PERSISTENT MAPPING IN QUASI-STATIC DYNAMIC ENVIRONMENTS USING UNMANNED AERIAL VEHICLES

by

Amy Deeb

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
March 2021

Table of Contents

List of Tables

List of Figures

Abstract

Mapping and tracking Arctic sea ice is critical to protecting the people and animals that inhabit and transit the Canadian Arctic. Unmanned aerial vehicles (UAVs) are well suited to these activities given the vast area of interest, the remote, potentially dangerous environment, and the need to persistently capture the most current state of the map. The scarcity of reliable global position references at high latitudes means that a UAV must localize itself within a map it is in the process of building – a problem called Simultaneous Localization and Mapping (SLAM). While traditional SLAM assumes all landmarks are static, UAVs in marine Arctic environments must perform persistent mapping while coping with the motion of ice masses in the dynamic environment. This research proposes piecewise-deterministic quasi-static pose graph SLAM (PDQS-SLAM) to exploit landmark motion information during graph construction. This will contribute to persistent mapping in a dynamic environment.

The primary contribution of this thesis is the relaxation of the static assumption inherent to traditional SLAM algorithms. This is achieved by assigning a kinematic motion model to each landmark, proposing a new loop closure factor structure, and estimating the landmark motion alongside the UAV trajectory in the pose graph. Applicability to a general dynamic environment is achieved by augmenting loop closure edges with a state, governed by a finite state machine (FSM), that captures the edge's behaviour over time. This FSM captures edge behaviours during constant-velocity epochs, and across disrupting events detected using score-based structure learning.

The resulting PDQS-SLAM is validated in both simulations and laboratory experiments. The localization and mapping performance of PDQS-SLAM in a dynamic environment including two mobile landmarks that experience events, is shown to have similar performance when compared to the baseline static SLAM in a static environment. This demonstrates that the static assumption can be relaxed by modelling landmark motion and responding appropriately to inevitable disruptions to that motion. This is a step towards persistent mapping by UAVs in a complex, quasi-static environment inspired by the Canadian marine Arctic.

List of Abbreviations and Symbols Used

Acronyms

ACML	Advanced Control and Mechatronics Laboratory
DATMO	detection and tracking of moving objects
DOF	degree-of-freedom
EM	expectation maximization
FOV	field-of-view
FSM	finite state machine
GPS	global positioning system
IMU	inertial measurement unit
IR	infra-red
iSAM	incremental Smoothing and Mapping
ISL	Intelligent Systems Laboratory
KMM	kinematic motion model
LV	latent variable
MBD	model-based dynamic
MBDF	model-based dynamic factors
MBQS	model-based quasi-static
MBQSF	model-based quasi-static factors
PD	proportional-derivative controller

PDQS	piecewise-deterministic quasi-static
QSA	quasi-static assumption
RMSE	root mean square error
ROS	robot operating system
SBSL	score-based structure learning
SLAM	simultaneous localization and mapping
UAV	unmanned aerial vehicle
UGV	unmanned ground vehicle
USV	unmanned surface vehicle
UUV	unmanned underwater vehicle
WLS	weighted least squares

Factor Graph Construction and Optimization Symbols

Δt	elapsed time, time step
Λ_{ij}	covariance matrix for loop closure factor
Σ_i	covariance matrix for odometry fractor
E	error, concatenation of e and e_v, used in MBQSFs
e	error between the estimate (h_s) and the measurement ($z_{ij,k}$)
e_v	error between the estimate (h_v) and the measurement (z_{vk})
$e_{ij,k}$	error between the estimate (h_{ij}) and the measurement ($z_{ij,k}$)
$f_{a,b}$	factor between nodes a and b

h_{ij}	estimate of the measurement between poses i and j found using values from the factor graph
h_s	estimate of the measurement between poses i and j if the landmark had been static
h_v	estimate of the measurement of the landmark velocity
L_k	pose of landmark k
t_i	time when the UAV was at pose i
T_{ik}	transformation (actual) between UAV pose i and landmark k at time t_i
$T_{k^ik^j}$	transformation (actual) of landmark k between time t_i and t_j
$T_{s_{ij}}$	transformation (actual) of UAV between time t_i and t_j assuming a static landmark
u_i	control action that resulted in arrival at pose i
V^*	optimal landmark velocities
X^*	optimal UAV trajectory
x_i	UAV pose at instance i
$z_{ij,k}$	measurement of transform of UAV between poses i and j calculated from observations z_{ik} and z_{jk}, assumes static landmark
z_{ik}	observation of landmark k made from UAV pose i
z_{vk}	measurement of velocity of landmark k

UAV Model Symbols

$\mathcal{N}(\mu, \sigma^2)$	Normal distribution governed by a mean and standard deviation
μ	mean

ϕ, θ, ψ	roll, pitch and yaw
σ	standard deviation
k_d	UAV PD controller derivative gain
k_p	UAV PD controller proportional gain
o_{alt}	mean intermittent altitude error step-change
t_{oalt}	mean time between intermittent altitude error step-changes
X, Y, Z	axes of the Cartesian reference system
x, y, z	coordinates in the Cartesian reference system

Mission and Environment Symbols

υ	characteristic velocity of a landmark
f	SLAM pose update frequency
h	UAV altitude
l	UAV pass length
l_{FOV}	length of the sensor FOV in the direction of UAV motion, on the plane of the landmarks
t_{obs}	time between observations of the common landmark
v_{ice}	iceberg speed
v_k	velocity of landmark k
v_{QS}	quasi-static speed threshold
v_{UAV}	UAV speed
x_{min}	minimum detectable change in position

Landmark Motion Model Symbols

$\Delta\theta$	change in landmark orientation
Δp	change in landmark position
ω	angular velocity of the body, world frame
ω_b	angular velocity of landmark, body frame
ω_{meas}	landmark angular velocity measurement used in MBQSFs
σ_ω	standard deviation on the landmark angular velocity measurement used in MBQSFs
σ_{QSA}	standard deviation on the assumed zero-velocity for landmarks under the QSA
σ_v	standard deviation on the landmark translational velocity measurement used in MBQSFs
σ_{WLS}	standard deviation on the WLS velocity estimate (also, $Var(\hat{m})$)
a_b	translational acceleration of landmark, body frame
R_b	initial orientation of the body, world frame
v	translational velocity of the body, world frame
v_b	translational velocity of landmark, body frame
v_{meas}	landmark translational velocity measurement used in MBQSFs

Weight Least Squares Regression Symbols

$\bar{x}_w$	weighted mean of the x coordinate for a linear system
$\bar{y}_w$	weighted mean of the y coordinate for a linear system
$\hat{b}$	estimated y-intercept for a linear system

$\hat{m}$	estimated slope for a linear system
$Var(\hat{m})$	variance of the estimated slope for a linear system
w_i	weight of observation i used in weighted least squares regression for a linear system

Event Detection Symbols

$\Delta\hat{v}$, $\Delta\hat{\omega}$	change in translational and angular velocity estimates that results from incorporating a new observation (z_k) in the WLS regression
θ^0, θ^1	mobile landmark heading before and after event
ε_v, ε_ω	event detection pre-filter thresholds on translational and angular velocity
v^0, v^1	mobile landmark speed before and after event

Augmented Edges Symbols

Σ_k	covariance on p_k
Σ_T	covariance on the geometrical transform that defines an edge
E_k^n	loop closure edge n for landmark k in current epoch
n	number of edges in E_k^e
p_k	parameters of landmark k's KMM
s_k	state of the set of edges an edge belongs to, governed by FSM
T	geometrical transform that defines an edge
E_k^e	set of loop closure edges in a graph related to landmark k, in epoch e

Acknowledgements

I would like to thank Dr. Mae Seto and Dr. Ya-Jun Pan for their guidance and generous support throughout this thesis program. Their examples as world-class researchers, professional engineers, and women have benefited me in innumerable ways.

They also welcomed me into their laboratory groups: the Intelligent Systems Lab and Advanced Control and Mechatronics Lab, where I found a community of engineers and researchers. I am so grateful for my lab-mates' encouragement, advice and willingness to challenge my perspectives. I feel so lucky to work with such amazing human beings.

My gratitude goes out to my family and friends, and especially my partner, for their unending patience and kindness through the ups and downs of this program. Whether near or far, this thesis would not have been completed without their support.

This research project was supported by the National Sciences and Engineering Research Council (NSERC), the Nova Scotia Government (NSGS), and the Irving Shipbuilding Research Chair in Marine Engineering and Autonomous Systems.

Chapter 1 Introduction

Understanding the Arctic environment is important to Canada's identity both domestically and internationally. As our climate continues to warm, mapping and tracking Arctic sea ice will be critical to protecting the people and animals that inhabit and transit the region. Autonomous systems are well suited to these activities given the vast area of interest, the remote, potentially dangerous environment and the need to capture the most current state of the environment. In particular, unmanned aerial vehicles (UAVs) can be deployed from a ship to persistently map the area of interest in near real-time allowing safe passage in unmapped dynamic environments, such as the scenario shown in Figure 1.1.

