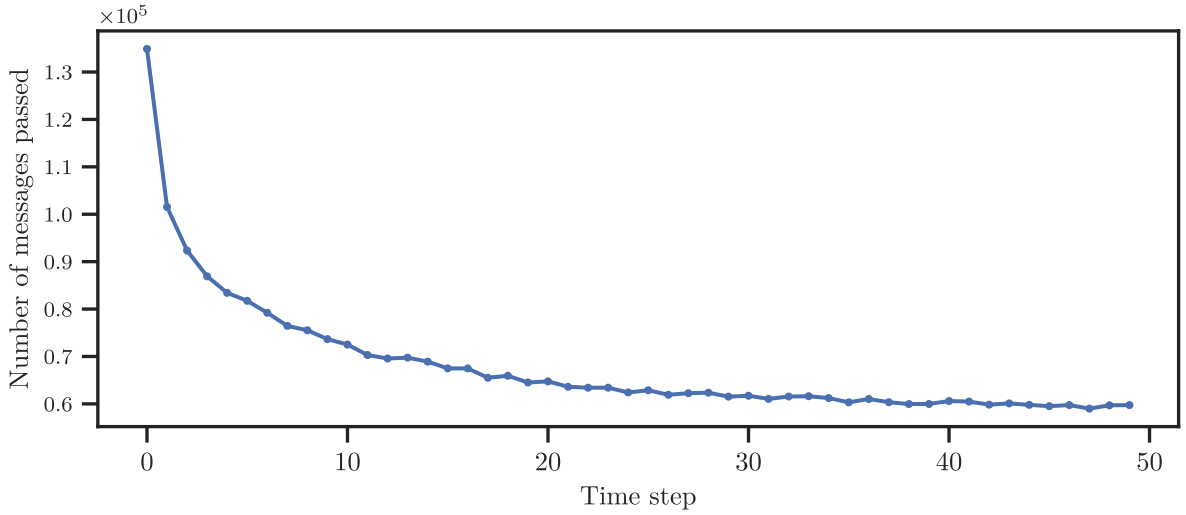


Figure 5.3: An experiment showing the cost of inference for the scenario in which the same region of the environment is observed repeatedly.

The results of the 2-D experiment are shown in Figure 5.4. The MSE of the STM map is better than that of the elevation map for all grid divisions greater than zero. When there is no subdivision (grid division depth = 0), the mean plane of the only surfel in the STM map is almost zero and at the same height as the elevation map (Figure 5.4c, grid division depth = 0), because the average gradient of the surface of the environment is close to zero. As the grid divisions increase past a certain point, the accuracy of both models begins to degrade (for the MSE, this occurs for grid divisions > 6 for the STM map and > 7 for the elevation map). This effect is attributed to the measurement noise (relative to the size of the grid element) becoming too large, and the number of measurements in each grid element decreasing with each grid division. The effect is apparent when looking at the grid division depth of 8 in Figure 5.4c. From the MSE alone, this result might not appear significant; looking at the log-likelihood ratio shows that, for this environment, the STM map is a more likely model by orders of magnitude. This is as a consequence of the elevation map estimating only the mean height of the surface, whereas the STM map represents the surface as a stochastic process, which is a better representation of the surface of the environment.

For the 3-D experiment, we ran a batch simulation for 10 synthetically generated environments (Figure 5.5). Compared to the 2-D experiment, we see a similar trend in the MSE, namely that the STM map is clearly and consistently better than the elevation map for grid division greater than 2. Interestingly, as the grid divisions decrease, there is also a decrease in the batch MSE standard deviation for both models (> 2 grid divisions). This is because the lower grid divisions are too coarse to effectively represent the environment, and therefore the model accuracy is largely dependent on the particular generated environment. Similarly to the 2-D experiment, the log-likelihood ratio between both models shows that an STM map is a significantly better representation than an elevation map.

From these results, we can conclude that, in comparison to a standard elevation map, an STM map is a more accurate and better representation of the environment for appropriately chosen grid divisions.

5.3.2 Comparison of Gaussian Process Mapping

Elevation mapping has been demonstrated practically to be a real time mapping technique [18] but, as we have shown, an STM map is a more accurate representation of the environment. Although GP mapping is intractable as an online dense mapping technique due to the cubic computational complexity, it has a more expressive modelling capability (as it also models the surface as a stochastic process). We therefore use GP mapping as a baseline for model accuracy. In order to perform the comparison, we implemented GP mapping using GP regression in GPy [79]—a standard GP toolbox—by using a Matérn kernel and optimising for the kernel hyperparameters.

We use a similar experimental setup to the elevation mapping comparison; that is, we perform 2-D and 3-D comparisons using the same environment setups. However, because a GP map is non-parametric, there is no notion of a grid division. We therefore only compare GP mapping to

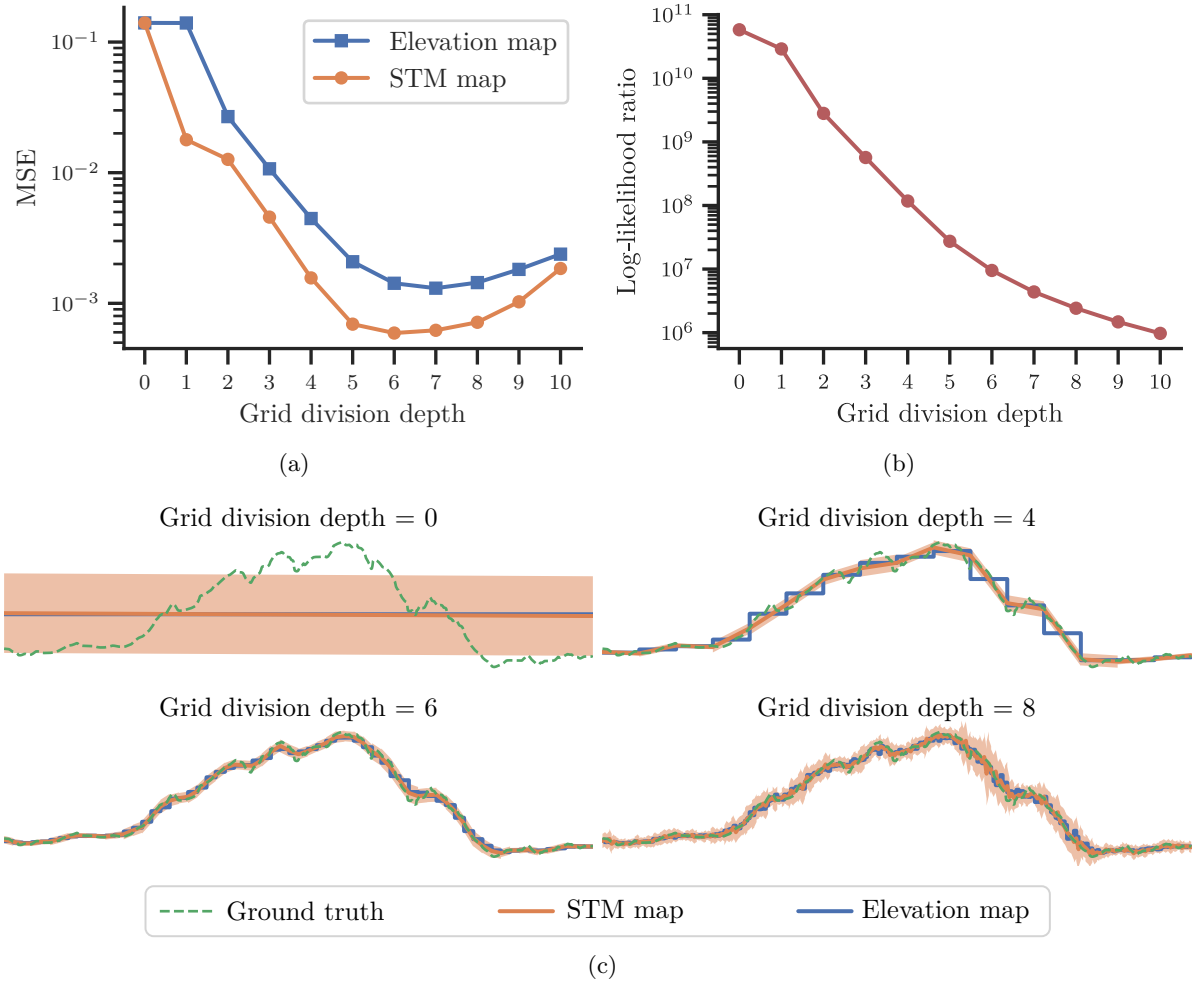


Figure 5.4: An experiment showing a 2-D simulation comparing a stochastic triangular mesh (STM) map with the standard elevation map [17, 18]. For increasing grid divisions, we calculate (a) the mean squared error (MSE) of each mapping technique compared to the ground-truth surface (lower is better), and (b) the log-likelihood ratio between the STM map and the elevation map is evaluated along the ground-truth surface (lower is better); that is, log (STM map likelihood) − log(elevation map likelihood). Note that the vertical axis for both comparisons uses a log scale. (c) We also show the ground truth and resulting models for selected grid depths. The shaded regions indicate the one standard deviation confidence intervals in both models (the standard deviations of the elevation map are too small to be seen).

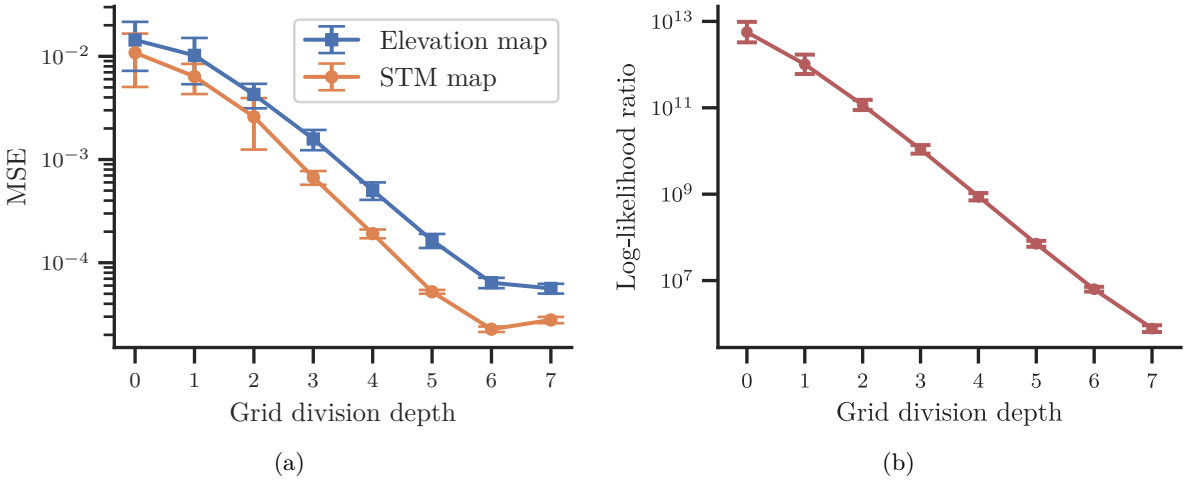


Figure 5.5: An experiment showing a 3-D batch simulation ($n = 10$) comparing a stochastic triangular mesh (STM) map with the standard elevation map [17, 18] for various generated environments. The mean and first standard deviation are shown for (a) the mean squared error (MSE) of each mapping technique compared to the ground truth surface, and (b) the log-likelihood ratio between the STM map and the elevation map is evaluated along the ground-truth surface; that is, log (STM map likelihood) − log(elevation map likelihood).

STM mapping at a fixed grid division.

For the 2-D GP mapping comparison, we use a grid division = 6 (from Figure 5.4 this was determined to be optimal for the STM map), and the same number of measurements were used as in the 2-D elevation map comparison. For the 3-D comparison, we use only a grid division = 3, because the GP map could not handle enough measurements to warrant further subdivision.

In Figure 5.6 we show the qualitative results of the 2-D comparison between GP mapping and STM mapping for two cases: accurate and inaccurate measurements. For accurate measurements, the GP map is a more precise model of the environment compared to the STM map, which is quite clear from the zoomed-in region. For the inaccurate measurements, the means of both maps are close. Additionally, because the GP map does not model the known non-independently and identically distributed measurement noise distributions (that the STM map does), the GP map was more uncertain than the STM map. Although no definite conclusions can be drawn from this 2-D experiment, it visually illustrates the behaviour of both mapping techniques.

For a quantitative comparison between STM and GP maps, we ran two batch 3-D simulations for the two cases of accurate and inaccurate measurements, over 10 environments synthetically generated using Perlin noise. We then compared the MSE and log likelihoods of both mapping techniques for both cases (Figure 5.7). For the case of accurate measurements, the GP map has an MSE that is 100 times more accurate, although the log likelihood of the GP map is only better by a factor of two. For the case of inaccurate measurements, the MSE of the GP map is better by a factor of 2, whereas the log likelihoods are on average almost equivalent. For the case of accurate measurements, we see that the GP map is significantly more accurate than the STM map. However, for the case of inaccurate measurements, this disparity is no longer as apparent, and the GP map is not significantly more accurate—especially when comparing the log likelihood.