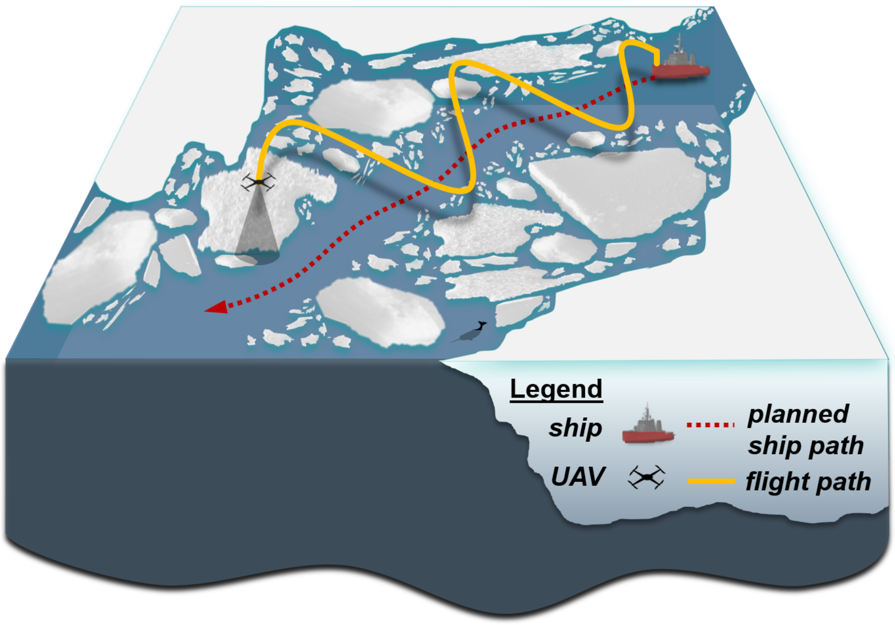


Figure 1.1: Motivating application for the thesis. A UAV flies ahead of a ship in the Arctic, mapping ice to create an up-to-date map and determine safe paths through the dynamic environment.

Deploying a UAV from a ship increases situational awareness by increasing the distance to the horizon. The distance to the horizon increases with altitude according to the relationship $d \approx 3.57\sqrt{h}$, where d_h is the distance to the horizon in km and h is the UAV altitude in m. While at 1 m above the surface, the horizon is only 3.5 km away, the horizon for a UAV at an altitude of 13 m would be more than 12 km away providing additional advantages to the ship.

UAVs function autonomously, using their own sensors and decision-making capabilities to perform the mission [1], while also detecting and recovering from failures including communication drop-outs, imperfect sensors, and inclement weather – all typical characteristics of the Arctic environment. Additionally, the scarcity of a reliable global position reference, such as from the global positioning system (GPS), at high latitudes means that a UAV must localize itself within the map it is in the process of building – *a problem called Simultaneous Localization and Mapping* (SLAM). While traditional SLAM algorithms assume all elements in the map are static, UAVs operating in marine Arctic environments must perform so-called *persistent autonomy* to cope with the complex dynamic environment and long-term missions [2].

In particular, *persistent mapping* is a key behaviour in which an autonomous vehicle develops a representation of a changing environment over the life of the system – including tracking targets that change their position and appearance over time. Persistent mapping not only enables navigation, it also provides data necessary for scientific insight to the changing Arctic environment. This thesis proposes an approach that addresses the challenges of persistent mapping to enable a UAV to operate autonomously in a dynamic environment.

1.1 Autonomous Systems

Autonomous systems are used in a wide variety of applications typically characterized as dull, dirty or dangerous [2]. These are applications that humans are not very good at because they either require consistently repeating a boring task or occur in an environment that puts their life at risk. On the other hand, robots excel at repetition and can be built more robustly than a human body as needed. Autonomous systems are also good choices for applications where frequent, detailed data is available to the robot to act upon more quickly than human reflexes may allow. However, these systems face challenges when responding to unexpected events in complex environments. This thesis develops the persistent mapping behaviour to address one such challenge that is common in the marine Arctic environment – dynamic landmarks.

It is convenient to describe the performance of autonomous systems through the behaviours they display. For instance, an autonomous car might perform a highway driving behaviour, a parking behaviour or an emergency stop behaviour. These are high level behaviours that can be built up from more detailed states and actions. It is also useful to capture the behaviour of a system as a series of transitions between states. As a simple example, the autonomous car might start in an highway-driving state, and transition to an emergency-stop state if an obstacle is detected. The field of finite automata has proposed using finite state machines (FSM) to succinctly define these states and transitions. This thesis will use this language of behaviours and FSM to enable SLAM graphs to adapt to dynamic environments.

Robotic systems can be broadly classified based on their capability to operate without direct human intervention [1, 2]. At the low end, non-autonomous systems are *human operated*. Here humans make every decision and adjust all parameters, control settings and tasks manually. The robot can be considered a remote sensor that follows direct instructions, sent in real-time. This is the typical remote control operation paradigm. If a robot is able to follow scripted instructions, using a timer or a look up table based on specified sensor measurements, it displays a low-level of automation termed *human delegated*. These systems are only as good as their instructions, meaning detailed programmed responses are necessary for all scenarios. In the case of a flying vehicle this could mean the pilot requests a displacement of 1 m in the forward direction and the vehicle calculates the correct power levels to control that motion, then waits for the next instruction from the pilot.

The next level is the first stage of proper autonomy, called *human supervised*. At this stage robots are asked to perform behaviours, and are enabled to take a broad set of actions in response to sensor data. The human sets a series of tasks or behaviours and monitors all operations. If there are unexpected circumstances, the operator can take over some or all of the operations to recover. In order to reach *full autonomy* the system must be able to recover from these anomalies and on-board faults in a safe manner. Typically, fully autonomous systems are also able to translate general goals programmed by an operator into the specific tasks and behaviours it must perform. For example, a fully autonomous aerial vehicle could be given the goal of surveying a field bounded by given coordinates to measure crop yield. It would then need to determine the imaging settings, plan a path and adjust this plan based on the in-situ measurements. This type of autonomy is highly desirable for robotic systems engaged in tasks that have limited communication with operators and occur in harsh environments that can change quickly, such as the marine Arctic.

At a high-level, robotic systems cycle from perception to planning, to acting and back again as shown in Figure 1.2. Depending on their level of autonomy the planning stage may be completed by the human operator.

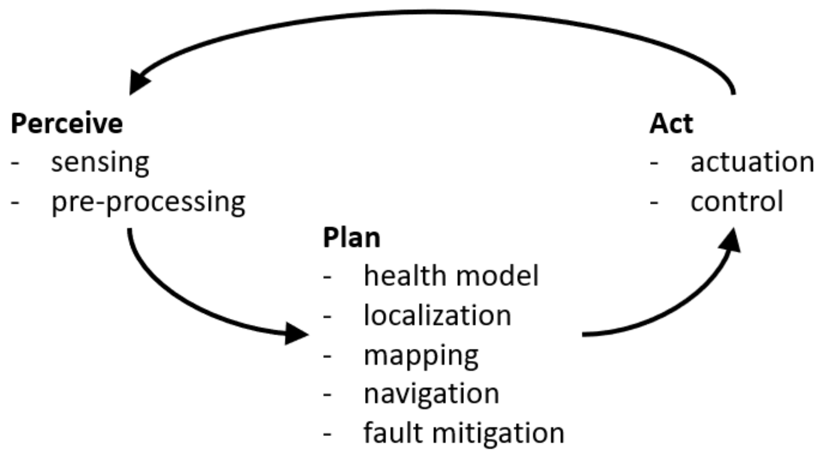


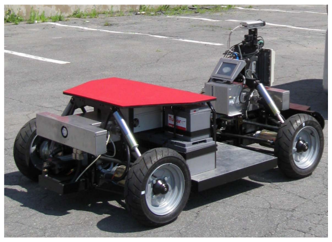
Figure 1.2: Architecture of an autonomous system showing the cycle of information flow.

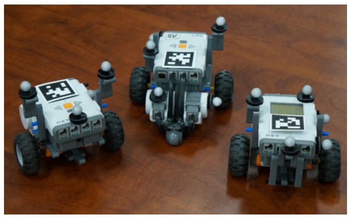
Perception is critical to an autonomous system's behaviour as sensors are how a robot gathers data about the current state of its environment and itself. As such, sensors are very application specific and pre-processing algorithms are highly tailored to the environment and robotic system on which they are integrated [1, 2]. In the context of UAVs, sensors can be grouped into two categories:

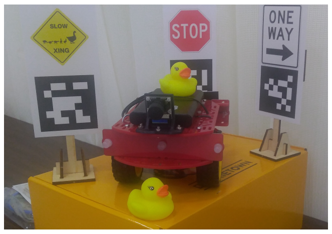
- proprioceptive: measures internal parameters/states of the UAV (e.g. inertial measurement unit (IMU)), and
- exteroceptive: measures external parameters/states of the UAV's environment (e.g. camera observations of landmarks).