Although a GP map can more accurately model the environment in comparison to an STM map, GP maps are very expensive to compute and store, and therefore are not suitable for online dense mapping. In the case where the measurements are inaccurate, although the mean estimate

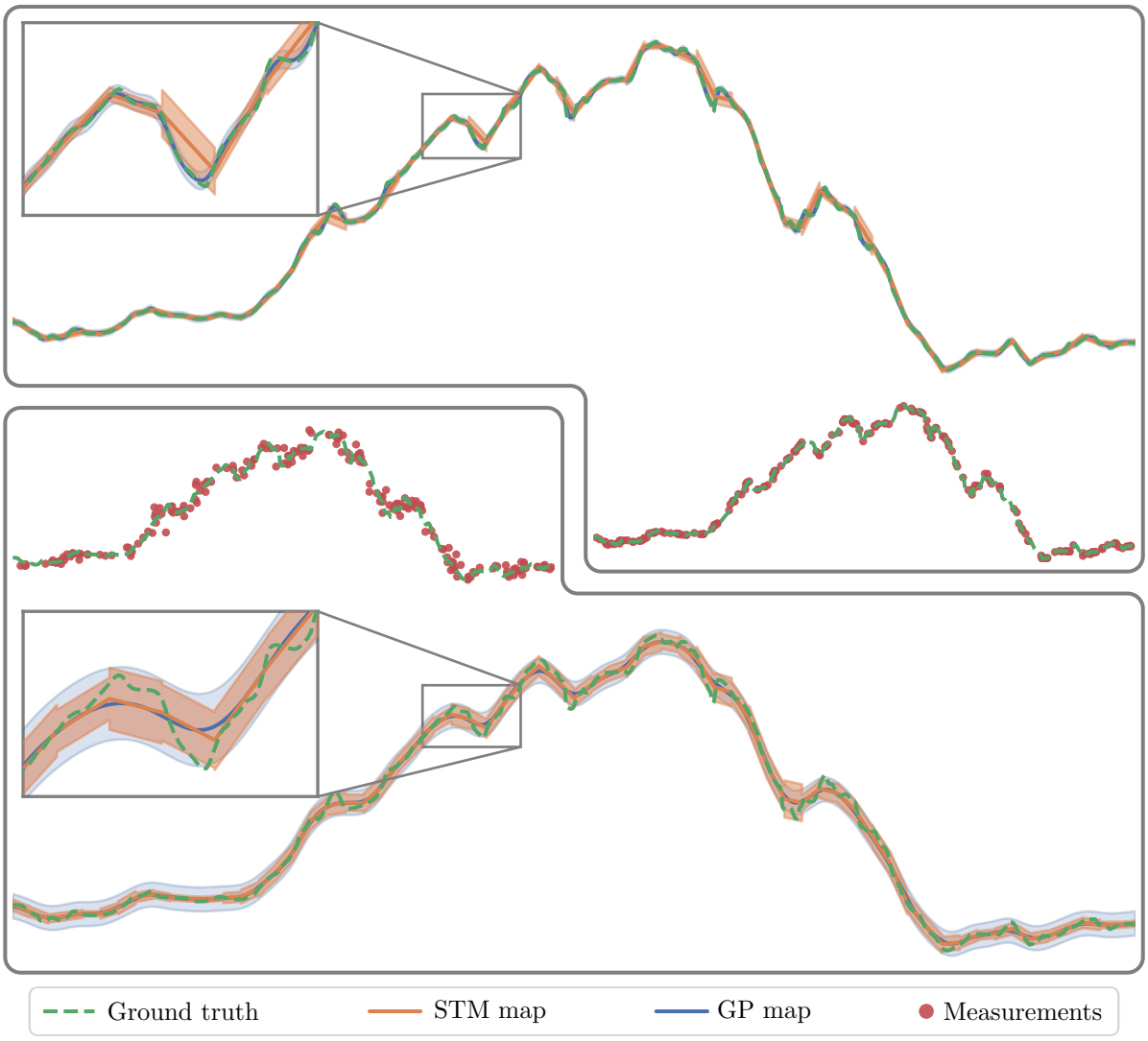


Figure 5.6: An experiment comparing STM mapping (orange) with Gaussian process (GP) mapping (blue) using (top) accurate and (bottom) inaccurate measurements in a 2-D simulated environment. The shaded regions indicate one standard deviation confidence intervals in both models.

of the GP map is better by a factor of two, the map likelihoods are almost exactly the same. If only the map mean is used, then the GP map is better, but if the full map distribution is used, then there is no advantage to using a GP map. In the case where accurate measurements are used, although the mean estimate of the GP map is orders of magnitude more accurate, this disparity is not as significant when comparing the full map distribution.

5.4 STM Mapping in a Global IRF

In order to demonstrate the ability to use STM mapping in a global IRF, we use the Canadian planetary emulation terrain 3-D mapping dataset by Tong et al. [80]. Specifically, we use the `box_met` dataset[4] taken in the Mars emulation terrain—a (60×120) m outdoor area. The `box_met` dataset consists of 19.04×10^6 LiDAR measurements, taken from 112 poses, and aligned using a differential global positioning system (DGPS).

[4] `http://asrl.utias.utoronto.ca/datasets/3dmap/box_met.html`

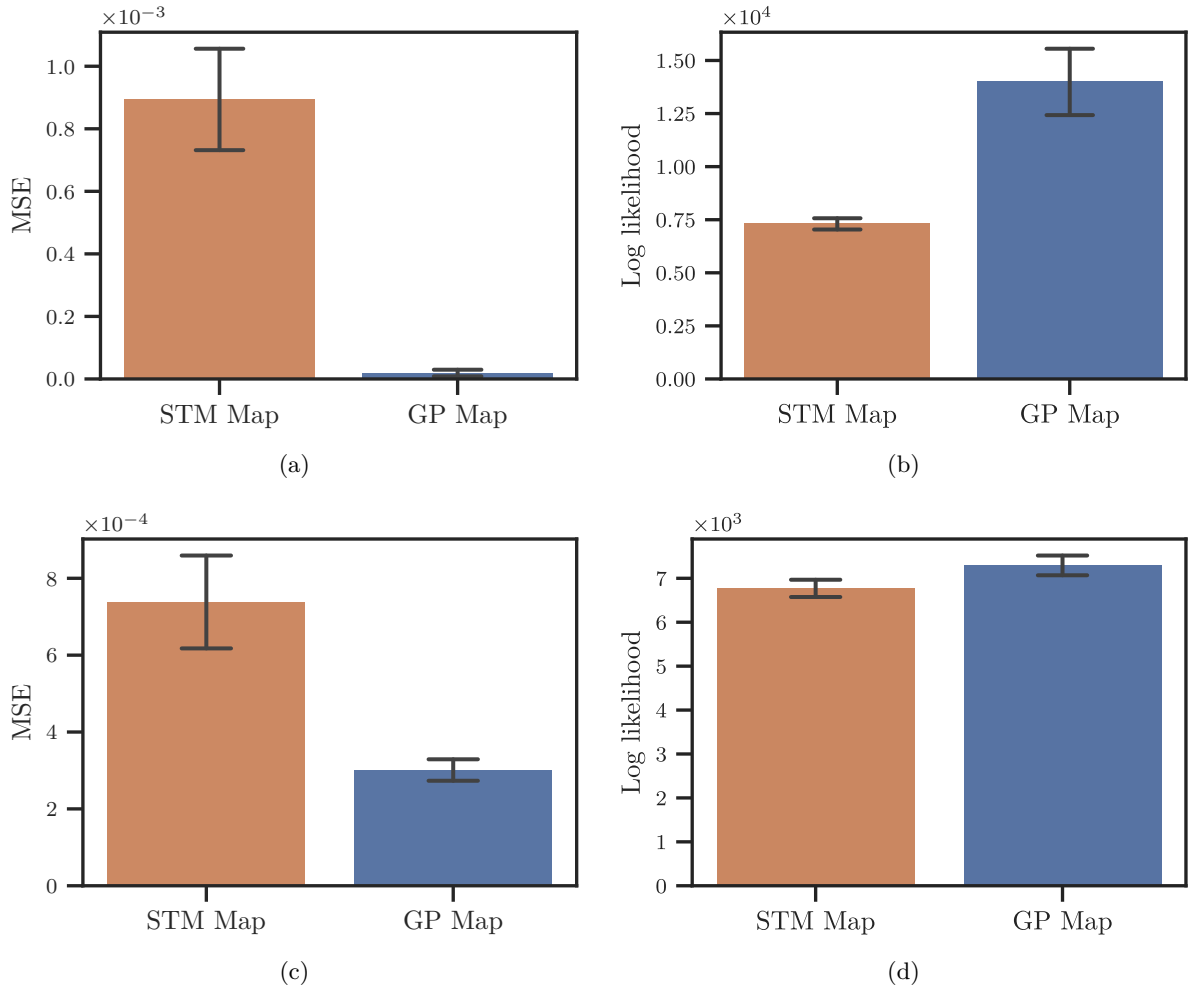
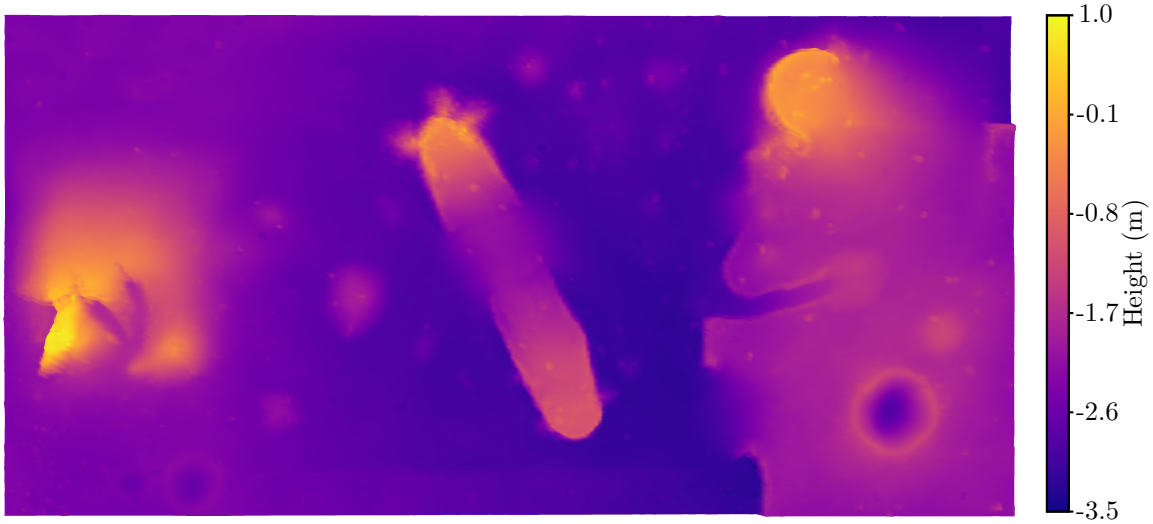


Figure 5.7: A 3-D batch simulation experiment comparing STM mapping (orange) with Gaussian process (GP) mapping (blue) using (a, b) accurate and (c, d) inaccurate measurements ($n = 10$ in each case). The mean and first standard deviation are shown for the mean squared error (MSE) (left column, lower is better), and the log likelihoods (right column, higher is better) of each mapping technique compared to the ground-truth surface.

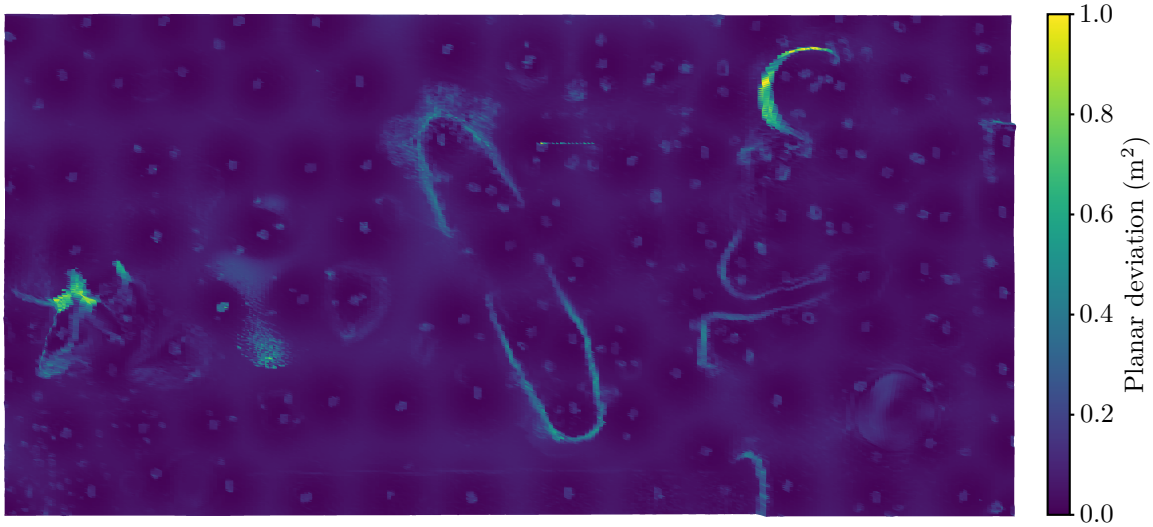
5.4.1 Large-Scale Mapping

In order to construct an STM map, a global IRF was created for the rectangular mapping region. The region was subdivided into two submaps, each containing 65.54×10^3 surfels. From the resulting STM map (Figure 5.8), we see that the mean mesh belief (Figure 5.8b) is visually consistent with satellite imagery. From the zoomed-in region (Figure 5.8d), we see that all the major features visible in the satellite imagery of the environment, even small rocks, are also visible in the map belief. We again see that the planar deviation beliefs (Figure 5.8c) are visually consistent with the terrain roughness. The steeper regions in the environment, which have a higher planar deviation, also form an outline of the obstacles in the environment. This experiment demonstrates that an STM map can be created in a global IRF framework. Although this lacks the benefits of a relative IRF, the added complexity of a relative IRF may not be necessary in situations where the robot localisation is guaranteed to perform accurately; however, this is generally not the case. Additionally, this experiment shows that STM maps can handle datasets with a large number of measurements.

(a)

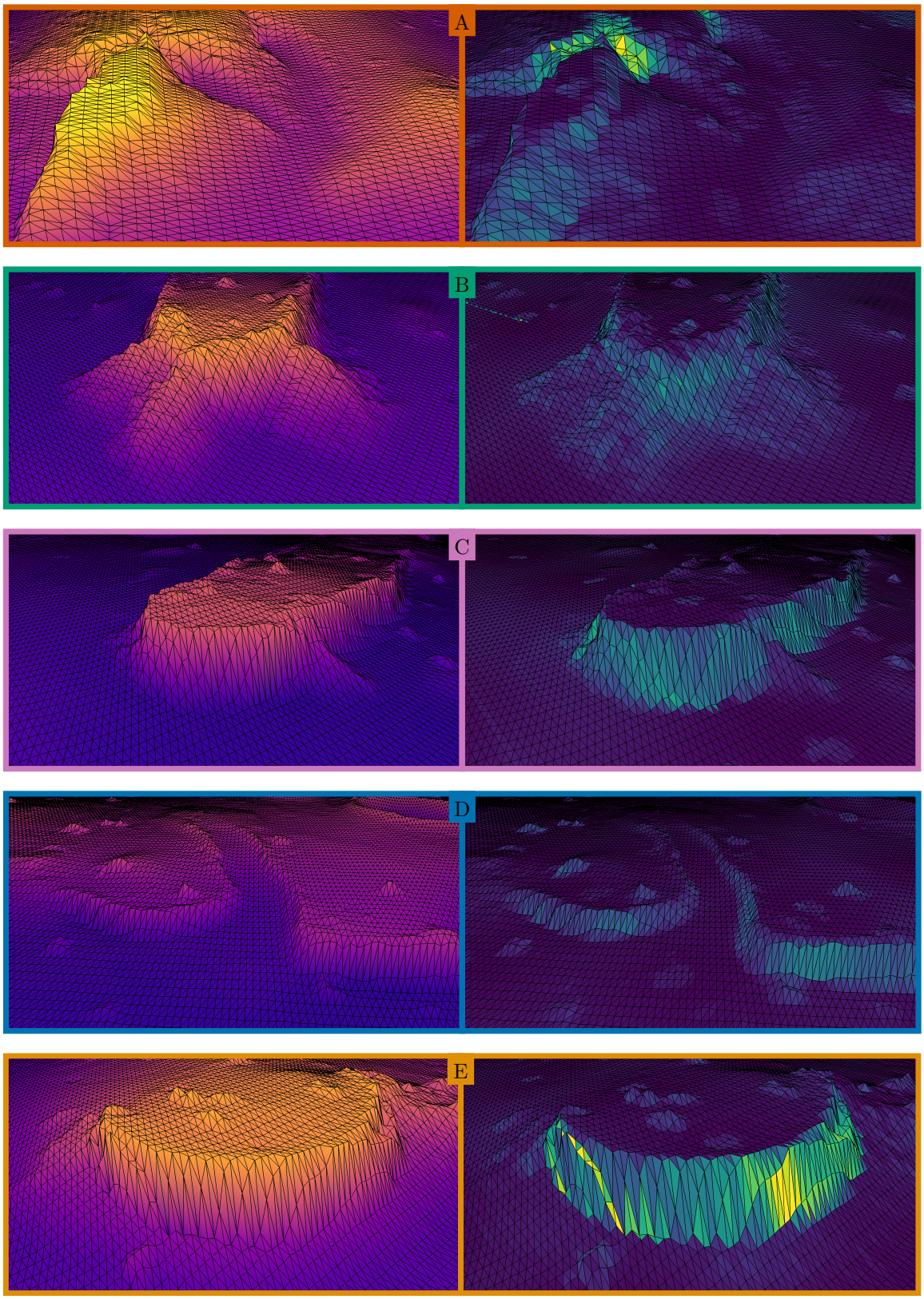


(b)



(c)

Figure 5.8: An experiment showing an STM map created in a global IRF using the Canadian planetary emulation terrain 3-D mapping dataset [80]. (a) A satellite view of the mapping region (Google Earth). Note that five zoom regions are demarcated using coloured rectangles and labelled A to E. The resulting map is shown for (b) the mean of the mean mesh belief, and (c) the planar deviation beliefs—calculated according to Equation 4.28.



(d)

Figure 5.8: (*Continued*) (d) The resulting map for the zoomed regions is shown for the mean of the mean mesh belief (left) and the planar deviation belief (right). These beliefs share the same colour maps as in (b, c).

5.4.2 Practical Comparison of Elevation Mapping

We previously compared elevation mapping with STM mapping (Section 5.3.1); however, this comparison was performed only on a synthetic simulation environment. In order to further substantiate the claim that an STM map is a superior model to an elevation map, we perform a comparison using practical data: a section of the environment from the Canadian planetary emulation terrain 3-D mapping dataset [80] (Figure 5.8d-E).

As we do not have a ground truth to validate each model, we instead perform Monte Carlo cross-validation[81]—a standard model-validation technique for situations where ground-truth data is not available. We randomly partition the measurement dataset using an 80 % training and 20 % testing split, repeating this process for 10 folds. For each fold, we build an STM and elevation map using the training data, and then calculate the log-likelihood ratio between both maps using the test data ($\log$ (STM map likelihood) $-$ $\log$(elevation map likelihood)). This approach is similar to that of Stoyanov et al. [27]; however, Stoyanov et al. perform evaluation using receiver operating characteristic (ROC) curves, whereas we use the log-likelihood ratio.

In Figure 5.9, we show the resulting maps from a fold. The resulting log-likelihood ratio mean was 1.592×10^9 with a standard deviation of 23.41×10^6 (1.471 % of the mean). Similarly to the synthetic simulation experiment, we see that STM mapping is a significantly more accurate model of the environment.

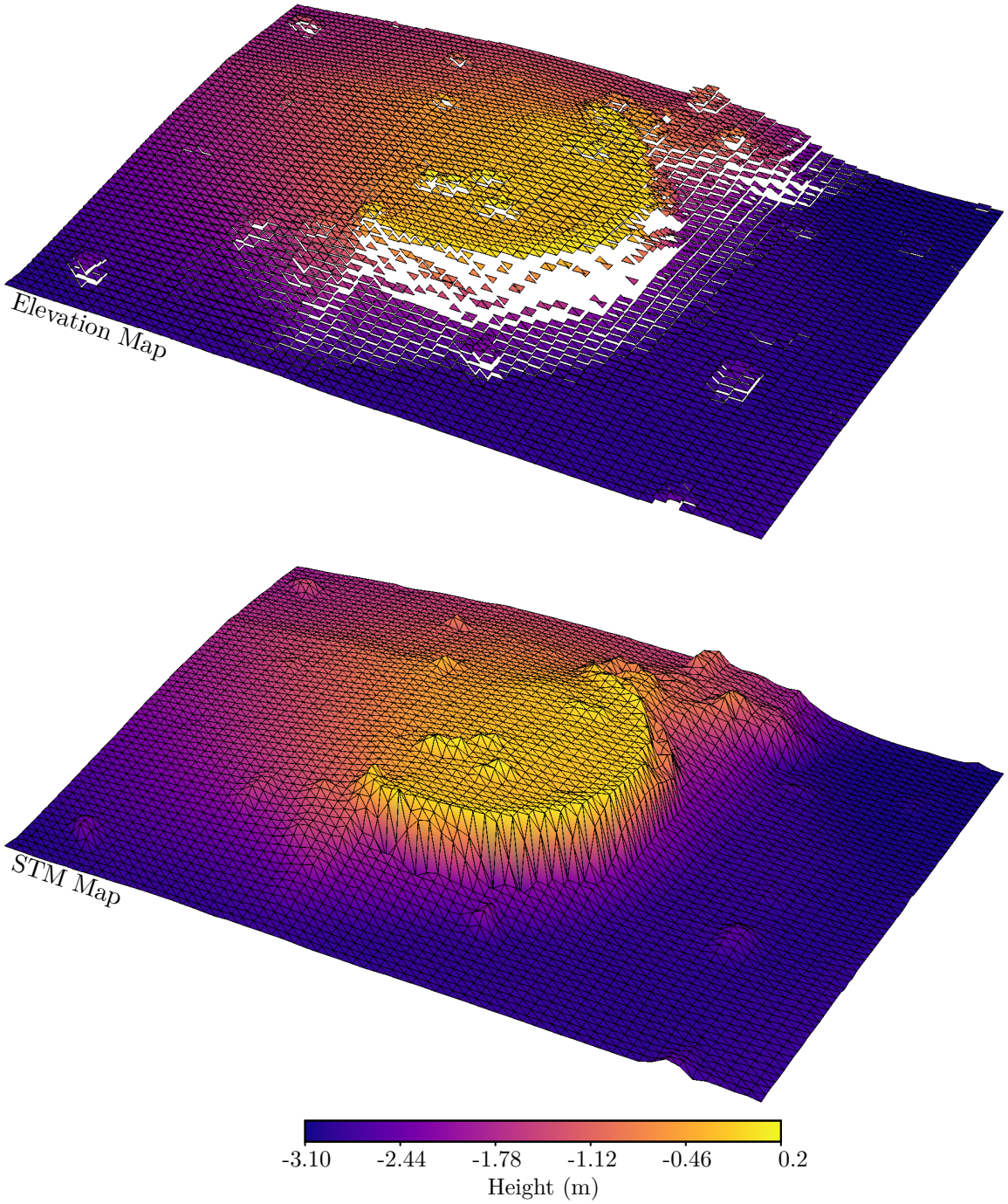


Figure 5.9: An elevation map (top) and STM map (bottom) built using a section of the Canadian planetary emulation terrain 3-D mapping dataset [80].

5.5 STM Mapping in a Relative IRF

Although robustly and persistently identifiable landmarks are not currently possible for general environments, there are still viable practical applications using landmark-based relative IRFs. Using two experiments, we demonstrate how performing dense mapping in a relative IRF avoids some of the traditional complications of using a global IRF.

Firstly, we construct an STM map using three sets of dense point measurements, without incorporating any knowledge of the absolute positions of each set of measurements (Section 5.5.2). This experiment simultaneously demonstrates the success of performing dense mapping in a relative IRF for two practical scenarios: the worst-case SLAM scenario, where there is no information about the robot motion (also commonly known as the kidnapped robot problem [50, chap. 7.1]); as well as the scenario where multiple robots are mapping the same region of the environment. From the perspective of the first scenario, this experiment shows how using a relative IRF fits into the SLAM framework, and how it can perform dense mapping even without any pose information (except that which can be calculated from the landmarks). From the perspective of the second scenario, we illustrate the simplicity of using a relative IRF to perform multi-robot mapping.

Secondly, we consider the situation where the robot returns to a previously visited area and loop closure is performed (Section 5.5.3). This experiment demonstrates that, when performing mapping in a relative IRF, the map does not need to be recalculated (as would be required in a global IRF).

5.5.1 Experimental Setup

In order to emulate robustly and persistently identifiable landmarks, we place artificial landmarks in the environment in the form of ArUco markers [82]—a popular open-source library enabling the creation and detection of binary square fiducial markers. Using these artificial landmarks, we perform simultaneous localisation and mapping (SLAM) using the extended Kalman filter (EKF) SLAM algorithm [50, chap. 10.2][5]. From the estimated landmark locations, we create relative IRFs (Section 3.1).

The datasets for these experiments were collected using our RooiBot mobile robotic testing platform (Figure 5.10): an Adept MobileRobots Pioneer 3-AT mobile robot, equipped with two 1.3 MP FLIR Flea3 GigE cameras at a baseline of 18 cm, and a SICK LMS100 2-D LiDAR sensor. The measurements captured by both sensors on the RooiBot were synchronised and stored for offline processing. Data collected from both sensor modalities needs to be preprocessed into 3-D point measurements before being incorporated into an STM map.

Stereo measurement preprocessing We calculated disparity maps (depth images) from image pairs using the library for efficient large-scale stereo matching (LIBELAS) of Geiger et al. [83]. The disparity maps were then transformed to 3-D point measurements using the unscented transform [67].

LiDAR measurement preprocessing Measurements from LiDAR sensors are already 3-D point measurements; however, to relate measurements taken in the LiDAR body reference frame (BRF) to the stereo BRF, we needed to determine the transformation between the BRFs. This process

[5]The EKF-SLAM algorithm performs Kalman filter state estimation, linearising using a first-order Taylor series approximation.

Figure 5.10: The RooiBot mobile robotic testing platform.

is known as extrinsic calibration, and was performed by a novel technique we developed that is based on using a multi-planar calibration board (Appendix A).

The noisy 3-D point measurements from the different sensors modalities were then transformed to the relative IRF of the respective submap (Section 3.2).

5.5.2 Multi-Robot/Kidnapped Robot Mapping

Using the landmarks to define the submapping region, we are able to create an STM map using only three pairs of stereo images (Figure 5.11). The three image pairs were preprocessed to extract 525.3×10^3 point measurements from the three image pairs and these were incorporated into an STM map. Note that the maps created from each stereo image pair represent a robot's perspective (Figures 5.11b to 5.11m).

When considering the resulting STM maps, we see that the height maps are visually consistent with the environment. In the rougher regions of the environment (the gravel heap and landmarks in the corners), we see elevated planar deviations, whereas in the smooth regions (the asphalt path), the planar deviations are low. This illustrates the usefulness of the planar deviation in an STM map that quantifies the surface roughness, which could be used as a measure of terrain drivability for ground robots. Additionally, despite the presence of occlusions in the viewpoints (Figures 5.11c, 5.11g and 5.11k), the resulting height maps in these occluded regions are affected by the inference process, although the map belief in these occluded regions is quite uncertain. This is due to the correlated prior over the surfel vertex heights (Section 5.1). From this experiment, the resulting map belief show that an STM map that is visually consistent with the environment can be constructed from only a few viewpoints. The map is successfully

constructed from stereo camera measurements, which have large range uncertainty. As each submap is created relative to the landmarks in the environment, the submaps are already aligned and can be fused together to form a single belief over the environment (Figure 5.11a).

5.5.3 Mapping with Loop Closure

When localisation is performed by using SLAM, without any absolute position information, the belief over the robot pose becomes increasingly uncertain and, as a result, the mean of the robot belief will drift from the actual pose. When a robot revisits, and recognises, a region of the environment, a large change in the robot pose belief is induced, which is known as loop closure [50, ch. 13.9.2]. This is problematic when performing dense mapping in a global IRF, because measurements at previous time steps are incorporated into the map at an extremely uncertain robot pose and must be reincorporated once loop closure is performed. Otherwise, if the robot pose uncertainty is correctly accounted for, then the incorporated measurements will be useless due to the amount of uncertainty. Jadidi et al. [35] note similar effects, which resulted in the map becoming blurred and essentially useless.

While using the online EKF-SLAM algorithm, we demonstrate how performing mapping in a relative IRF can mitigate the issues associated with loop closure in mapping (Figure 5.12). When loop closure is performed, the estimated robot pose jumps as it recognises a previously visited area, and its uncertainty decreases significantly (Figure 5.12a). If the estimated robot poses are consistent, then incorporating the corresponding LiDAR measurements (Figure 5.12b) into a map in a global IRF would result in a map that is uninformative. This is due to the large uncertainty in the robot pose before the loop closure. However, the LiDAR measurements in a relative IRF (Figure 5.12c) are aligned with the environment (Figure 5.12d), and therefore contain much less uncertainty than in a global IRF—this effect was shown in Section 3.2. Therefore, building an STM map in this relative IRF results in a more accurate map of the environment (Figure 5.12e).

When looking closely at the uncertainty ellipses in Figure 5.12a, one will notice that the uncertainty before loop closure is smaller than what would be expected given the significance in the jump. This overconfidence in the robot pose belief distribution results in what is known an inconsistent distribution. This is typical in EKF-SLAM, as a result of modelling inaccuracies due to linearisation [84, 85]. Inconsistent distributions are, however, an inherent limitation in most SLAM algorithms, as has been shown for unscented Kalman filter (UKF) SLAM [86] and FastSLAM [87]. Using a relative IRF for mapping does not fix the issue of an inconsistent belief distribution, but it does mitigate some of the effects.

5.6 Discussion

The results in this chapter show that STM mapping is a tractable mapping technique for online dense mapping, and that it significantly outperforms elevation mapping—the only comparable explicit surface representation—in terms of modelling accuracy. Additionally, we show that using a relative IRF to build dense maps negates some of the issues that arise when using a global IRF.