By fusing together information from these sensors, the UAV is able to, depending on its level of autonomy, adapt its mission to navigate a changing environment [2]. This process of fusing information from multiple sensors will be leveraged in the dynamic SLAM developed in this thesis.

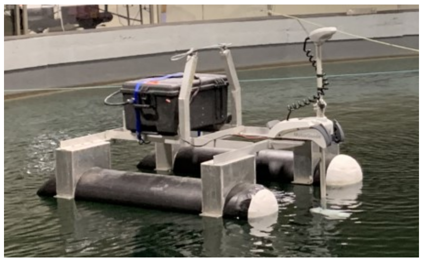
Autonomous systems come in many forms, but are typically classified by their medium of locomotion. Unmanned ground vehicles (UGV) use wheels or tracks to drive across the ground or floor, unmanned surface vehicles (USV) use motors or sails to propel them across the water, UAVs typically use propellers to fly through the air and unmanned underwater vehicles (UUV) typically use propellers to swim through the water [1]. Unmanned spacecraft can also be considered autonomous systems using momentum, magnetic fields or propellant to change their orbits. Members of the Advanced Control and Mechatronics Laboratory (ACML) and the Intelligent Systems Laboratory (ISL) at Dalhousie University, have access to many different autonomous systems. Several examples are shown in Figure 1.3.


(a) RobuCar UGV.

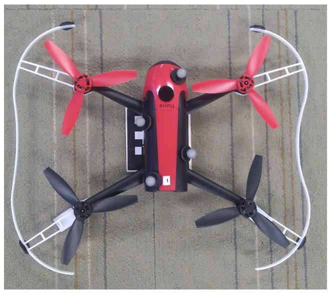

(b) LEGO NXT UGVs.



(c) DuckieBot UGV.


(d) ISL USV.

(e) Riptide UUV.


(f) Bebop UAV.

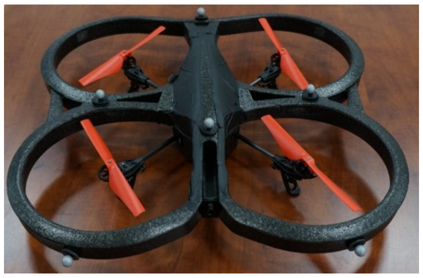

(g) AR.Drone UAV.

Figure 1.3: Examples of autonomous systems available in the ACML and ISL. Image permissions: (a) Y.J. Pan, (d) J. Lindsay and (e) E. Wetter.

In the Arctic environment, we are particularly interested in small UAVs that can provide increased line of sight for ships navigating ice-fields. UAVs encompass two primary types of aircraft: fixed-wing (i.e. airplanes) and rotary-wing (i.e. helicopters, quadrotors, hexacopters). This thesis will focus on quadrotors as they provide a stable platform that can hover in place and move laterally to improve a UAV's capability as a remote sensor.

1.2 Persistent Mapping

As autonomous systems become more capable they are asked to perform long-term autonomous behaviours in increasingly complex environments, also called *persistence*. This requires autonomous systems to not only react, but to reason about their environment and develop an understanding of it [2]. Critical to this process is a robot's ability to localize itself and build a map. For robots operating in unknown environments without a global position reference, localization and mapping are a simultaneous problem, in which the robot cannot find its position without a map and cannot add sensor measurements (observations) to a map without knowing its position. This is the SLAM problem.

The majority of solutions to the SLAM problem have only addressed operations in environments that are assumed to be static [1, 3]. As robots are tasked with remaining on station for longer durations, this assumption will be violated. In addition, marine environments are particularly challenging for SLAM applications as they are unstructured and populated with landmarks that change their appearance over time [2]. For instance, in Arctic environments, icebergs are available as landmarks, yet they are amorphous, which makes them difficult to characterize, they move and, as they melt, they often flip over – changing their appearance. This thesis is interested in the persistent mapping problem and its intersection with SLAM in dynamic environments. A particularly motivating example is the Arctic environment, however there are also parallel challenges in other scenarios such as agricultural mapping where crops grow and change appearance over time, or underwater where terrain and sunken objects may be evolving or in motion.

1.3 Factor Graph SLAM

This thesis develops new approaches based on recent advances in using graph optimization techniques to solve the SLAM problem. Early SLAM work utilized different forms of Kalman filters and later particle filters to simultaneously estimate the trajectory of the robot and the map of landmarks in the environment [1, 3, 4]. These solutions used the fact that the error between the true and estimated landmark positions is common, as it stems from uncertainty in the pose of the landmark at the time it is observed. This means that while the position of a landmark from a single observation may be noisy, the relative position between two landmarks can be known with high accuracy [5]. Futher, this accuracy increases monotonocally as more observations are made [5]. Smoothing algorithms, such as Graph SLAM, were a later approach to solving the SLAM problem and have been shown to be efficient in their use of non-linear graph optimization methods [6].

Factor graphs are a particular type of probabilistic model that express a function in terms of its composing factors [7, 8]. The term *graph* refers to a structure consisting of a set of nodes that represent random variables, and a set of edges that represent dependencies between the connected nodes. In the case of SLAM, the nodes can represent positions of the vehicle over time and edges can represent geometrical transformations (constraints) between those positions. An example of a factor graph is shown in Figure 1.4 with x_i, x_{i+1}, x_j representing the vehicle poses over time, L_k representing a landmark and $f_{a,b}$ representing factors (geometrical constraints) between nodes a and b. Optimization on the graph implies determining the best values for the variables (nodes) given the constraints (edges) [8]. Graphs are also helpful for visualizing a probabilistic problem, such as SLAM, to gain insight to efficient approaches for estimating the relevant variables given the constraints.

The accuracy of measurements used to construct the graph impacts the accuracy of the estimate of the trajectory and map [8]. When dynamic landmarks are assumed to be static, the geometric constraints cannot account for the motion of the landmark and subsequently drive the trajectory estimate away from the true values. For this reason, when applied to dynamic environments the traditional SLAM algorithms seek to identify and remove any dynamic landmarks, thereby ensuring the map (and all factors) only contain static landmarks. Although in some cases it is possible

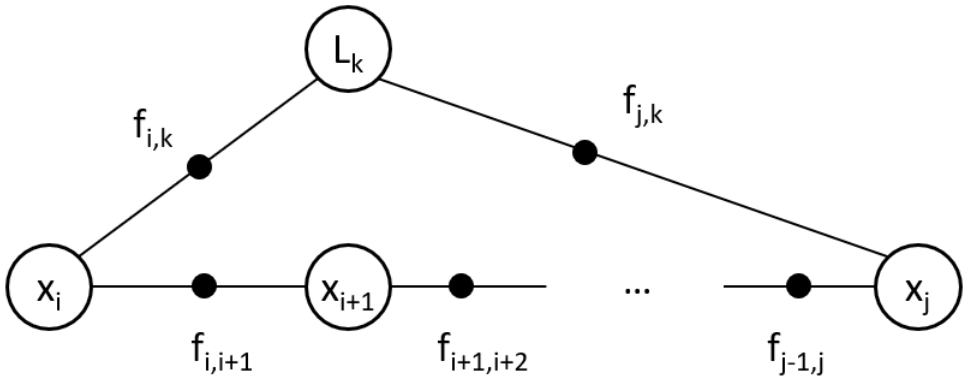


Figure 1.4: Example of a factor graph for SLAM, where circles are nodes (x are vehicle poses, L_k is a landmark) and lines with a dot on them are factors that geometrically constrain the nodes.

to consider small motions to be a category of noise contribution, this increases the uncertainty of the estimate. Instead, this thesis hypothesizes that estimating and accounting for the dynamics of the landmark separately from the measurement noise can improve performance.

1.4 Research Motivation and Objective

Persistent mapping is an enabler for autonomous operations in complex dynamic environments, such as the Canadian marine Arctic, that motivates this research. Recent strides towards incremental graph optimization techniques, such as [9], have been significant and inspired this thesis work. While the specific case of interest shown in Figure 1.1 requires a UAV to persistently develop a map of a quasi-static dynamic marine Arctic environment, this work aims to develop a SLAM approach that can be used in a variety of complex, dynamic environments by any class of autonomous vehicle.

Objective: The objective of this research is to relax the static assumption of existing SLAM solutions to allow autonomous operations in quasi-static dynamic environments populated with evolving, amorphous landmarks.

1.5 Contributions

The primary contribution of this thesis is the relaxation of the static assumption. In support of that work, two further contributions were made: assigning kinematic motion models (KMM) to landmarks, and designing an FSM that governs the behaviour of edges (factors) associated with non-stationary landmarks. A fourth contribution is the end-to-end validation of the proposed piecewise-deterministic quasi-static pose graph SLAM (PDQS-SLAM) algorithm in simulations and experiments. Details of these contributions are described below.

1. **Relaxation of Static Assumption**
   A basic assumption of the SLAM problem, and subsequently SLAM solutions, is that all landmarks are static. This thesis shows that this assumption can be replaced with an assumption that landmarks follow a motion model whose structure is known a priori and whose values can be learned through observation during the mission. While in principle this is no different than adding any other variables into a factor graph, these particular variables allow the factors to represent not only geometrical constraints, but also time-dependent constraints. This in turn allows dynamic elements to be exploited, rather than excluded, as they are in traditional SLAM approaches. In complex environments, this is especially important, as there are comparatively few static landmarks. The cases demonstrated in this thesis focus on constant velocity motion that is common in natural and artificial environments, however, this could be more generally applied in environments that behave according to other kinematic or appearance-based evolutionary models. The implementation of this contribution is presented in Chapters 5 and 6.

2. **Landmark Kinematic Motion Models**
   The relaxation of the static assumption is supported by the design of piecewise-deterministic landmark KMMs and the definition of quasi-static environments as a distinct regime of landmark motion. While most work divides dynamic environments into the static elements that are useful for mapping and the dynamic elements that are filtered and removed or tracked separately, in this work, dynamic environments are conceptualized on a continuum defined not only by

the speed of change, but also by the sensor's observation frequency and precision. Inherent to this is a novel attribution of a KMM or appearance-based evolutionary model to landmarks used in SLAM.

In particular, this work addresses piecewise-deterministic quasi-static dynamic landmarks. The concept of quasi-static motion describes landmarks that evolve so slowly that their motion cannot be directly measured in consecutive observations. Meanwhile, the concept of piecewise-deterministic models describes motion of landmarks that follow a deterministic (constant parameter) model during epochs that are disrupted by discontinuities that are called events [10]. Together, *piecewise-deterministic quasi-static* defines a category of dynamic environments that is also useful for other areas of research into operations in complex environments, whether related to SLAM or detection and tracking more generally. The definitions of quasi-static and piecewise-deterministic motion are presented in Chapter 3.

3. **Finite State Machine Edge Attributes**
   This work contributes a refinement of the definition of edges in a factor graph to include the state of an edge as it evolves through epochs and events. This extends the concept of assigning attributes to graph nodes, but goes further by constructing an FSM that allows the behaviour of each edge to change over time as the landmark through which it is constructed evolves. This FSM can be adapted to a variety of environments and cases where SLAM is implemented. For this application, the state transitions are focused on detecting events based on a measure of the edge's correctness calculated using the observations and proposed graph structure. This event detection applies the learnings from the field of score-based structure learning (SBSL) to the SLAM problem. The work on event detection is presented in Chapter 7 and the FSM contribution is detailed in Chapter 8.

4. **Validation of Piecewise-Deterministic Quasi-Static Dynamic SLAM**
   Finally, the proposed PDQS-SLAM algorithm is extensively demonstrated on a quadrotor UAV in both simulations and experiments. Several cases are analysed to show the capabilities and limitations of this approach in the motivating

scenario shown in Figure 1.1. This end-to-end SLAM approach is demonstrated in Chapter 8.

1.6 Organization

The main chapters of this thesis each apply to a different regime of landmark motion as exemplified in Figure 1.5. This thesis starts by implementing the baseline static SLAM algorithm (Chapter 4), then proposes solutions for epochs where the landmark motion is slow (Chapter 5) or quasi-static (Chapter 6). Chapter 7 develops a novel event detection approach that enables the proposed piecewise-deterministic quasi-static pose graph SLAM (Chapter 8) to apply within and across events that disturb static, slow and quasi-static epochs.

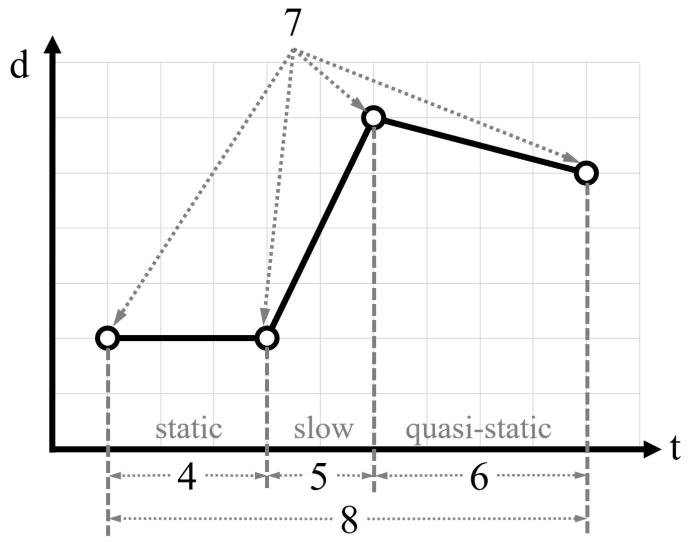


Figure 1.5: Diagram of organization of thesis chapters (4-8) by applicable regimes of motion in a piecewise-deterministic dynamic environment, consisting of four events disturbing three epochs: one static, one slow and one quasi-static.

The organization of this thesis is shown in Figure 1.6. After this introduction, a review of relevant literature is presented in Chapter 2. The problem formulation is detailed in Chapter 3 and provides context and definitions required for the main chapters of this thesis. Chapter 4 describes the experimental set-up used throughout the thesis including the mission, the experimental environments and the platforms. It also implements a baseline static SLAM, first in a static environment, and then in a slowly evolving environment to justify the need for a SLAM approach that accounts for landmark motion. Chapter 5 describes the development of novel model-based dynamic factors that form the basis for this work. These factors are subsequently applied to the limiting case of quasi-static environments in Chapter 6. In parallel, Chapter 7 shows how disruptions to the landmark motion model can be detected using SBSL. Next, the approaches developed in Chapters 6 and 7 together lead to the general dynamic SLAM approach for piecewise-deterministic quasi-static environments, proposed in Chapter 8. Finally, conclusions and future work are discussed in Chapter 9.

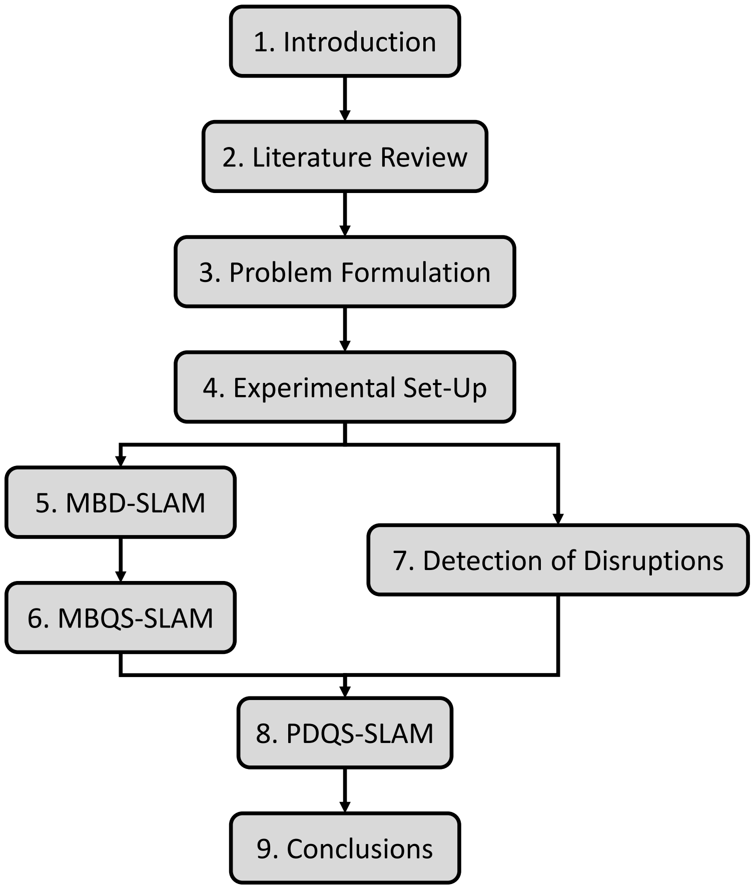


Figure 1.6: Diagram of flow of information through the thesis.

Chapter 2 Literature Review

This chapter describes the state-of-the-art for fields related to SLAM in dynamic environments. Literature is reviewed in the following four themes:

1. probabilistic Static SLAM (Section 2.1): describes the development of SLAM with a particular focus on the promising field of graph SLAM and the challenge of data association.

2. detection and tracking of moving objects (DATMO) (Section 2.2): provides a summary of methods in the field of DATMO that are useful towards the estimation of iceberg motion.

3. anomaly detection (Section 2.3): outlines the tools available to address the detection of events that disrupt quasi-static landmark motion in the piecewise-deterministic environment.

4. dynamic SLAM (Section 2.4): describes current approaches to SLAM in a dynamic environment including: methods that filter out moving objects, methods that treat the static and dynamic parts of environments separately and semi-static methods that assume dynamic objects are static while the UAV is mapping, then move abruptly before the next mapping session.

Each section concludes with a comment on the relevance of the approaches reviewed to the thesis work.