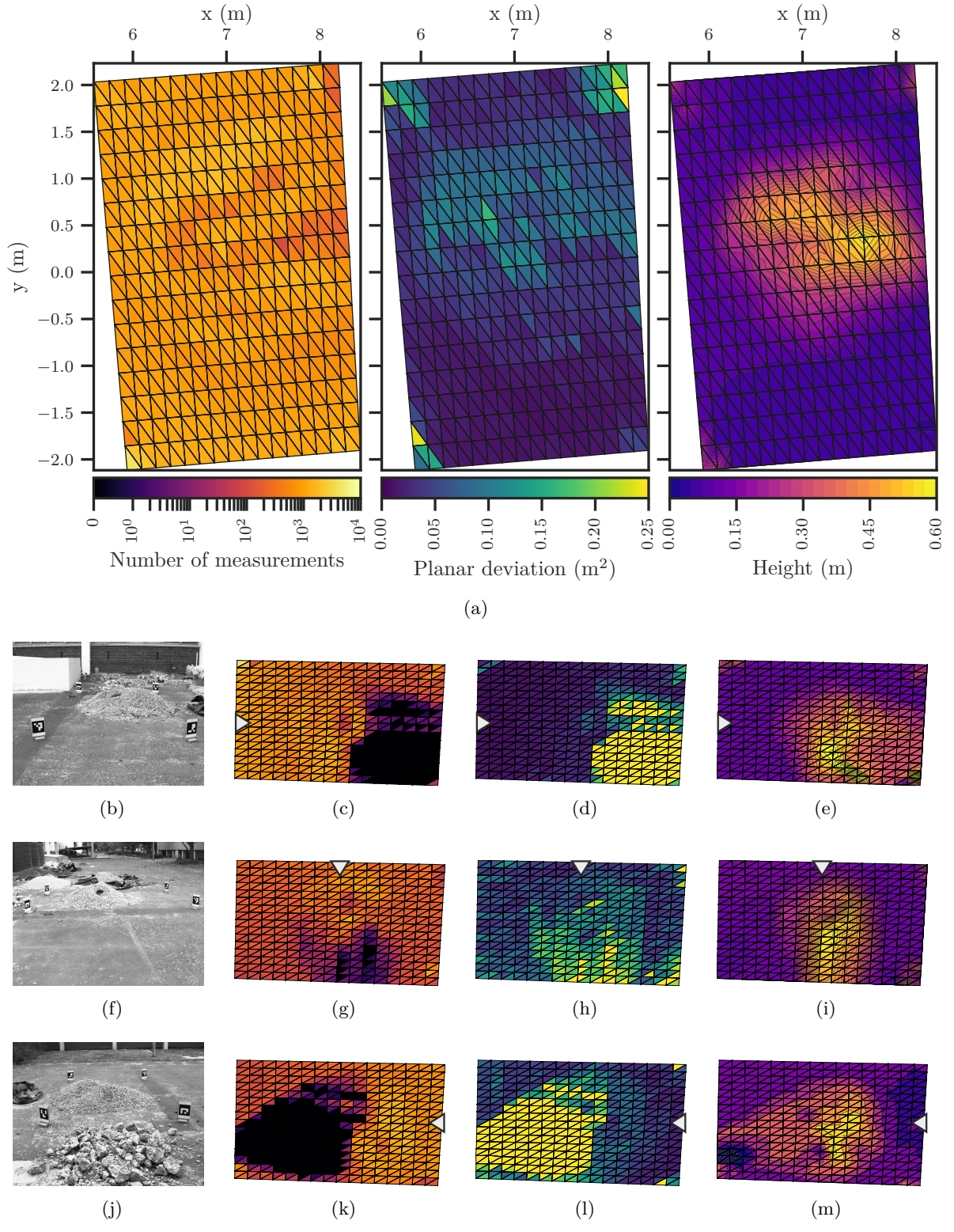
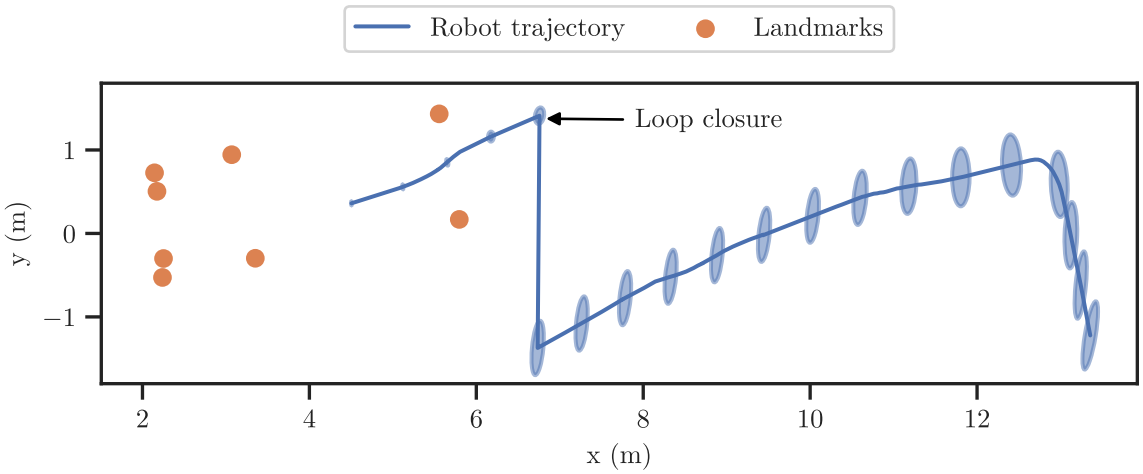
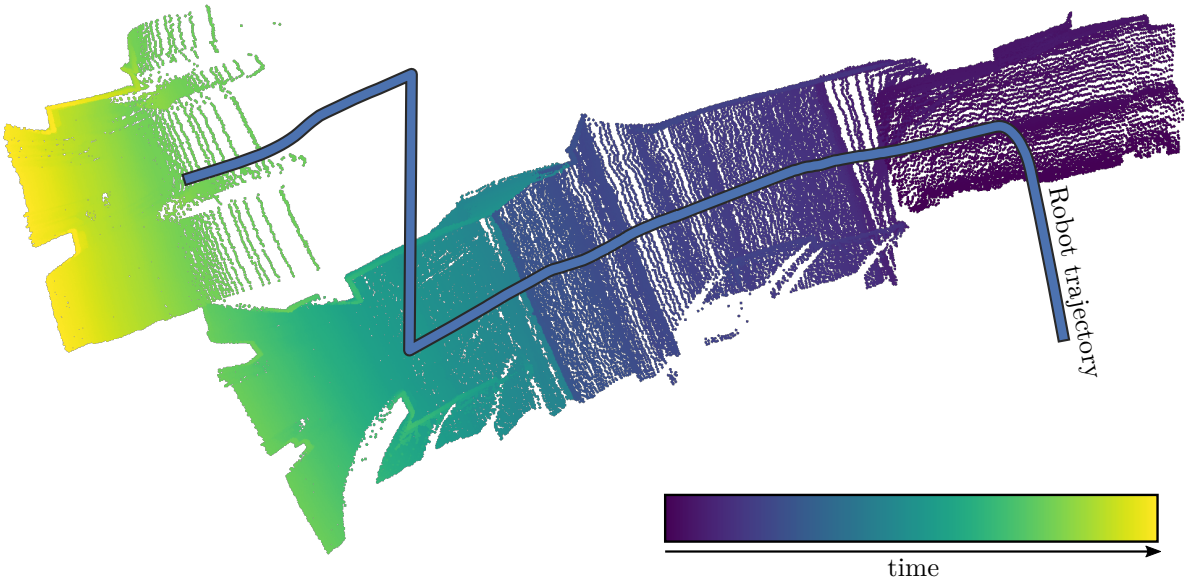


Figure 5.11: An experiment showing an STM map created using only three pairs of stereo images of an outdoor environment. This was made possible by using relative IRFs to construct two submaps, with artificial ArUco markers [82] as landmarks. Each submap contains 256 surfels. (a) The resulting map is shown for the mean of the mean mesh belief and the planar deviation belief—calculated according to Equation 4.28. We also show the number of measurements incorporated into each surfel on a symmetrical logarithmic scale, with a linear region on $[0, 1]$. (b-m) The STM maps built for each stereo image pair—namely (b-e) a front view, (f-i) a side view, and (j-m) a rear view—are on the same scale as (a). The viewpoints are indicated using arrows. Notably, the occlusions are evident due to a lack of measurements (c, g, k). The orientation of (a) is visually consistent with the front view (b)—that is, a 90° rotation of (c-e).

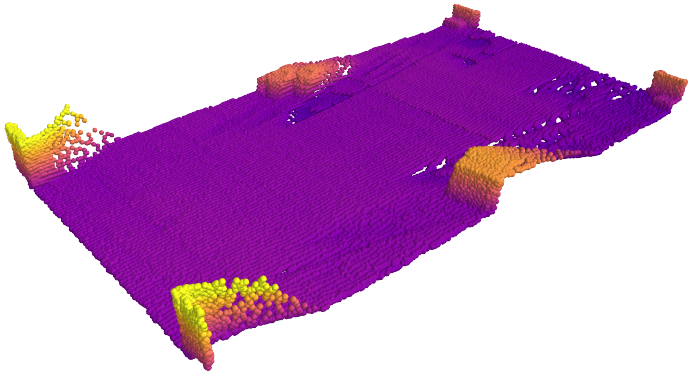


(a)



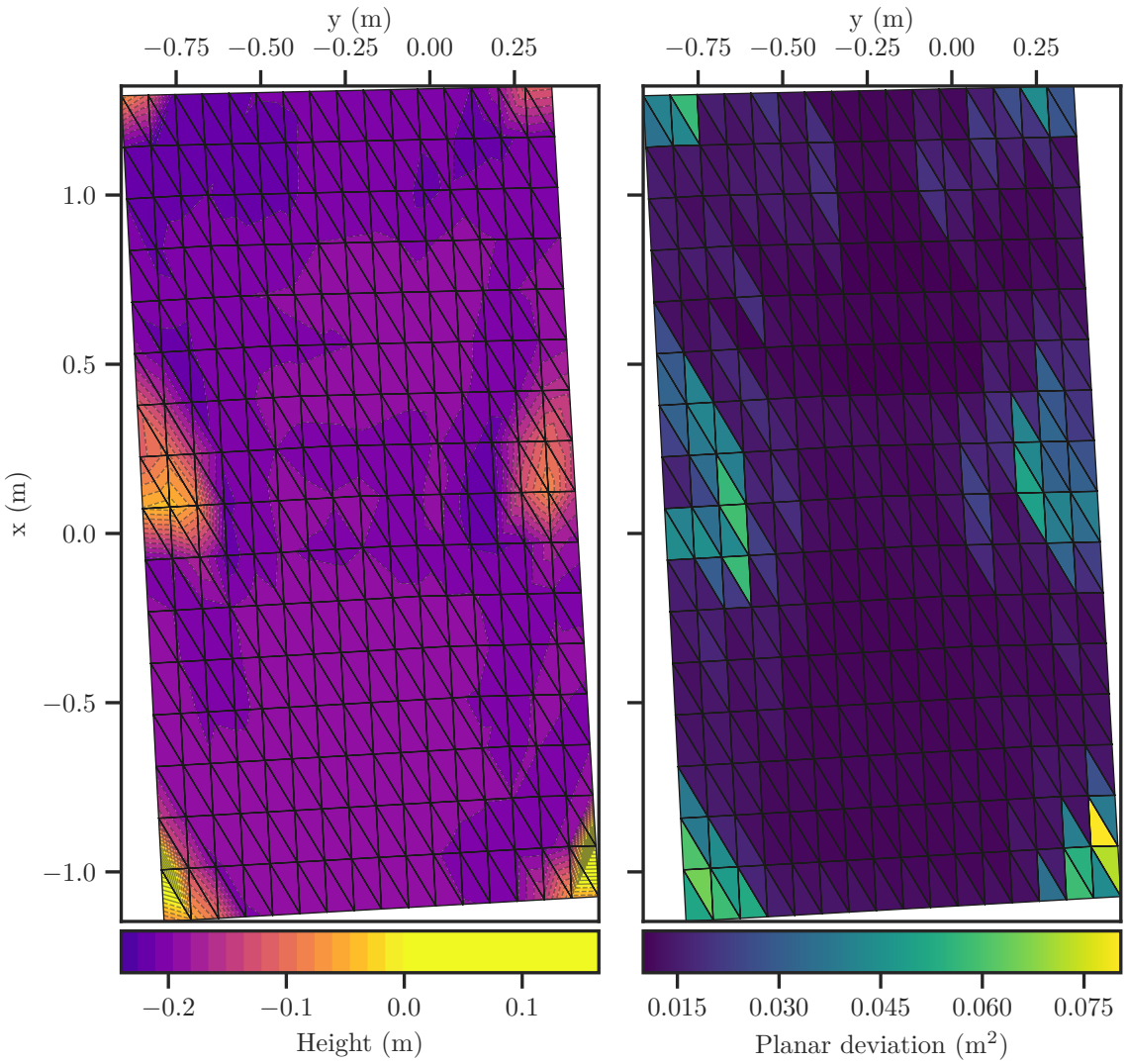
(b)

Figure 5.12: An experiment showing an STM map successfully built when loop closure is performed. (a) The output of the EKF-SLAM algorithm of the mean robot trajectory and final landmark locations clearly shows loop closure being performed. The first standard deviation confidence ellipses are also shown for select robot positions. There is a significant change in the uncertainty in the robot pose after loop closure is performed. (b) The corresponding LiDAR measurement point cloud in the global IRF using the estimated robot trajectory is coloured according to time (darker is older).



(c)

(d)



(e)

Figure 5.12: (*Continued*) (c) The measurement point cloud in the relative IRF is visually consistent with (d) the view of the environment, and (e) the height and planar deviations of the resulting STM map.

6 Conclusion

The problem of incrementally building a dense map of the environment as an autonomous mobile robot navigates, is critical to enabling autonomous mobile robots to effectively operate in any environment. In this dissertation, we set out to develop an online dense mapping technique that would address the shortcomings of the existing dense mapping techniques, and provide an efficient and effective representation of the surface of the environment.

We presented the stochastic triangular mesh (STM) mapping technique, which:

- Represents the structure in the environment using a collection of stochastic processes, forming a continuous representation.
- Is efficient to update, with approximately linear computation time in the number of measurements.
- Is able to handle uncertainty in both measurements and the robot pose.
- Allows online operation by incrementally updating the map.

6.1 Critical Evaluation of STM Mapping

An STM map represents the surface of the environment as collection of triangular surface elements (surfels). Together these surfels create a continuous triangular mesh, which models the average terrain topology as well as the terrain roughness.

In order to keep STM mapping tractable we needed to make some assumptions:

- All measurements of surface of the environment are of *static* regions; that is, we assume that any measurements of dynamic objects are removed before being incorporated into the dense map. This could, for example, be done using the probabilistic graphical model (PGM) based approach by Brink [88].
- Each measurement being incorporated into the dense map can be accurately modelled as being Gaussian distributed. More sophisticated measurement models have been proposed [50, ch. 6.3]; however, these models are used to handle dynamic environments (which is not necessary in our case). Additionally, as these models are only for 1-D range measurements, they are not tractable for full 3-D measurements.
- The surface of the environment within a submapping region is 2.5-D. Note that this assumption does not limit the entire environment to 2.5-D.

- In order to use relative inertial reference frames (IRFs) based on the hybrid metric map (HYMM) framework, landmarks need to be persistently and robustly identifiable. Although this is not currently possible without using synthetic landmarks, we believe this will be possible in the future.

We show that an STM map can handle large-scale mapping; however, due to our implementation being suboptimal, we could not perform a fair comparison on the absolute computational speed. We instead show that the asymptotic computational complexity is approximately linear, which therefore provides evidence that STM mapping is a scalable approach.

Because our problem formulation is quite general, STM mapping is also applicable to all types of autonomous robots. Despite the fact that we only tested STM mapping on practical datasets that were only obtained using ground vehicles, we expect it to be agnostic to the type of mobile robot.

We only tested two of the most common exteroceptive sensors: light detection and ranging (LiDAR) sensors and stereo cameras; however, we expect the results to extend to other sensor modalities—for example depth cameras. This is because we explicitly incorporate the sensor measurement uncertainty into our solution. Therefore, if a sensor can produce point measurements, it should be possible to use it in an STM map.

6.2 Contributions

Although existing techniques have some of the attributes of an STM map, STM mapping is the only technique that is able to achieve them all. We have shown that an STM map is a more accurate and better representation of the environment than the standard elevation map [17, 18], when evaluated in terms of mean squared error (MSE) and model likelihood. We have also shown that an STM map can be as expressive as a Gaussian process (GP) map—when comparing the log likelihoods of both models—whereas STM mapping is tractable for online dense mapping. Qualitative results from practical datasets show that STM maps provide an accurate and descriptive model of the surface of the environment.