2.1 Probabilistic Static SLAM

In his review of the principles of SLAM with Tim Bailey, Hugh Durrant-Whyte, defined the problem in the following way: "The simultaneous localization and mapping (SLAM) problem asks if it is possible for a mobile robot to be placed at an unknown location in an unknown environment and for the robot to incrementally build a consistent map of this environment while simultaneously determining its location within this map" [5]. Beginning with Smith et al.'s foundational paper in 1990 [11], early works by Cox [12] and Leonard and Durrant-Whyte [13] set out the primary challenges and opportunities that define the SLAM problem to this day. Topological mapping is also a clear precursor to the field of SLAM including the works of Dudek et al. [14, 15] and Kuipers and Beeson [16]. Since those early works, a myriad of approaches have been demonstrated that successfully solve the SLAM problem, there is still a long way to go before truly autonomous robotic operations is possible due to implementation challenges in real (large, unstructured, dynamic etc.) environments. This section outlines the SLAM problem statement based on [1, 5, 17, 3, 18].

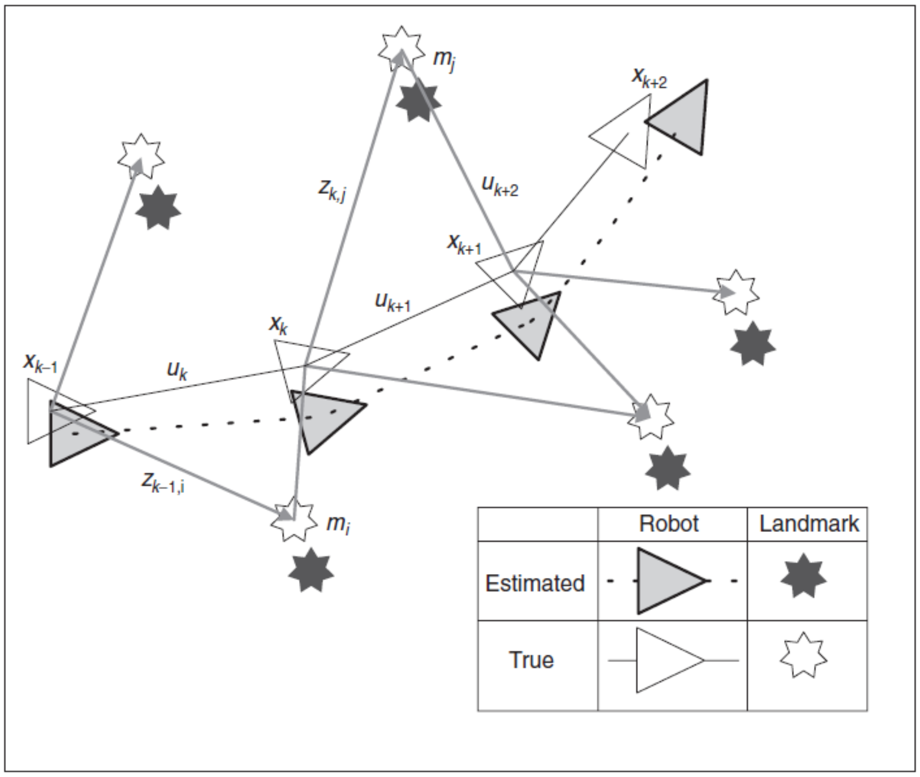


Figure 2.1: Essential description of the SLAM problem as defined by Durrant-Whyte and Bailey [5] © 2006 IEEE.

As shown in Figure 2.1, for each time instant k, several quantities are defined:

- $\mathbf{x}_k$: robot pose state (vector, location and orientation of the vehicle)
- $\mathbf{u}_k$: control action (vector) that led to the vehicle arriving at $\mathbf{x}_k$ (ie. applied at $\mathbf{x}_{k-1}$)
- $\mathbf{z}_{ik}$: observation of the ith landmark from $\mathbf{x}_k$

- $\mathbf{m}_i$: location of the ith landmark (vector which is assumed constant)

Further, the history of the poses, control inputs and landmark observations are grouped in the sets $\mathbf{X}_{0:k}$, $\mathbf{U}_{0:k}$ and $\mathbf{Z}_{0:k}$ respectively. Finally, the map that includes all landmarks is $\mathbf{m}$.

With these definitions, probabilistic SLAM has the goal of finding the posterior of the robot's trajectory and the environment's map, given the control action history, the landmark observation history and the robot's initial pose ($\mathbf{x}_0$). This is given as:

$$P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_{0:k}). \tag{2.1}$$

This probability distribution shows that the SLAM problem is a coupled one – requiring both the robot poses and landmark positions (or map) be calculated simultaneously. This posterior can in general be calculated in two ways. *Full SLAM*, requires the determination of the entire trajectory and map given the full history, while *Online SLAM*, requires the determination of only the current pose and map given the history up to the current time. In either case, to calculate the posterior, two steps are implemented recursively – a time-update (prediction) and a measurement-update (correction), given by Equations (2.2) and (2.3) [5], respectively.

$$P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \int P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \times P(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k-1}, \mathbf{x}_0)\mathrm{d}\mathbf{x}_{k-1} \tag{2.2}$$

$$P(\mathbf{x}_k, m|\mathbf{Z}_{0:k}, \mathbf{U}_{0:k}, \mathbf{x}_0) = \frac{P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k}, \mathbf{x}_0)}{P(\mathbf{z}_k|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \tag{2.3}$$

These updates require two models. The motion (or state transition) model given by, $P(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_k)$, is assumed to be a Markov process. The observation model ($P(\mathbf{z}_k|\mathbf{x}_k,\mathbf{m})$), represents the probability of an observation given a known state (position and map) – this is the state estimation with a known map problem. For SLAM, the time update makes a prediction about where the robot should be using the motion model, then the measurement update provides a correction by using the map. Although it is possible to separate the localization and mapping problems into two separate probability distributions, this makes the assumption that either the map is known (localization) or that the vehicle's trajectory is known (mapping), neither of which are generally valid assumptions.

One important thing to note about the structure of the SLAM problem is that the error between the true and estimated landmark positions is, for the most part, common between landmarks since it stems (related by a transformation) from the uncertainty in robot pose at the time of observation. This causes high correlation in estimates, which shows that although the position of a single landmark may not be well known, the distance between two landmarks can be known with high accuracy [5]. Further, this correlation increases monotonically as more observations are made [5]. The observed landmarks form a network of relative locations that converges as more measurements are made of any landmark in the network. Ideally, the map accuracy is only limited by the localization accuracy of the robot (rather than the sensor accuracy, which is stationary).

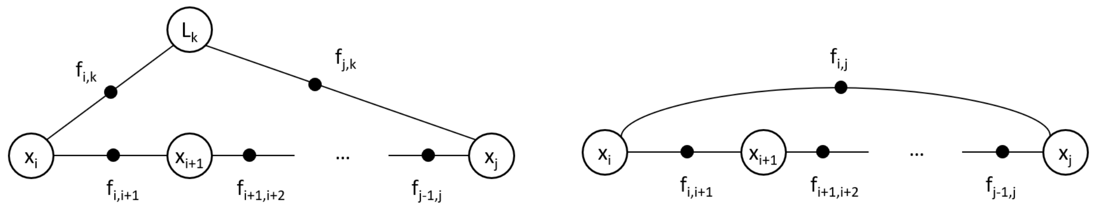
To summarize, solutions to the SLAM problem require applying the appropriate observation and motion models and computing a consistent posterior distribution for the map and robot trajectory [5]. Reviews of approaches and taxonomy of the SLAM problem are available in *The Handbook of Robotics* [18], *Probabilistic Robotics* [1], Durrant-Whyte and Bailey's SLAM tutorial (Part I and II) [5, 17] and a 2016 SLAM review from Saeedi et al. [3].

2.1.1 Graph SLAM

New tools and techniques have been recently emerging in the field of graph optimization which have allowed more flexibility and capability for the application of graph SLAM in unstructured and dynamic environments. The ability to incrementally solve the full SLAM problem is advantageous in the complex environments we are interested in here. This section will detail the mathematical basis of graph SLAM, drawing on the 2010 tutorial from Grisetti [19], as well as the graduate works of Walcott-Bryant [20] and Kretzschmar [21].

Graph SLAM was originally proposed in 1997 by Lu and Milios [22]. A graph is composed of a set of soft constraints that arise from observations that are encoded as edges between nodes that represent poses (spatial coordinates) of the robot at each step in time (x_t) and locations of landmarks in the environment (m). By relaxing all of the constraints (finding the minimum energy of the system) the best estimate for the robot's full trajectory and the map (set of all landmarks) is found. Probabilistically, graph optimization results in the most likely trajectory and map given the set of observations and control inputs – also known as the full SLAM problem.

Recent approaches have used factor graphs to represent the SLAM problem, clarifying the structure to enable more efficient inference solutions [7, 8]. An example of a factor graph for graph SLAM and pose graph SLAM are shown in Figure 2.2.



(a) Graph SLAM (b) Pose Graph SLAM

Figure 2.2: Formulation of a (pose/feature) graph (left) and pose graph (right) for SLAM using factor graphs.

In a factor graph, each node represents a variable to be estimated, while each edge represents a constraint between the nodes characterized by a factor shown here as a solid dot. The graph is constructed (called the front-end) by using measurements to establish constraints, then the variables are optimized (called the back-end) to minimize the nonlinear least squares problem the factor graph represents.

Graph SLAM is targeted at environments with point landmarks that can be distinguished from the environment and localized. In pose graph SLAM, only the trajectory is found using the graph, then the map is built up from observations made at the (optimized) poses. In pose graph SLAM, edges between sequential poses arise from odometry measurements as well as between poses that observe a common landmark, resulting in loop closure constraints. Since features are not tracked directly, pose graph SLAM can use measurements of arbitrary features as well as point landmarks. This makes pose graph SLAM the preferred structure for dynamic environments.

While graph construction is particular to the environment, platform, and sensors, the back-end graph optimization algorithms are more widely applicable. Popular back-ends include square root smoothing and mapping (SAM) [23], hierarchical optimization for pose graphs on manifolds (HOGMAN) [24], g^{2}o [25], and incremental

SAM (iSAM) [9]. iSAM is a particularly popular back-end proposed by Kaess in 2008 [9]. iSAM incrementally updates the solution by only re-optimizing portions of the graph that are affected by recently added constraints. This significantly improves computational time allowing it to produce online estimates for the robot's pose. Recent developments in graph optimization have built off of iSAM including bayes tree [26] and multi-hypothesis iSAM2 [27]. Using iSAM as a back-end optimizer means that these advances could be incorporated with the developments of this thesis in the future.

2.1.2 Data Association

Graph construction (the front-end) relies upon to ability to recognize features/objects in the environment in order to insert constraints between non-sequential poses. Known correspondence is the name for the case when landmark recognition is trivial, and is generally accomplished by labelling objects in the environment with a defined *tag* that is easily detected. There are many different forms of tags for different sensing modalities and performance requirements. For visual sensors where localization and identification is needed, Olson et al. developed AprilTags that are easily detected and can be quickly interpreted [28].

With unknown correspondence, data association requires (1) segmenting recognizable elements in the environment, (2) search a list of previously observed landmarks for potential matches, (3) estimate the likelihood that the current observation corresponds to that landmark and (4) construct a constraint using the most likely correspondence. Each of these steps is a potential source of error, making data association a significant challenge.

Data association algorithms are specific to the types of objects/features that can be extracted from the environment as well as the method by which they are measured (the sensors used). Some common approaches include scan matching techniques such as iterative closest point [29], random sample consensus (RANSAC) [30], joint compatibility branch and bound [31], and nearest neighbour approaches [32] among others.

In general, data association is susceptible to perceptual aliasing where objects with similar appearances are mistaken for the same object and it is considered preferable to avoid adding any correspondence that is potentially in error due to the strong negative impact of false correspondences on the optimization results [2]. In some cases, this has driven researchers to maintain multiple hypotheses for the data association for extended periods, however this cannot be done indefinitely due to the size of the resulting hypothesis tree [2]. In dynamic environments, data association becomes further complicated as objects and features move and evolve. In a piecewise-deterministic model of the environment, a correct landmark recognition may still result in an incorrect correspondence if an event has occurred between the observations.

2.2 Detection and Tracking of Moving Objects

In the field of detection and tracking of moving objects (DATMO), several authors have attempted to model the changes of objects in the environment for the purpose of predicting the future position of objects.

The literature related to DATMO and dynamic mapping shows a variety of structures used to model environmental elements. The most common selection is a constant velocity model, which was used, among others, by Wang, Thorpe and Thrun in SLAM for urban environments [33], Pomerleau et al. to produce models of each dynamic point in a map of a campus environment [34], Wang et al. to develop a hidden Markov model for the state of each cell in an occupancy grid map [35] and Galceran, Olson and Eustice [36] as well as Ushani et al. [37] to model the motion of other vehicles on the road while they are occluded from an autonomous vehicle's sensors. Other approaches use Deep Neural Networks to generate state estimates (such as Murphy [38]), Probabilistic Transition Models (such as Moratuwage et al.'s Bernouilli Random Finite State Transition Models [39, 40]) or estimate periodic trends for presence/absence of objects (such as Fentanes et al. [41]).

Many of these solutions consider structured environments such as office buildings or streets, rather than natural environments such as marine or Arctic environments. An exception to this is Kimball and Rock's work on mapping the underside of icebergs using AUVs, which required tracking the iceberg's motion to remove the motion's contribution to the estimation error [42, 43]. Their work used a constant velocity model in [42] and both a constant velocity model as well as a four-segment (5 knots) third degree polynomial in [43].

Another important motion model structure from the field of DATMO is piecewise-deterministic motion, such as the structures used in [10]. In these structures, targets are assumed to follow a deterministic model during a period of time called an epoch. Eventually, this epoch is disturbed, by an instantaneous event, which starts a new epoch with a new instance of the same deterministic model structure. These piecewise-deterministic motion models mean a simple model such as a constant velocity model can be applied within an epoch, while infrequent disruptions to this model can be managed by detecting and appropriately responding to events. This approach is suitable to natural environments where rare forces change the otherwise constant velocity motion of landmarks.

2.3 Anomaly Detection

The piecewise-deterministic, dynamic SLAM problem formulated in this thesis requires distinguishing between observations of a landmark before and after a disturbing event. This problem is related to several other types of anomaly detection problems in the literature. These include:

- the data association problem: a challenge to identify and remove incorrect correspondences. Robust SLAM implementations have been designed to counter this challenge. This includes Switchable Constraints [44] and Dynamic Covariance Scaling [45] that each detect erroneous factors by comparing the covariance of a particular factor to a threshold.

- perceptual aliasing: a particular type of data association problem that typically occurs in an environment that has repeating structures that make it difficult to determine if an observation corresponds to a new or existing landmark. Several researchers have attempted to address this problem, by using methods such as max-mixtures models in [46] and Discrete-Continuous Graphical Models in [47].

- statistical local outlier detection: attempts to identify and if appropriate remove an anomalous measurement that was improperly captured, incorrectly assigned to a set or otherwise deviates from the natural variance of the population. As reviewed in [48], outliers are identified because they do not fit with the rest of the model. For example, Z-score analysis assesses each value's distance from the dataset's mean is relative to the typical variance of the data in the set. Then, a threshold is defined to remove observations that are further from the mean than that variance. Residual analysis [49] for outlier detection is a common approach in logistic regression.

- anomaly detection in signal processing: a challenge of detecting when an unusual behaviour in a system or network has occurred. Relies on correlations between 'correct' measurements by applying tactics such as principal component analysis (PCA), Kalman filtering, wavelet-based detection and hypothesis testing discussed for example in [50, 51]. In the SLAM domain, spectral analysis of time-series data was applied by Krajnik et al. to model the long-term variation of an environment with a Fourier transform [52]. Here, periodic processes were captured and predicted, allowing anomalous environment states to be detected, when the state differed from the predicted state by more than the threshold confidence level.

- score-based structure learning (SBSL): a branch of machine learning in which the edges of a graph are selected based on some quantitative criteria [53, 54]. It first determines the set of all possible graphs, then evaluates each according to criteria to select the best fit structure for the data. SBSL has been limited in its applicability due to the need to search for the set of possible graphs. Since in SLAM the number of possible graph structures is small (only a small number of edges are potential removal candidates at any time), this problem is avoided. Selection criteria are typically in the form of scoring functions [55]. For Bayesian networks, there are two main classes: Bayesian scoring functions and Information-theoretic scoring functions.

- edge selection problem: a particular part of SBSL in which factors are ordered from strongest to weakest, then a number of edges are selected to be retained in the graph. An example is degeneracy-aware loop closures that implement a threshold on eigenvalues of the loop closures such that only well-conditioned measurement directions (eigenvectors) are used in the loop closures [56]. Unlike typical solutions for edge selection that pre-select a number of edges to keep, a threshold is used in this case, based on the ratio between the largest and smallest eigenvalues. Only if this ratio is large are some eigenvectors removed.

In each of these fields, the essential task is to recognize when a new measurement differs in a measurable way from the existing population of measurements. Score-based structure learning is a useful tactic for this problem as it uses the existing graph structure to detect incorrect or anomalous edges that could indicate an event.

2.4 Dynamic SLAM

Extending SLAM algorithms into dynamic environments has proven to be difficult, due to the static assumption embedded in the original SLAM problem statement. In traditional SLAM it is assumed that landmarks are static so common observations of a landmark can imply a relative position change of the UAV. The problem with this assumption is evident when the dynamic scenario is placed into the factor graph form as in Figure 2.3.