The algorithm to incrementally update an STM map—Algorithm 1—uses a combination of loopy belief propagation (LBP) and variational message passing (VMP) to perform efficient inference, which was demonstrated on a large-scale practical dataset. This combination of message-passing techniques is also, to our best knowledge, the first of its kind.

In addition to presenting the STM mapping technique, we have also demonstrated dense mapping using relative IRFs in the HYMM submapping framework [44, 45], and extended the framework to 3-D. Using this submapping approach largely decouples the process of localisation from that of dense mapping. This negates some of the issues in performing localisation using simultaneous localisation and mapping (SLAM)—specifically, it seamlessly allows the incorporation of measurements from multiple robots into a single, consistent representation, mapping between robot poses when the absolute position of the robot is unknown, or mapping when loop closure is performed. The HYMM submapping framework also complements the STM mapping technique, because it uses triangular submapping regions that seamlessly incorporate the triangular surfels in an STM map.

6.3 Future Work

The environment often consists of large, homogeneous regions, and naively dividing a dense map into fixed-sized elements is an inefficient use of storage. Therefore, in order for STM maps to better model the environment, future work will look at adaptively subdividing the grid. Surfels with very low planar deviations could be grouped into a coarser resolution and, conversely, surfels with high planar deviations could be subdivided into a finer resolution. This, however, would add complexity to the inference process.

As most environments with practical significance are not static, it would be desirable to relax the static environment assumption. Future work will look at incorporating temporal information into the map to facilitate this.

Although we propose the terrain roughness as a possible metric for drivability analysis, we do not demonstrate this. As a motivation for STM maps to be used in the context of autonomous navigation, future work will look at specialised planning algorithms to exploit the rich environment representation of an STM map.

We currently treat each submap independently. However, as each submap lies on a plane defined by the landmarks at its vertices, the partitioning of 3-D space between neighbouring submaps will contain overlapping- or dead-zones. Future work could look at a handling the boundaries between submaps accordingly.

An STM map is a 2.5-D representation of the surface of the environment; however, to handle 3-D general environments, future work will look at extending the representation accordingly. As each STM submap is associated to the plane defined by the landmarks vertices, a solution could be to use a more sophisticated strategy of submapping, and then handling the boundaries as mentioned previously. Alternatively, a 3-D triangular meshing strategy could be investigated; however, this will most likely be intractable for online dense mapping if done in a fully probabilistic manner—as with STM mapping.

A Extrinsic Calibration

In this appendix we reproduce our paper detailing the extrinsic calibration between a light detection and ranging (LiDAR) sensor and camera, which appeared in the proceedings of the 2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech) [48].

Extrinsic Calibration of a Push-broom LiDAR and Camera Using 3-D Multi-planar Association

Abstract

In this paper we propose a method of extrinsically calibrating two of the most commonly used sensors in autonomous robotic systems: a 2-D lidar sensor and camera. Using a tri-planar calibration target, our method extracts rich point and planar feature information from the measurements of both sensors. We approach the problem of estimating the extrinsic parameters using optimisation, and by minimising the direct 3-D point-plane correspondences between these features, are able to accurately estimate the parameters. Our method is shown to be visually accurate and robust.

A.1 Introduction

Most autonomous robotic systems rely on measurements from multiple sensors about their environment for effective operation. These measurements are taken with respect to the relative reference frame of each sensor. In order to fuse the information from all these sources, the measurements first need to be transformed into a common reference frame. This requires the relative transformation between each sensor's reference frame, and the common reference frame, to be determined. A process known as extrinsic calibration.

In this work we focus on the extrinsic calibration of two of the most widely used sensors for autonomous robots: LiDAR sensors and cameras. Specifically, we consider the calibration of a 2-D LiDAR mounted in a pushbroom configuration, and a monocular camera.

It is usually impossible to accurately measure the extrinsic parameters directly, due to the sensors being in isolated casings; therefore, alternative indirect methods are required in determining the extrinsic parameters. A possible approach is to measure the same features from both sensors, and then calculate the extrinsic parameters by comparing several such feature measurements. However, measuring the same features and identifying the matching features in different sensor measurements becomes difficult if the sensors measure different properties of the environment.

The most popular methods of achieving this utilise a calibration target, where the structure of the target is such that different features are visible for different sensors, and the relative locations of these features are known. However, the extrinsic parameters cannot be easily calculated directly from feature measurements due to measurement uncertainty and non-linearities in the transformation. Therefore, a preferred approach is to optimise for the extrinsic parameters through the minimisation of an error function, which describes the correspondence between the measured features using their relative locations on the calibration target.

This approach was pioneered by Wasielewski and Strauss [89], who used a V-shaped calibration target with black and white faces. The contrasting faces in camera measurements were used to determine a line feature defined by the middle corner, and the maximum curvature in LiDAR measurements were used to identify the middle corner of the target as a point feature. By reprojecting the point feature onto the image plane, the point-line feature correspondence was used with the Gauss-Newton algorithm to optimise for the extrinsic parameters. A similar method was used by Li et al. [90], using a planar triangular calibration target. The edges of the target were used to determine two point and line features, from LiDAR and camera measurements respectively. Kwak et al. [91] also used a V-shaped calibration target, and extracted three point and line features, from outer and middle corners of the target. Using the feature correspondence in the image plane, they used a robust optimisation strategy to determine the extrinsic parameters. Their method, verified using an IR-camera as a ground truth, was shown to be more accurate when compared to the methods of Wasielewski and Strauss [89] and Li et al. [90]. However, the disadvantage with these methods [89, 90, 91] is that the point features extracted from the LiDAR measurements are not direct measurements of the edges of the respective calibration targets (as used in the camera features), due to discrete angular range steps in the LiDAR measurements; although Kwak et al. [91] inserted a virtual point on the edges of their LiDAR measurements to slightly improve this accuracy.

An alternative approach by Zhang and Pless [92] used a single planar checkerboard as a calibration target. They estimated the parameters describing the 3-D plane in the camera reference frame, and manually identified segments of LiDAR points that corresponded to the plane. The extrinsic parameters were then optimised by minimising the 3-D Euclidean distance between the plane and point features. Their method required at least six measurement pairs. Using the same calibration target, the methods of Vasconcelos et al. [93] and Zhou [94] improved this minimum requirement to only three measurement pairs.

Ha [95] used a precisely modelled checked calibration plane with a triangular hole, and from this was able to directly estimate the position of the LiDAR points on the target. The extrinsic parameters were calculated using direct 3-D point-point correspondence between features in camera and LiDAR measurements. This method was able to find a solution with a minimum of two pairs of measurements, without any optimisation techniques. A significant disadvantage, however, is the requirement that the physical dimensions of calibration target be known precisely.

One drawback with using a calibration target is the inability to update and improve calibration once the target is no longer in view, limiting its use in online applications. To avoid these limitations, other methods use natural scenes directly for calibration. This makes the problem of data association challenging, since the shape and appearance of the observed scene are not known beforehand. This can be easily solved by using a 3-D LiDAR, as it provides an instantaneous dense point cloud of the environment. Pandey et al. [96] created a 3-D point cloud with the raw LiDAR reflectivity, and by projecting onto the image plane, aligned scans with the pixel intensities. Napier et al. [97], and Scott et al. [98] similarly used reflectivity to calibrate a 2-D LiDAR in natural scenes. The drawback, however, to these methods is that they rely heavily on accurate motion estimates to create consistent reflectivity point clouds.

Our method takes an approach which combines favourable aspects from Zhang and Pless

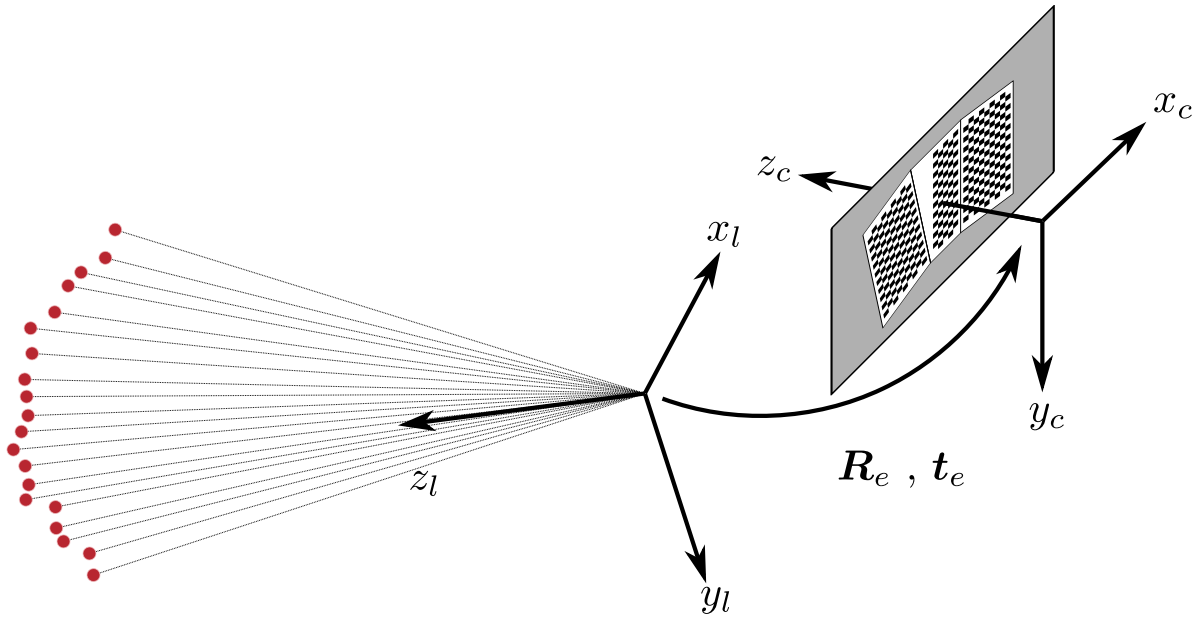


Figure A.1: The rigid body transformation between the LiDAR and camera reference frames—described by $\boldsymbol{R}_e$ and $\boldsymbol{t}_e$—allows features to be combined into a single common reference frame.

[92], together with Kwak et al. [91]. We designed a concave tri-planar calibration target with checkerboard patterns on each plane. By extracting 3-D descriptions of the planes using the camera, we are able to optimise for the extrinsic parameters by accurately aligning the LiDAR measurements to these planes. Unlike the method of Kwak et al., we use direct feature correspondence, similar to that of Zhang and Pless [92]. However, through the use of a more complex calibration target design we are able to enhance the amount of information captured in each pair of measurements. Additionally, our method of preprocessing camera measurements for features requires no human supervision, compared to the manual feature selection used by Kwak et al. [91]. As opposed to that of Ha [95], our approach places very few limitations on the design of the calibration target, making it robust to variations in the mechanical manufacturing thereof.

The subsequent sections of this paper are structured as follows. Next we will look in detail at the problem of extrinsic calibration, specific to a 2-D LiDAR sensor and camera. In Appendix A.3 we present our approach to solving this problem. In Appendix A.4 we describe the experimental setup used. We then present the experimental results of our method in Appendix A.5. Finally, we conclude in Appendix A.6.

A.2 Problem Description

The purpose of extrinsic calibration is to calculate the set of parameters describing the relative rigid body transformation between the two sensors' reference frames, and a common reference frame. From this set of extrinsic parameters, one can transform observed features from each sensor's reference frame to the common reference frame. However, most often the common reference frame is chosen as one of the sensor reference frames. In our case we assume this to be the camera reference frame, as shown in Figure A.1.

In order to describe the extrinsic parameters, consider a three-dimensional feature point, $\boldsymbol{p}_l$, in the LiDAR reference frame. This point is transformed to the camera reference frame, resulting

in the feature point, $\boldsymbol{p}_c$. This can be expressed by

$$\boldsymbol{p}_c = \boldsymbol{R}_e \cdot \boldsymbol{p}_l + \boldsymbol{t}_e. \tag{A.1}$$

Here $\boldsymbol{R}_e$—a 3×3 rotation matrix in Euclidean space—and $\boldsymbol{t}_e = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^\top$—a translation vector—fully describe the relative three-dimensional rotation and translation between the LiDAR and camera reference frames.

Three-dimensional rotations are usually parameterised in one of three ways: Euler angles, quaternions, or Rodrigues' rotation formula. We choose to use the Euler angle parameterisation, as it is intuitive to understand. This parameterisation of the rotation matrix can be viewed as a sequence of rotations:

$$\boldsymbol{R}_e(\alpha, \beta, \gamma) = \boldsymbol{R}_x(\alpha) \cdot \boldsymbol{R}_y(\beta) \cdot \boldsymbol{R}_z(\gamma), \tag{A.2}$$

where $\boldsymbol{R}_x(\alpha)$, $\boldsymbol{R}_y(\beta)$, $\boldsymbol{R}_z(\gamma)$ are, respectively, the elementary rotation matrices describing the rotations about the x-, y-, and z- axes of the LiDAR reference frame, according to the angles α, β and γ.

Based on the chosen parameterisations, the required extrinsic parameters which fully describe the transformation between sensor reference frames are: α, β, γ, t_x, t_y, and t_z. In this paper we propose a method of estimating these parameters accurately, from a set of noisy feature observations, which are assumed to contain information indirectly coupled to the extrinsic parameters.

A.3 Methodology

Our approach to the problem of extrinsic calibration can be separated into various aspects: the design of the calibration target, the choice of feature descriptions and correspondence, measurement preprocessing, and finally the estimation of the extrinsic parameters. This method is an amalgamation of those proposed by Zhang and Pless [92], and Kwak et al. [91].

A.3.1 Calibration Target Design

Our method requires the use of a calibration target that has features visible to both sensors. To satisfy this critera, we designed a tri-planar, concave calibration target with precisely printed checkerboard patterns on each plane (Figure A.2). The camera is easily able to identify the checkerboard patterns, while the LiDAR is able to detect the planar arrangement. The checkerboard patterns were designed to contain as many blocks as possible, without the blocks being too small. The angles between the left and right outer planes, with respect to the centre plane, were chosen to be 140° and 130°, respectively. This was based on the idea that a LiDAR scan from a certain viewpoint should have a unique shape.

A concave planar configuration was choosen—as opposed to convex—as it would allow the camera to view the planes from a wider range of angles (Figure A.3). This reasoning is based on the common practice in camera calibration that a checkerboard pattern cannot be easily identified when observed from extreme angles. From this, we imposed the restriction that the angle between the light ray from the point on the plane, to the camera centre and the checkerboard plane, should not be more than 45°, as shown in Figure A.3a. From this restriction, the region where the full calibration target would be visible, can be calculated as the union of the overlapping visible regions of each individual plane[1]—for both the convex and concave cases. As shown in

[1] We make the assumption that the field of view of the camera is large enough to completely view the calibration target from the visible region.