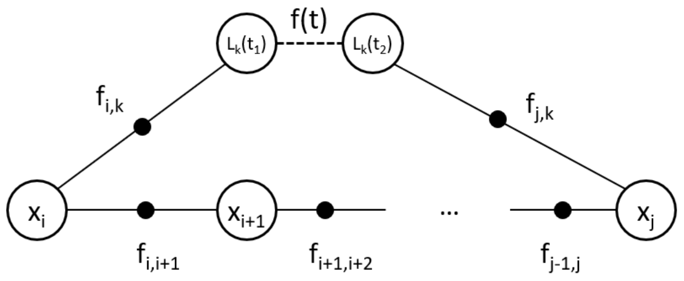


Figure 2.3: Factor graph representation of the *dynamic* pose/feature graph SLAM problem.

In the dynamic factor graph it is clear that ignoring the motion of the landmark is equivalent to an inaccurate loop closure constraint, which causes the solution to degenerate. For this reason, SLAM algorithms applied to dynamic environments have settled into three primary categories:

1. filter out dynamic objects and remove/ignore them;

2. track the dynamic objects, but store them separately from the static map (see DATMO review in Section 2.2)); or,

3. treat the dynamic objects as static during each mapping session and potentially moving in between sessions (semi-static).

2.4.1 Removing Dynamic Objects

The first option neglects the insight that dynamic objects and features can provide into the state of the environment as well as the processes that influence it. Further, in environments with few static landmarks, it reduces the number of potential loop closure constraints. This is the approach taken by the majority of SLAM algorithms in dynamic environments including [57, 58, 59, 60, 61].

This decision to only utilize static landmarks relies on the assumption that there are a sufficient number of static landmarks to successfully SLAM. In the marine Arctic environment, landmarks are sparse and all landmark information should be exploited. This thesis hypothesizes that when treated properly mobile landmarks can be incorporated in the graph, without degrading the graph estimate.

2.4.2 Using DATMO for Dynamic Objects Separately from SLAM

The second option separates the problems of target tracking and SLAM, so that the SLAM algorithm only uses the static elements in the environment. As with the first category, this disregard for the dynamic objects in the SLAM algorithm ignores information that could be helpful. Nonetheless, a significant amount of work on the second option has been performed. For instance, an early work by Wang, Thorpe and Thrun in 2003 studied the SLAM and DATMO problem within a probabilistic framework by estimating the velocity of dynamic objects and tracking them separately from the static landmarks used in the SLAM algorithm [33]. In 2015, Einhorn and Gross extended DATMO for SLAM by fading out objects that were labelled dynamic but moved beyond the sensor field of view – solving the problem of artifacts left behind when part of a dynamic object was removed in occupancy grid maps [62]. Another interesting approach was developed by Moratuwage, Vo and Wang as an extension to a multi-robot SLAM implementation based on random finite sets (RFS), thereby allowing for complex motion models beyond constant velocity [39].

While this method does provide insight to the motion of the dynamic objects, it does not use that insight to improve the SLAM estimate, which is a key limitation of this approach.

2.4.3 Assuming Semi-Static Environment

The third option includes the dynamic objects in the map used for navigation, but treats them as static at any particular instant. This has been accomplished by using multiple map layers each capturing objects with a certain temporal scale.

For example, Dynamic Pose Graph SLAM (DPG-SLAM), developed by Walcott-Bryant et al., extends pose graphs to low dynamic environments by using two maps: one active and one dynamic [29]. Here, dynamic objects are semi-static and only move between mapping passes of the robot. Using the two maps, means a history of the environment is maintained and a map of the current environment exists for every point in time [29].

Another tactic used by several authors (including Milford and Wyeth [63], Labbe and Michaud [64], Li, Li and Ge [65], Bacca, Salvi and Cufi [66] and Dayoub, Cielniak and Duckett [67]) to incorporate dynamic objects in the map, is to maintain two representations of the environment where one functions as short term memory (STM) (holding all recent observations) and the other as long term memory (LTM) (capturing features that are static or that persist in the environment for long periods of time). This is a biologically inspired solution that works well with different map representations and large numbers of moving targets, but remains limited by the assumption that the landmarks used in the long-term are static.

These methods miss the opportunity of capturing the underlying structure of a dynamic environment by simplifying the landmarks to a series of static positions.

2.4.4 Exceptions

Two notable exceptions exist to these three categories: a DATMO-inspired multi-layer RANSAC-based algorithm developed by Bahraini et al. in 2018 [68], and an object-aware, model-free semantic SLAM algorithm developed by Henein et al. in 2020 [69]. Both of these approaches attempt to simultaneously estimate the vehicle pose and the landmark speed, as is done in this thesis.

Bahraini uses a two-step algorithm, iteratively estimating the landmark states and velocities, then estimating the robot poses, until the system converges [68]. This work focuses on developing the data association to incorporate the velocity estimate using filtering techniques. This approach was shown to be successful in a fast application where the motion is easily detected by the front-end machine learning object detector.

The dynamic SLAM proposed by Henein et al. is similarly well suited to fast moving environments, such as autonomous driving scenarios [69]. Their approach is to segment rigid objects in the scene as either static or dynamic, then build a factor graph to jointly optimize the rigid body motion parameters, the landmark poses and the vehicle poses. The ternary factors that they propose to describe the motion model of a point on a rigid body are the most similar approach to the MBDFs developed in this thesis and the pose change vertices are an alternative solution to the event detection problem addressed in the thesis. However, [69] does not address the quasi-static environment or the long-term behaviour of dynamic loop closure factors in a unified way – two challenges that are essential to persistent mapping in an Arctic environment, which is studied in this thesis.

The recent acceleration of work to address dynamic environments indicates the importance of developing a solution for persistent mapping in the presence of dynamic landmarks. This further motivates the thesis objective to relax the static SLAM assumption to allow a UAV to operate autonomously in dynamic environments such as in the marine Arctic. These advances also support the hypothesis that exploiting landmark motion in predominantly dynamic environments will improve the SLAM performance compared to removing or segmenting landmark motion. The next chapter will define the dynamic environment of interest to enable the subsequent contributions.

Chapter 3 Problem Formulation

This chapter describes the context and problem formulation needed for the theoretical developments in the thesis. After defining the project scope (Section 3.1), the concept of piecewise-deterministic KMM are defined in the context of dynamic SLAM (Section 3.2), leading to the justification of the constant velocity KMM used to model icebergs in this thesis (Section 3.3). Finally, quasi-static motion is defined and the factors that effect whether a landmark or an environment can be considered quasi-static are discussed (Section 3.4).

3.1 Problem Scope

Inspired by the application of autonomous systems in the Canadian Arctic, this thesis explores persistent mapping in the context of unstructured environments populated with slowly moving or evolving amorphous landmarks, exemplified by icebergs, as shown in Figure 3.1.

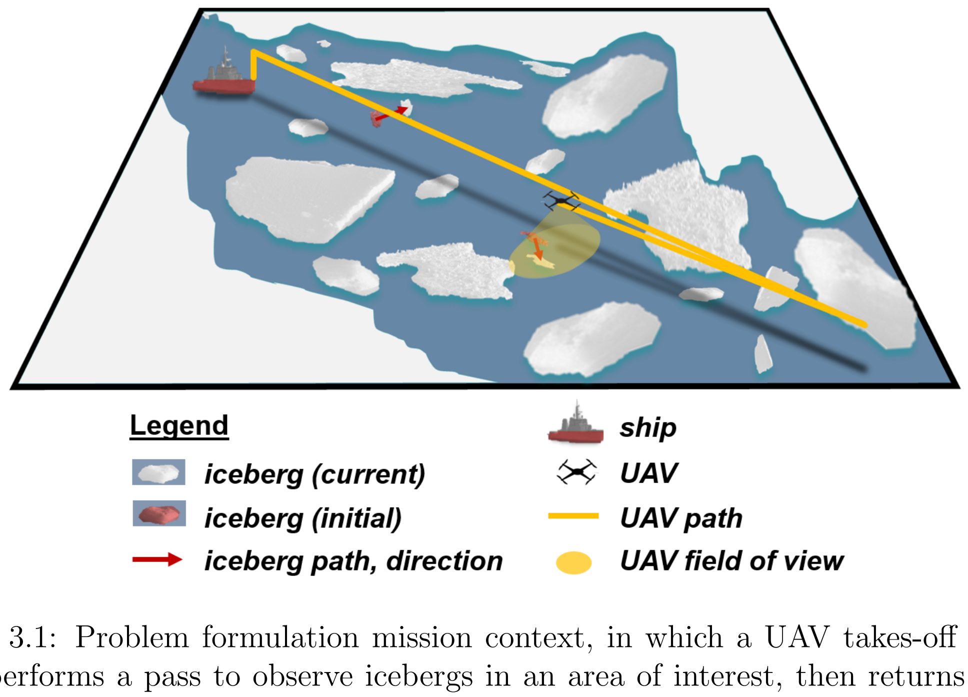


Figure 3.1: Problem formulation mission context, in which a UAV takes-off from a ship, performs a pass to observe icebergs in an area of interest, then returns to the ship. Note the few recognizable landmarks, all of which are, or may become, mobile.