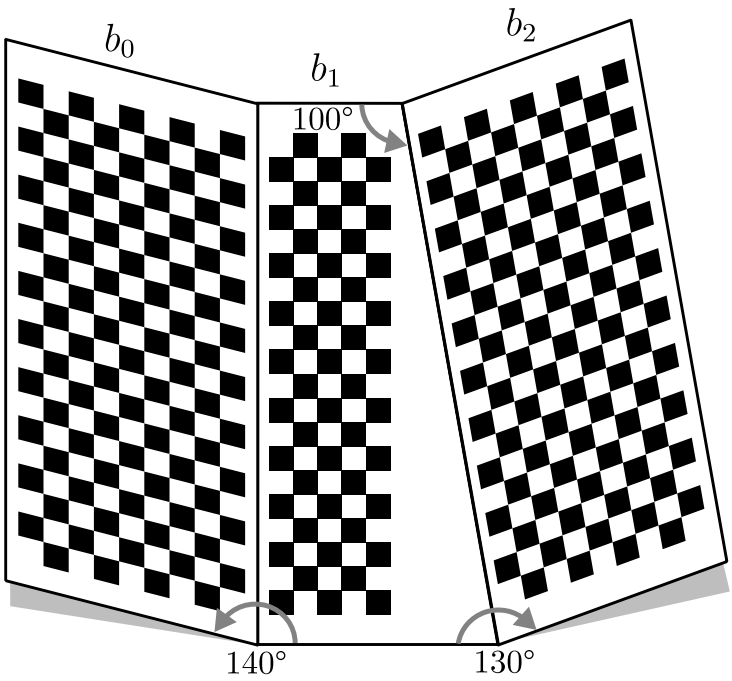
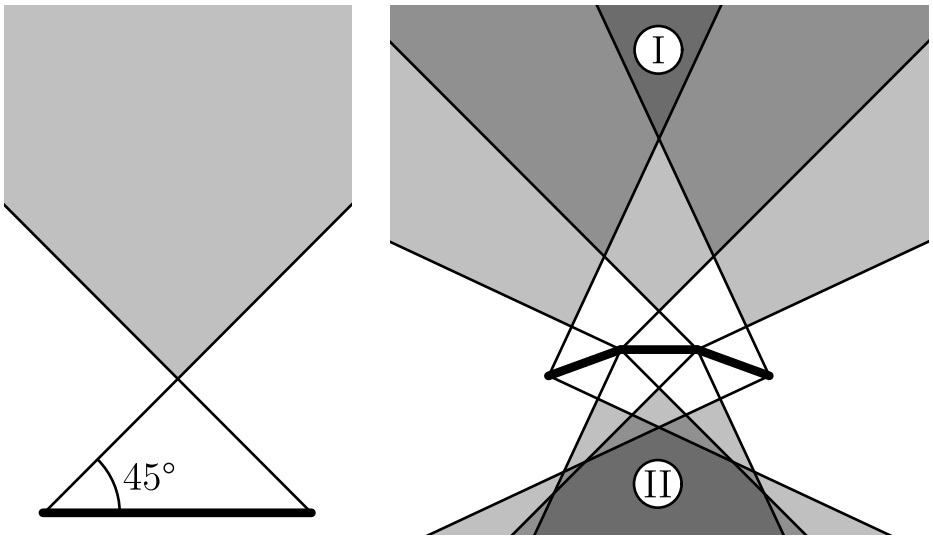


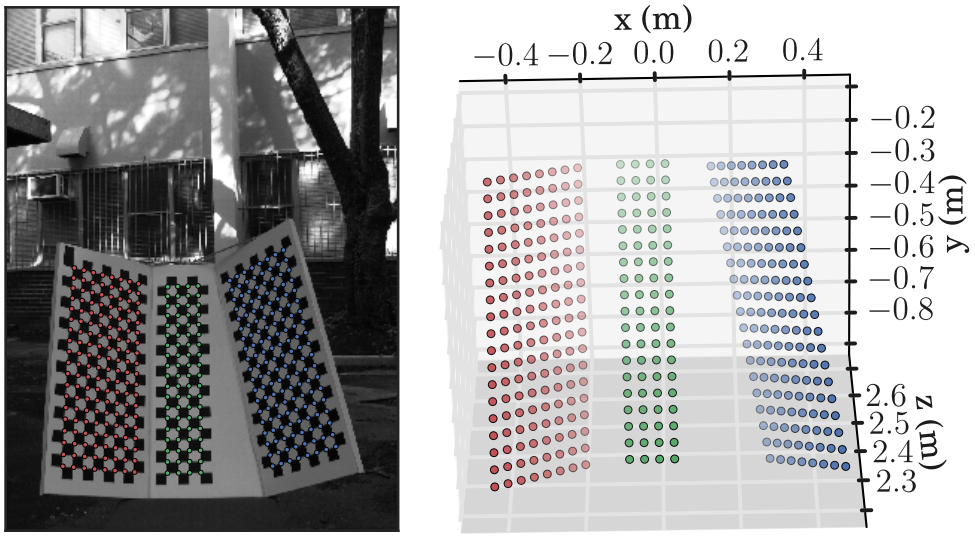
Figure A.2: The designed calibration target, a concave arrangement of three planar boards, each printed with a precise checkerboard pattern. The angular dimensions were used as approximate guidelines in the manufacturing process.



(a) (b)

Figure A.3: (a) The maximum viewing angle for the visible region of each plane is restricted to 45°. (b) The combination of the overlapping visible regions of each plane for both the convex (I) and concave (II) cases, shows that it is preferable to use concave arrangement of the planes as it creates a larger visible region.

Figure A.3b, the visible region for concave configuration is much larger than that of the convex case. Also, if the outer planes are angled at less than 135° with respect to the centre plane, then the convex configuration would no longer have any overlapping visible region.



(a) (b)

Figure A.4: (a) The identified 2-D corners of each checkboard on the calibration target can be projected to 3-D (b) using the calculated planar feature parameters.

A.3.2 Feature Descriptions and Correspondence

As it is not possible to determine the exact points on the calibration target that the LiDAR is measuring, direct point-point feature correspondence between LiDAR and camera measurements is not possible. Alternatively, we utilise the fact that the LiDAR measurements are generated from the planes of the calibration target. The three-dimensional position and orientation (pose) of these planes can also be extracted from camera measurements (Section A.3.3), as well as the corresponding sets of points associated with the relevant planes from LiDAR measurements (Section A.3.4). Consequently, this information allows direct feature correspondence of the planar features with point features.

A.3.3 Preprocessing Camera Measurements

In order to extract the 3-D poses describing the planar features from the 2-D camera measurements, the intrinsic (internal) parameters of the camera first need to be determined. We model the camera using the pinhole camera model [99], and estimate the intrinsic parameters of the camera using the process detailed by Zhang [100]. Importantly, we assume that the intrinsic parameters have been estimated to a far greater degree of accuracy than that of the extrinsic parameters.

Once the intrinsic parameters have been determined the next problem lies in measuring all three planar features on the calibration target. We approach this task as the Perspective-n-Point (PnP) problem [101], individually for each plane. The 2-D corners of the checkerboard are identified from a camera measurement (Figure A.4a), along with the relative locations of the corresponding 3-D points from the known planar structure of each checkerboard. The 3-D pose of the plane is then optimised by minimising the reprojection error of the 3-D points using the Levenberg-Marquardt (LM) algorithm [102, 103]. From this, the pose of each planar feature can be described – similarly to that in Equation A.1—by a rotation matrix, $\boldsymbol{R}_i$, and translation vector, $\boldsymbol{t}_i$, where i refers to the relevant plane. The results of this process are visualised in Figure A.4b.

A.3.4 Preprocessing LiDAR Measurements

The LiDAR point features[2] need to be extracted from each scan, specifically, the sets of points which correspond to the planes of the calibration target. However, each LiDAR scan contains information of both the calibration target and the environment, of which the latter is clutter. It is therefore required that only information pertinent to the calibration target be extracted from each scan, which we perform manually, as shown in Figure A.5a.

Next, these points need to be associated with the relevant planes on the target. In order to achieve this, a piece-wise linear curve—comprising of three segments—is fit to the data. This is done using the LM algorithm [102, 103], by minimising the square of the Euclidean distance of each point to its associated segment. An initial estimate for the optimisation is specified based on the average plane widths, and the designed angular separation. From the result of the optimisation (Figure A.5b), the points are segmented into the feature sets $\mathcal{X}_i$, where i refers to the associated planar feature in the corresponding camera measurement.

A.3.5 Parameter Estimation

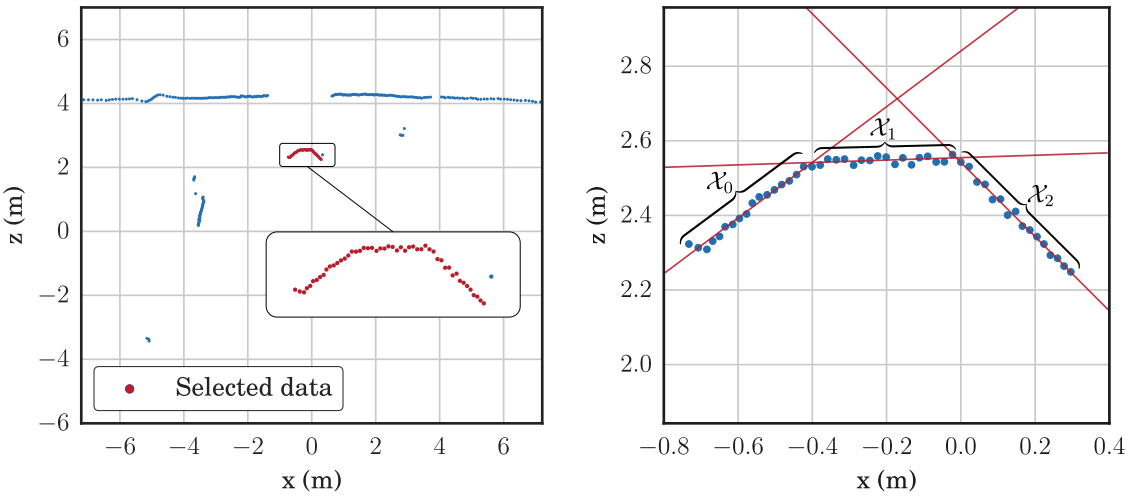
From the extracted point and planar features, we estimate the extrinsic parameters (as defined in Section A.2):

$$\boldsymbol{\theta} = \begin{bmatrix} t_x & t_y & t_z & \alpha & \beta & \gamma \end{bmatrix}^\top, \tag{A.3}$$

by minimising an error function describing the point-plane feature correspondence. This error is constructed using the Euclidean distance between the LiDAR point features and associated camera planar features in 3-D. For a single LiDAR point feature, $\boldsymbol{x}_i^k \subset \mathcal{X}_i^j$, associated with the i-th planar feature description, $\boldsymbol{R}_i^j$ and $\boldsymbol{t}_i^j$, from the j-th measurement pair. We can calculate the point-plane Euclidean distance as:

$$d_i^k(\boldsymbol{\theta}) = ||\boldsymbol{R}_i^{j\top}(\boldsymbol{R}_e(\boldsymbol{\theta}) \cdot \boldsymbol{x}_i^k + \boldsymbol{t}_e(\boldsymbol{\theta}) - \boldsymbol{t}_i^j) \cdot \boldsymbol{k}||, \tag{A.4}$$

[2]In order to represent these features in 3-D the LiDAR measurements are assumed to be in the xz-plane as shown in Figure A.1.



(a) (b)

Figure A.5: The process of segmenting a single scan into the three planes of the calibration board. (a) First the calibration board must be manually selected out of the raw LiDAR data. (b) Second, three straight lines are fit to the data, and the points are segmented into different sets, $\mathcal{X}_i$, based on the fit.

where $\boldsymbol{k} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$. This distance metric is, however, not robust to outliers. Therefore, similarly to the method by Kwak et al. [91], we apply the Huber loss penalty [104], a technique used frequently in robust non-linear regression to reduce sensitivity to outliers. This is achieved by using an error function which scales quadratically for small errors, and linearly for large errors, expressed by

$$H(d) = \begin{cases} d^2, & \text{if } |d| < \Delta \\ \Delta(2|d| - \Delta), & \text{otherwise} \end{cases}, \tag{A.5}$$

where Δ defines the changeover distance between the quadratic and linear scaling. We chose Δ as 0.05 m based on the noise levels observed on the LiDAR data.

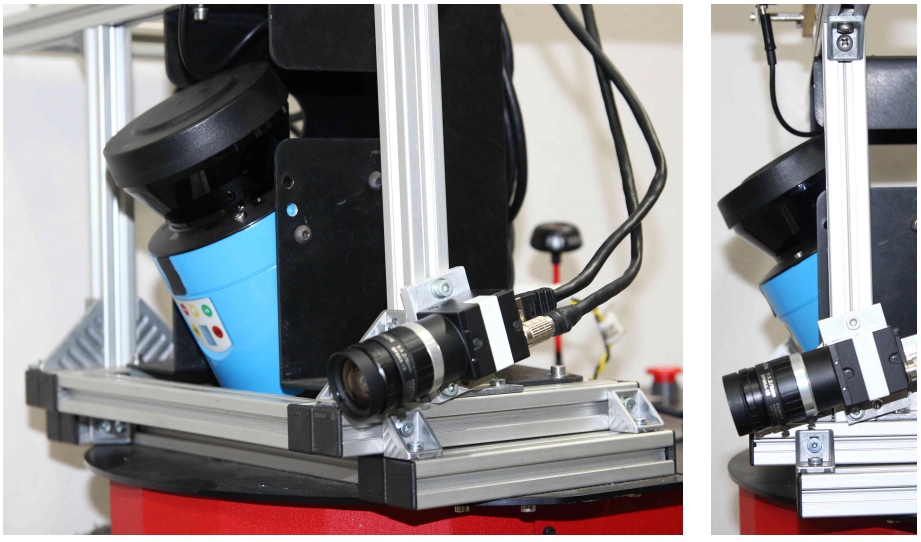
Using this robust distance metric, the complete error function for a set of N measurement pairs is calculated as:

$$E(\boldsymbol{\theta}) = \sum_{i=0}^{2} \sum_{j=1}^{N} \sum_{k=1}^{|\mathcal{X}_i^j|} H(d_i^k(\boldsymbol{\theta})). \tag{A.6}$$

This error function is minimised using the LM algorithm [102, 103], where an initial estimate of the extrinsic parameters is approximated from physical measurements of the setup.

A.4 Experimental Setup

We verified our method using a pratical setup consisting of a SICK LMS100 LiDAR and a FLIR Flea3 GigE camera. The LiDAR has an angular resolution of 0.5°, and is mounted in a push-broom configuration—tilted at approximately 30°. The camera is fitted with a 6 mm lens, providing a vertical and horizontal field of view (FOV) of 32.40° and 43.60°, respectively. Due to the tilt of the LiDAR, the camera's vertical FOV is too small for there to be any overlap between both sensor views. Therefore, the camera was turned on its side to utilise the horizontal FOV, and angled by approximately 16° downwards. This setup can be seen in Figure A.6.



(a) (b)

Figure A.6: The practical setup showing the configuration of a SICK LMS100 LiDAR and a FLIR Flea3 GigE camera with a 6 mm lens.

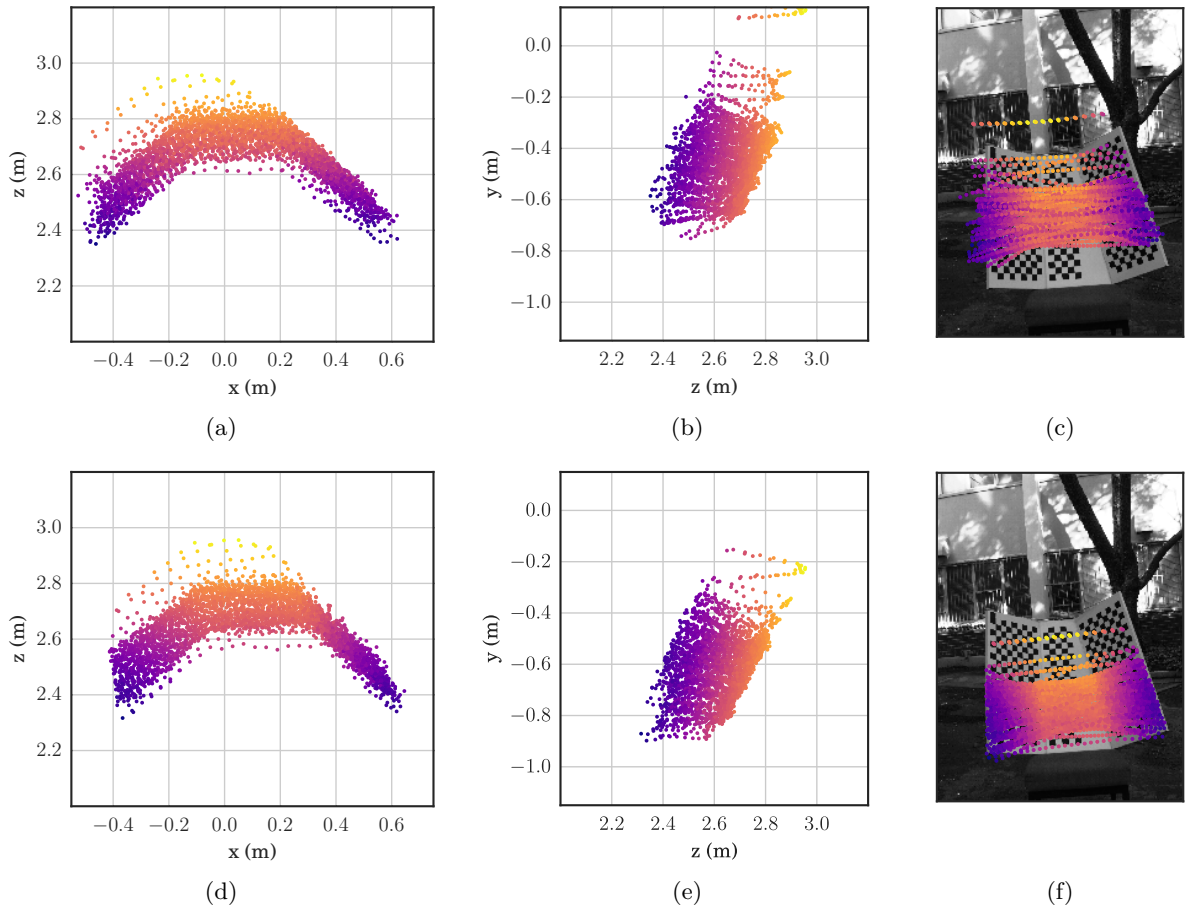


Figure A.7: A 3-D reconstruction of the calibaration target, from the training dataset, is generated using the extrinsic parameters obtained from both the physically measured (top row), and our calibration method (bottom row). The projection of the reconstuction is shown for the $zx-$ (a, d) and $yz-$ (b, e) axes, whereas for the $xy-$axis, the points are reprojected onto an image from the training dataset (c, f). The LiDAR point ranges are colour-mapped from violet (near) to yellow (far). The physically measured parameters result in a misalignment of the LiDAR scans with the calibration target (c), which is remedied by the optimised parameters (f).

A.5 Experimental Results

The performance of the proposed extrinsic calibration method is evaulated on a calibration dataset and a range of real-world datasets.

A.5.1 Calibration Dataset

A calibration dataset was obtained with a total of 90 valid lidar-camera measurement pairs, each with an average of 13 LiDAR points per plane. In order to capture as much information about the extrinsic parameters in the dataset as possible, a wide range of different poses of the calibration target were taken. The dataset was randomly split into training (70 %) and test (30 %) subsets. Using the training data, the extrinsic parameters were estimated based on our proposed method.

As it was not possible to obtain a ground truth, we first compare our method against the physically measured values used as the initialisation in the optimisation process. The differences

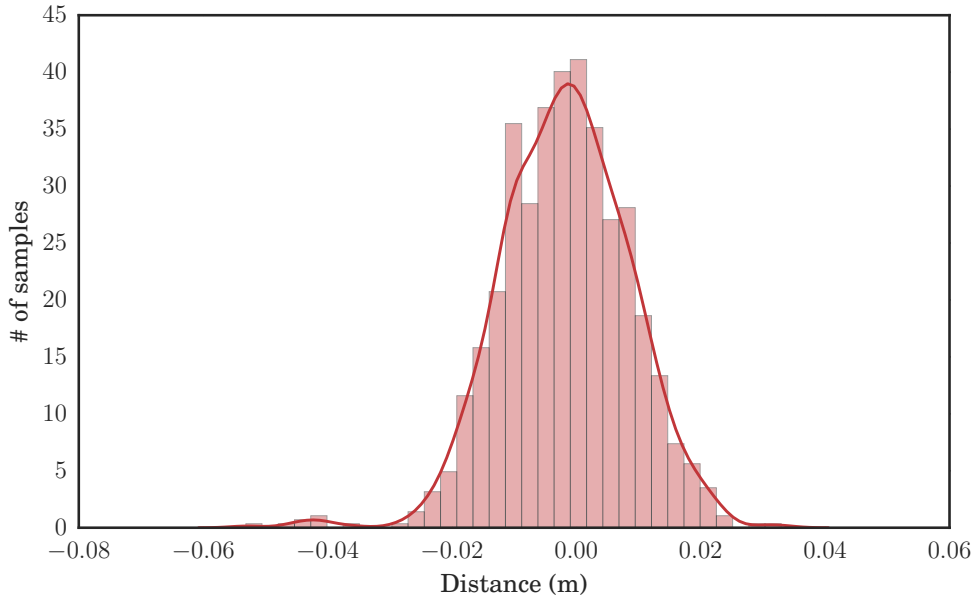


Figure A.8: The experimentally determined distribution of the distances of LiDAR points to their associated planes for the test dataset. The resulting distribution has a mean of −0.1760 mm, and standard deviation of 10.24 mm.

Table A.1: Comparison of the Measured and Calibrated Values of the Extrinsic Parameters.

Parameters	t_x (mm)	t_y (mm)	t_z (mm)	α (°)	β (°)	γ (°)
Measured	100	186.8	1.552	0.	−16.00	90.00
Calibrated	108.7	182.3	10.82	−1.108	−19.55	87.90
Difference	−8.712	4.515	−9.270	1.108	3.548	2.104

shown in Table A.1 may seem small, but are not negligible. This becomes very evident when a 3-D reconstruction of the calibration target from the training data is shown for both cases in Figure A.7. There is a clear difference in accuracy between the two cases, with our calibration method yielding a consistent reconstruction of the calibration target.

The extrinsic calibration results are also analysed on the test dataset. We evaluated the distribution of the distances of the LiDAR points to planes extracted from the camera measurements using parameters estimated from the training dataset. The experimentally determined distribution, as shown in Figure A.8, is aquired using the process of histogram density estimation, where the bin width is chosen according to Freedman and Diaconis [105]. The resulting distribution has a mean and standard deviation of −0.1760 mm and 10.24 mm, respectively. The mean is small enough to be considered zero, and the standard deviation almost matches the reported statistical noise of the LiDAR sensor, 12 mm. This error distribution is close to the expected error distribution of the LiDAR alone; the extrinsic calibration and camera intrinsic calibration errors are therefore not a major component in the error distribution.

A.5.2 Real-world Verification

A series of datasets were taken in real-world scenes to further verify the results of the calibration. The datasets were obtained on different days, and under different conditions to verify the

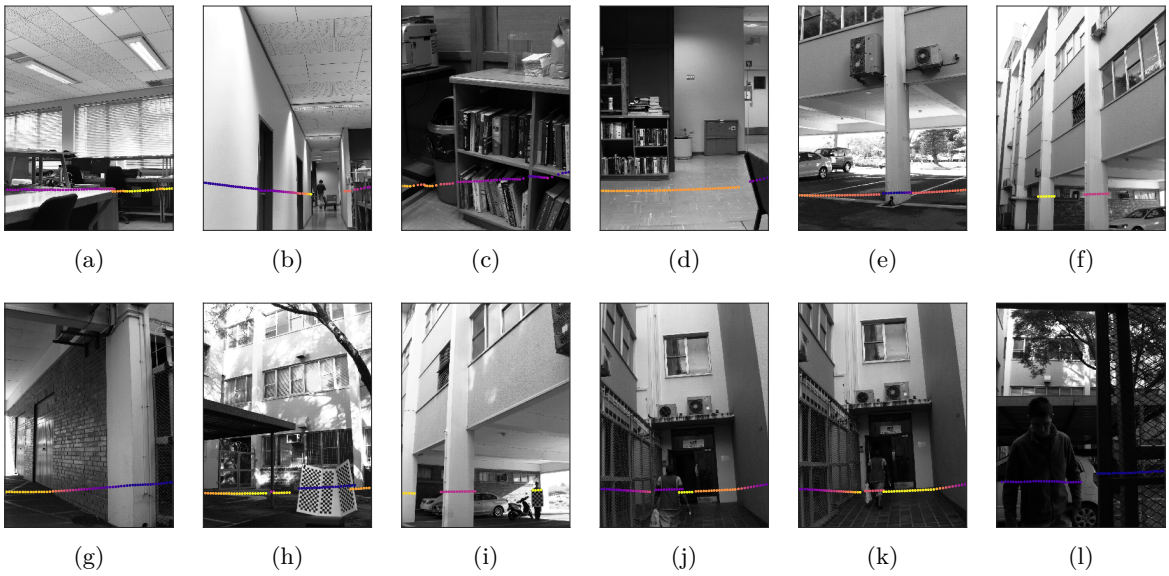


Figure A.9: A series of real-world datasets were obtained to verify the extrinsic parameter calibration. Each dataset was obtained under different conditions: one was obtained indoors (a-d), another two outdoors under different lighting (e-h and i-l). Selected camera and LiDAR measurements are visualised. The LiDAR measurements are reprojected onto the image plane, and the ranges are colour-mapped from violet (near) to yellow (far) to indicate the ranges more clearly.

robustness over time. Using serveral measurement pairs from each of the datasets, the LiDAR points are projected onto the image using the transformation determined by extrinsic calibration. The results shown in Figure A.9 appear to be visually accurate. There also does not appear to be any loss in calibration across the different days, highlighting robustness of the method.

A.6 Conclusion

In this paper, we proposed and experimentally evaluated a method to perform extrinsic calibration between a 2-D LiDAR and camera. Our method utilises a custom designed tri-planar calibration target to extract rich point and planar feature information. This is used to optimise for the extrinsic parameters via point-plane feature correspondence in 3-D. The results are shown to be visually consistent and accurate, and clear increase in accuracy was achieved compared to approximate physical measurement of the parameters.

Bibliography

[1] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the sky: Leveraging UAVs for disaster management," *IEEE Pervasive Computing*, vol. 16, no. 1, pp. 24–32, Jan 2017. [Cited on page 1.]

[2] M. Ono, T. J. Fuchs, A. Steffy, M. Maimone, and J. Yen, "Risk-aware planetary rover operation: Autonomous terrain classification and path planning," in *2015 IEEE Aerospace Conference*. IEEE, Mar 2015. [Cited on page 1.]

[3] M. Daily, S. Medasani, R. Behringer, and M. Trivedi, "Self-driving cars," *Computer*, vol. 50, no. 12, pp. 18–23, Dec 2017. [Cited on page 1.]

[4] C. E. van Daalen, "Conflict detection and resolution for autonomous vehicles," PhD Thesis, Stellenbosch University, 2010. [Cited on page 1.]

[5] V. Guizilini and F. Ramos, "Towards real-time 3D continuous occupancy mapping using Hilbert maps," *The International Journal of Robotics Research*, vol. 37, no. 6, pp. 566–584, May 2018. [Cited on pages 2 and 6.]

[6] S. Thrun, C. Martin, Y. Liu, D. Hahnel, R. Emery-Montemerlo, D. Chakrabarti, and W. Burgard, "A Real-Time Expectation-Maximization Algorithm for Acquiring Multiplanar Maps of Indoor Environments With Mobile Robots," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 433–442, Jun 2004. [Cited on page 3.]

[7] T. Wiemann, K. Lingemann, and J. Hertzberg, "Optimizing Triangle Mesh Reconstructions of Planar Environments," *IFAC-PapersOnLine*, vol. 49, no. 15, pp. 218–223, 2016. [Cited on page 3.]

[8] J. Zienkiewicz, A. Tsiotsios, A. Davison, and S. Leutenegger, "Monocular, Real-Time Surface Reconstruction Using Dynamic Level of Detail," in *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, Oct 2016, pp. 37–46. [Cited on page 3.]

[9] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1985, pp. 116–121. [Cited on page 3.]

[10] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, Apr 2013. [Cited on page 3.]

[11] E. Einhorn, C. Schroter, and H.-M. Gross, "Finding the adequate resolution for grid mapping - Cell sizes locally adapting on-the-fly," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2011, pp. 1843–1848. [Cited on page 3.]

[12] S. Khan, D. Wollherr, and M. Buss, "Adaptive rectangular cuboids for 3D mapping," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2015, pp. 2132–2139. [Cited on page 3.]

[13] D. Droeschel, M. Schwarz, and S. Behnke, "Continuous mapping and localization for autonomous navigation in rough terrain using a 3D laser scanner," *Robotics and Autonomous Systems*, vol. 88, pp. 104–115, Feb 2017. [Cited on pages 3 and 20.]

[14] D. Joubert, W. Brink, and B. Herbst, "Pose Uncertainty in Occupancy Grids through Monte Carlo Integration," *Journal of Intelligent & Robotic Systems*, vol. 77, no. 1, pp. 5–16, Jan 2015. [Cited on

page 3.]

[15] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous Robots*, vol. 15, no. 2, pp. 111–127, 2003. [Cited on page 3.]

[16] M. Hebert, C. Caillas, E. Krotkov, I. Kweon, and T. Kanade, "Terrain mapping for a roving planetary explorer," *IEEE International Conference on Robotics and Automation*, pp. 997–1002, 1989. [Cited on page 4.]

[17] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *IEEE International Conference on Intelligent Robots and Systems*. IEEE, Oct 2006, pp. 2276–2282. [Cited on pages 4, 46, 48, 49, and 63.]

[18] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic Terrain Mapping for Mobile Robots with Uncertain Localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3019–3026, 2018. [Cited on pages 4, 20, 46, 48, 48, 49, and 63.]

[19] T. K. Marks, A. Howard, M. Bajracharya, G. W. Cottrell, and L. H. Matthies, "Gamma-SLAM: Visual SLAM in unstructured environments using variance grid maps," *Journal of Field Robotics*, vol. 26, no. 1, pp. 26–51, Jan 2009. [Cited on pages 4 and 9.]

[20] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136. [Cited on page 4.]

[21] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense RGB-D SLAM with volumetric fusion," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, 2015. [Cited on page 4.]

[22] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "BundleFusion: Real-Time Globally Consistent 3D Reconstruction Using On-the-Fly Surface Reintegration," *ACM Transactions on Graphics*, vol. 36, no. 3, May 2017. [Cited on page 4.]

[23] V. Dietrich, D. Chen, K. M. Wurm, G. v. Wichert, and P. Ennen, "Probabilistic multi-sensor fusion based on signed distance functions," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2016. [Cited on page 4.]

[24] P. Biber and W. Strasser, "The normal distributions transform: a new approach to laser scan matching," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2003, pp. 2743–2748. [Cited on page 5.]

[25] E. Takeuchi and T. Tsubouchi, "A 3-D scan matching using improved 3-D normal distributions transform for mobile robotic mapping," in *IEEE International Conference on Intelligent Robots and Systems*. IEEE, Oct 2006, pp. 3068–3073. [Cited on page 5.]

[26] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan Registration for Autonomous Mining Vehicles Using 3D-NDT," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, 2007. [Cited on page 5.]

[27] T. Stoyanov, M. Magnusson, and A. J. Lilienthal, "Comparative evaluation of the consistency of three-dimensional spatial representations used in autonomous robot navigation," *Journal of Field Robotics*, vol. 30, no. 2, pp. 216–236, Mar 2013. [Cited on pages 5, 54, and 54.]

[28] J. P. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "3D normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1627–1644, Sep 2013. [Cited on page 5.]

[29] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. MIT Press, 2006. [Cited on page 5.]

[30] T. Lang, C. Plagemann, and W. Burgard, "Adaptive Non-Stationary Kernel Regression for Terrain Modeling," in *Robotics: Science and Systems*. Robotics: Science and Systems Foundation, Jun 2007. [Cited on page 5.]

[31] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard, "A Bayesian regression approach to terrain mapping and an application to legged robot locomotion," *Journal of Field Robotics*, vol. 26, no. 10, pp. 789–811, Oct 2009. [Cited on page 5.]

[32] S. Vasudevan, "Data fusion with Gaussian processes," *Robotics and Autonomous Systems*, vol. 60, no. 12, pp. 1528–1544, 2012. [Cited on page 5.]

[33] R. Hadsell, J. a. Bagnell, D. Huber, and M. Hebert, "Space-carving Kernels for Accurate Rough Terrain Estimation," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 981–996, 2010. [Cited on page 5.]

[34] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 42–62, 2012. [Cited on page 5.]

[35] M. G. Jadidi, J. V. Miro, and G. Dissanayake, "Warped gaussian processes occupancy mapping with uncertain inputs," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 680–687, Apr 2017. [Cited on pages 6 and 58.]

[36] V. Tresp, "A Bayesian committee machine," *Neural Computation*, vol. 12, no. 11, pp. 2719–2741, 2000. [Cited on page 6.]

[37] S. Kim and J. Kim, "Recursive Bayesian updates for occupancy mapping and surface reconstruction," in *Australasian Conference on Robotics and Automation*, 2014. [Cited on page 6.]

[38] J. Wang and B. Englot, "Fast, accurate Gaussian process occupancy maps via test-data octrees and nested Bayesian fusion," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 1003–1010. [Cited on page 6.]

[39] F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1717–1730, Dec 2016. [Cited on page 6.]

[40] K. Doherty, J. Wang, and B. Englot, "Probabilistic map fusion for fast, incremental occupancy mapping with 3D Hilbert maps," in *IEEE International Conference on Robotics and Automation*, May 2016, pp. 1011–1018. [Cited on page 6.]

[41] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, Feb 2007. [Cited on page 7.]

[42] T. Jones, "Tractable Conflict Risk Accumulation in Quadratic Space for Autonomous Vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 1, pp. 39–48, Jan 2006. [Cited on page 7.]

[43] C. E. van Daalen and T. Jones, "Fast conflict detection using probability flow," *Automatica*, vol. 45, no. 8, pp. 1903–1909, Aug 2009. [Cited on page 7.]

[44] J. Guivant, E. Nebot, J. Nieto, and F. Masson, "Navigation and Mapping in Large Unstructured Environments," *The International Journal of Robotics Research*, vol. 23, no. 4, pp. 449–472, 2004. [Cited on pages 8, 20, 21, 22, and 63.]

[45] J. Nieto, J. Guivant, and E. Nebot, "DenseSLAM: Simultaneous Localization and Dense Mapping," *The International Journal of Robotics Research*, vol. 25, no. 8, pp. 711–744, Aug 2006. [Cited on pages 8, 20, 22, 23, 24, and 63.]

[46] C. D. Lombard and C. E. van Daalen, "Stochastic triangular mesh mapping: A terrain mapping technique for autonomous mobile robots," 2019. [Online]. Available: https://arxiv.org/abs/1910.03644 [Cited on page 9.]

[47] ——, "Stochastic triangular mesh mapping: A terrain mapping technique for autonomous mobile robots," *Robotics and Autonomous Systems*, p. 103449, Feb 2020. [Cited on page 9.]

[48] ——, "Extrinsic calibration of a push-broom LiDAR and camera using 3-D multi-planar association," in *Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*. IEEE, Nov 2016. [Cited on pages 9 and 65.]

[49] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT Press, 2009. [Cited on pages 10, 11, 12, 12, 13, 13, 13, 16, and 17.]

[50] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005. [Cited on pages 10, 15, 17, 34, 56, 56, 58, and 62.]

[51] A. T. Palacios and A. S. Lopez, "A Topological SPLAM Approach for Robust Exploration," in *Mexican International Conference on Artificial Intelligence*. IEEE, Nov 2014. [Cited on page 10.]

[52] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006. [Cited on pages 15, 15, 15, and 32.]

[53] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *IEEE International Conference on Communications*. IEEE, 1993. [Cited on page 17.]

[54] R. J. McEliece, D. J. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's "Belief Propagation" algorithm," *IEEE Journal on selected areas in communications*, vol. 16, no. 2, pp. 140–152, 1998. [Cited on page 17.]

[55] A. Ranganathan, M. Kaess, and F. Dellaert, "Loopy SAM," in *International Joint Conference on Artifical Intelligence*. Morgan Kaufmann Publishers Inc., 2007, pp. 2191–2196. [Cited on page 17.]

[56] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational Inference: A Review for Statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017. [Cited on page 18.]

[57] T. Minka, "Divergence measures and message passing," Technical report, Microsoft Research, Tech. Rep., 2005. [Cited on pages 18, 19, 19, and 32.]

[58] J. Winn and C. M. Bishop, "Variational Message Passing," *Journal of Machine Learning Research*, vol. 6, pp. 661–694, 2005. [Cited on pages 19, 31, 33, and 35.]

[59] T. P. Minka, "Expectation propagation for approximate Bayesian inference," in *Uncertainty in Artificial Intelligence*, 2001, pp. 362–369. [Cited on page 19.]

[60] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision*. Springer, 2014, pp. 834–849. [Cited on page 20.]

[61] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017. [Cited on page 20.]

[62] S. Friedman, H. Pasula, and D. Fox, "Voronoi Random Fields: Extracting the Topological Structure of Indoor Environments via Place Labeling," in *International Joint Conference on Artifical Intelligence*. Morgan Kaufmann Publishers Inc., 2007, pp. 2109–2114. [Cited on page 20.]

[63] K. Konolige, E. Marder-Eppstein, and B. Marthi, "Navigation in hybrid metric-topological maps," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2011, pp. 3041–3047. [Cited on page 20.]

[64] P. Schmuck, S. A. Scherer, and A. Zell, "Hybrid Metric-Topological 3D Occupancy Grid Maps for Large-scale Mapping," *IFAC-PapersOnLine*, vol. 49, no. 15, pp. 230–235, 2016. [Cited on page 20.]

[65] S. Lowry, N. Sunderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual Place Recognition: A Survey," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 1–19, 2016. [Cited on page 21.]

[66] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI*. SPIE, Jul 1997. [Cited on page 23.]

[67] S. Julier, "The scaled unscented transformation," in *American Control Conference*. IEEE, 2002. [Cited on pages 23 and 56.]

[68] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, Jun 2006. [Cited on page 23.]

[69] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun 2001. [Cited on page 23.]

[70] L. K. Saul and M. I. Jordan, "Exploiting Tractable Substructures in Intractable Networks," in *Advances in Neural Information Processing Systems*. MIT Press, 1996, pp. 486–492. [Cited on page 31.]

[71] B. J. Frey and D. J. C. MacKay, "A revolution: Belief propagation in graphs with cycles," in *Advances in Neural Information Processing Systems*. MIT Press, 1998, pp. 479–485. [Cited on page 38.]

[72] S. Streicher and J. du Preez, "Graph coloring: Comparing cluster graphs to factor graphs," *ACM Multimedia 2017 Workshop on South African Academic Participation*, 2017. [Cited on page 38.]

[73] P. Rusmevichientong and B. Van Roy, "An analysis of belief propagation on the turbo decoding graph with Gaussian densities," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 745–765, 2001. [Cited on page 40.]

[74] Y. Weiss and W. T. Freeman, "Correctness of Belief Propagation in Gaussian Graphical Models of Arbitrary Topology," *Neural Computation*, vol. 13, no. 10, pp. 2173–2200, Oct 2001. [Cited on page 40.]

[75] Y. Qi and T. P. Minka, "Window-based expectation propagation for adaptive signal detection in flat-fading channels," *IEEE Transactions on Wireless Communications*, vol. 6, no. 1, pp. 348–355, 2007. [Cited on page 40.]

[76] M. Seeger and M. Jordan, "Sparse gaussian process classification with multiple classes," Department of Statistics, University of Berkeley, CA, Tech. Rep., 2004. [Cited on page 42.]

[77] O. Zoeter and T. Heskes, "Gaussian quadrature based expectation propagation," in *Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2005, pp. 445–452. [Cited on page 42.]

[78] K. Perlin, "An image synthesizer," *SIGGRAPH Computer Graphics*, vol. 19, no. 3, pp. 287–296, Jul. 1985. [Cited on page 44.]

[79] GPy, "GPy: A gaussian process framework in python," http://github.com/SheffieldML/GPy, since 2012. [Cited on page 48.]

[80] C. H. Tong, D. Gingras, K. Larose, T. D. Barfoot, and Érick Dupuis, "The Canadian planetary emulation terrain 3D mapping dataset," *The International Journal of Robotics Research*, vol. 32, no. 4, pp. 389–395, 2013. [Cited on pages 50, 52, 54, and 55.]

[81] Q.-S. Xu and Y.-Z. Liang, "Monte Carlo cross validation," *Chemometrics and Intelligent Laboratory Systems*, vol. 56, no. 1, pp. 1–11, Apr 2001. [Cited on page 54.]

[82] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014. [Cited on pages 56 and 59.]

[83] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Asian Conference on Computer Vision*. Springer, 2011, pp. 25–38. [Cited on page 56.]

[84] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the EKF-SLAM algorithm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct 2006. [Cited on page 58.]

[85] S. Huang and G. Dissanayake, "Convergence and consistency analysis for extended kalman filter based SLAM," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 1036–1049, Oct 2007. [Cited on page 58.]

[86] G. Huang, A. Mourikis, and S. Roumeliotis, "On the complexity and consistency of UKF-based SLAM," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2009. [Cited on page 58.]

[87] T. Bailey, J. Nieto, and E. Nebot, "Consistency of the FastSLAM algorithm," in *IEEE International Conference on Robotics and Automation*. IEEE, 2006, pp. 424–429. [Cited on page 58.]

[88] D. Brink, "Using probabilistic graphical models to detect dynamic objects for mobile robots," Ph.D. dissertation, Stellenbosch University, 2016. [Cited on page 62.]

[89] S. Wasielewski and O. Strauss, "Calibration of a multi-sensor system laser rangefinder/camera," in *Intelligent Vehicles Symposium*. IEEE, 1995. [Cited on pages 66, 66, and 66.]

[90] G. Li, Y. Liu, L. Dong, X. Cai, and D. Zhou, "An algorithm for extrinsic parameters calibration of a camera and a laser range finder using line features," in *IEEE International Conference on Intelligent Robots and Systems*, 2007, pp. 3854–3859. [Cited on pages 66, 66, and 66.]

[91] K. Kwak, D. F. Huber, H. Badino, and T. Kanade, "Extrinsic Calibration of a Single Line Scanning LIDAR and a Camera," in *IEEE International Conference on Intelligent Robots and Systems*, 2011, pp. 3283–3289. [Cited on pages 66, 66, 66, 67, 67, 67, 68, and 72.]

[92] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, pp. 2301–2306. [Cited on pages 66, 67, 67, and 68.]

[93] F. Vasconcelos, J. P. Barreto, and U. Nunes, "A Minimal Solution for the Extrinsic Calibration of a Camera and a Laser-Rangefinder," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2097–2107, Nov 2012. [Cited on page 66.]

[94] L. Zhou, "A New Minimal Solution for the Extrinsic Calibration of a 2D LiDAR and a Camera Using Three Plane-Line Correspondences," *IEEE Sensors Journal*, vol. 14, no. 2, pp. 442–454, Feb 2014. [Cited on page 66.]

[95] J. E. Ha, "Extrinsic calibration of a camera and laser range finder using a new calibration structure of a plane with a triangular hole," *International Journal of Control, Automation and Systems*, vol. 10, no. 6, pp. 1240–1244, 2012. [Cited on pages 66 and 67.]

[96] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic extrinsic calibration of vision and lidar by maximizing mutual information," *Journal of Field Robotics*, vol. 32, no. 5, pp. 696–722, 2015. [Cited on page 66.]

[97] A. Napier, P. Corke, and P. Newman, "Cross-calibration of push-broom 2D LIDARs and cameras in natural scenes," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 3679–3684. [Cited on page 66.]

[98] T. Scott, A. A. Morye, P. Piniés, L. M. Paz, I. Posner, and P. Newman, "Choosing a Time and Place for Calibration of Lidar-Camera Systems," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 4349–4356. [Cited on page 66.]

[99] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, 2003. [Cited on page 70.]

[100] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000. [Cited on page 70.]

[101] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An Accurate O(n) Solution to the PnP Problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, Jul 2008. [Cited on page 70.]

[102] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944. [Cited on pages 70, 71, and 72.]

[103] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, Jun 1963. [Cited on pages 70, 71, and 72.]

[104] P. J. Huber, "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, Mar 1964. [Cited on page 72.]

[105] D. Freedman and P. Diaconis, "On the histogram as a density estimator: L2 theory," *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, vol. 57, no. 4, pp. 453–476, 1981. [Cited on page 74.]