The Canadian Arctic environment has several characteristics that pose a challenge for autonomous robots, and by choosing to address them, the resulting approach is more generally applicable to other complex environments. The characteristics of a UAV mission in the Arctic environment include:

- **unavailable global references (ex. GPS) and absolute heading reference by compass** requires the vehicle to use local positing references and SLAM techniques, this also occurs when operating underground, in deep water or in urban canyons,
- **limited communication infrastructure** requires autonomous behaviours rather than direct operator control,
- **sparse environment** with few recognizable landmarks means every landmark is valuable and should be exploited, unlike environments where there is the opportunity to select the best landmarks (ex. indoors, underground mines etc) and discard dynamic landmarks,
- **icebergs are amorphous landmarks**, this makes them good templates for general environments that likewise have limited structure,
- **all landmarks must be assumed to be potentially moving** now, or in the future, placing no prior expectation regarding environment structure,
- **landmark motion is very slow** typically noticeable only over the course of days and a solution that reduces the lower limit on motion detection is more generally applicable, and,
- **full six degree-of-freedom (6-DOF) vehicle** allows flexible path plans and is the most general pose model for an unmanned vehicle.

To address this complex, unstructured environment the following assumptions will be made:

- **first landmark observed is static:** The first landmark the vehicle observes, the one directly below the start position, is assumed static and known to be static *a priori*. This provides a critical link to the absolute reference frame that makes it possible to separate the UAV's motion from the average landmark motion. All other landmarks will be assumed mobile.

- **long mission duration:** The set of observations captured by the UAV are assumed to span a long enough duration that even very slow landmark speeds can be measured by comparing observations from the start and the end of the mission. This is justified by the UAV's task being persistent monitoring of a region.

- **omni-direction vehicle:** The UAV is assumed to be a quadrotor or other omni-directional vehicle. This simplifies the path planning problem compared, for example, to fixed wing UAVs that have to change their heading before moving in that direction.

- **perfect data association:** Perfect data association (detection and recognition of landmarks) is assumed, by using an abstracted landmark tagging system. This allows the thesis to focus on the persistent mapping problem and the challenges of associating and modelling mobile landmarks.

- **unlimited computational resources:** On-board computational power is assumed to be sufficient to perform any algorithm that can be performed on a desktop PC. This implies that the thesis does not study the efficient optimization for graph inference or optimize the code (implemented algorithms) for computational efficiency.

These assumptions are used throughout the thesis to develop a proposed approach to persistent mapping in the Arctic environment as characterized above. The work is demonstrated at five stages starting from the baseline static and progressively adding capabilities resulting in the complete PDQS-SLAM algorithm. While parent cases and variants are tested in simulations, exemplary cases are tested in both a simulated environment and in a controlled laboratory setting to validate the simulated environment.

3.2 Piecewise-Deterministic Dynamic Environments

The field of DATMO has identified that many natural and artificial dynamic environments can be characterized by piecewise-deterministic motion of the mobile elements in the scene [10]. Targets or landmarks in these environments follow a deterministic (constant-parameter) model for a period of time, called an epoch, then are disturbed, called an event, before starting to follow a new deterministic model of the same structure, with different values. The motion of a piecewise-deterministic target can be described as in Figure 3.2 which shows a target following a constant velocity KMM (constant slope when describing distance travelled over time) during three epochs.

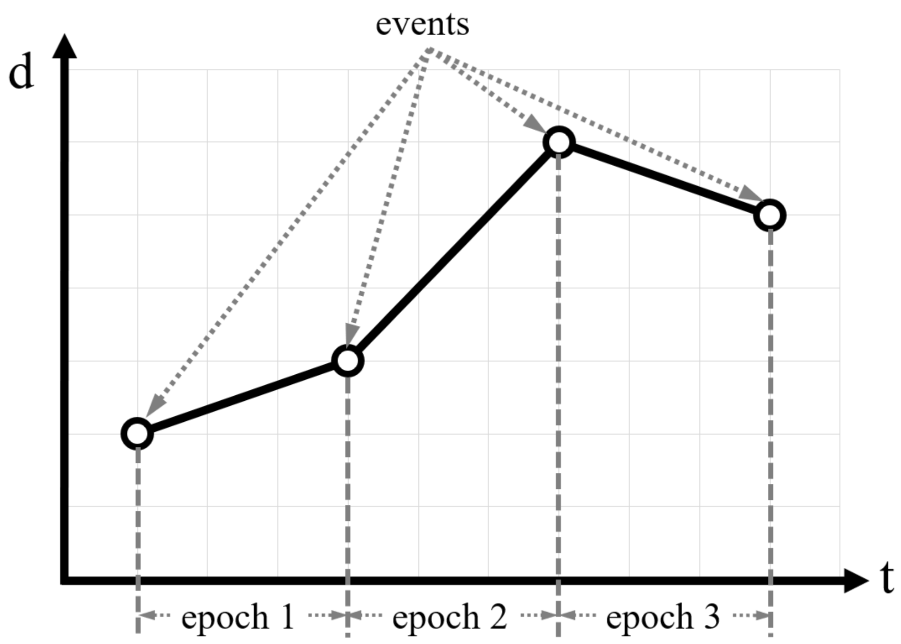


Figure 3.2: Definition of piecewise-deterministic motion, characterized by deterministic motion during epochs, disturbed by events.

An example from the research in DATMO, is the motion of an aircraft [10]. Typically pilots select a constant altitude, orientation and velocity for the majority of a trip, with distinct turns, ascents or descents before a new constant velocity segment. Natural phenomena such as large icebergs follow a similar pattern, with a period of constant velocity disturbed by an event such as a significant melt, strong storm or running aground, that results in the iceberg moving at a new constant velocity or becoming static. Different models apply in different environments (for example, a constant velocity model for icebergs [42], or constant acceleration models for objects in free fall) and while the velocities and accelerations the targets experience are not likely to be known a priori, the type (structure) of the model can be known in advance.

While piecewise-deterministic motion models can apply to the evolution of parameters of many kinds of models (changes in shape, colour, transparency, radiation levels etc), this thesis is particularly interested in changes in an iceberg's pose as it moves through the water. For clarity, when the model is of a landmark's change in position or orientation relative to an inertial reference frame, it will be referred to as a *kinematic motion model* (KMM).

In developing an approach to dynamic SLAM, Chapters 5 and 6 propose methods for estimating KMMs during an epoch while Chapter 7 proposes a method for identifying events that disturb the model. Modelling the motion during each epoch, then detecting and recovering from an event, results in the novel approach to dynamic SLAM presented in Chapter 8.

3.3 Iceberg Kinematic Motion Models

In the context of SLAM, attributing a piecewise-deterministic KMM to a landmark is necessary in dynamic environments to correctly compute the geometrical constraints between the vehicle and landmarks over time. Now, it remains to select the structure of motion model. While in specific cases, the choice of the motion model may be obvious, for general amorphous landmarks under the assumption of piecewise-deterministic, slow motion, a constant velocity (constant translational and rotational speeds) model is proposed. Conceptually, very slow motion lends itself well to constant velocity estimates, as the error due to small accelerations would be even smaller than the already low velocity. Large accelerations would manifest as events in the piecewise-deterministic framework and could be handled in that way. Icebergs are an example of amorphous, slowly moving landmarks and are proposed as a template by which the proposal to use a constant velocity model can be studied.

The motion of icebergs is affected by a large variety of factors making it difficult to predict without a complete model of the iceberg's keel shape, mass and density. Kimball and Rock studied iceberg motion models in order to develop an iceberg-centric map from an AUV in 2008 [42] and 2010 [43]. They use a motion model for the translation and rotation of the iceberg to transform measurements from inertial frame to an iceberg-fixed reference frame, thereby removing the warping of the map that results from the iceberg's motion. In [42], a constant translational and rotational velocity model was used, while in 2010 they use basis splines to model the motion. In particular, they choose a four-segment (5 knots) third degree polynomial (continuous, piecewise-linear acceleration) [43, 70].

Based on the results shown in [70], the shape of the iceberg was recoverable using either the constant velocity (i.e. constant translational speed and constant rotational speed) approach from [42] or the piecewise-linear acceleration approach in [43]. Thus, a **constant velocity model is a sufficient approximation for the iceberg motion** and will be used in this thesis, starting in Chapter 5.

3.4 Quasi-Static Dynamic Environments

In characterizing an operational environment as dynamic, there is an acknowledgement that an underlying pattern in the observations of landmarks, if not captured appropriately, would contribute to the error in the localization and mapping estimates. From this perspective, very slow motion could be considered a source of noise in an assumed static environment. While treatment of fast moving objects using DATMO-inspired approaches has been studied (see review in Section 2.4), the Arctic environment presents a different challenge, that of very slowly moving landmarks. These landmarks traverse huge waterways, but do so over many months. Some dynamic SLAM approaches treat these landmarks as semi-static – landmarks that move only between visits by the robot, as was done in Dynamic Pose Graph SLAM [20] – however, this thesis is interested in modelling landmark motion, even if it is very slow. The term quasi-static is proposed to describe environments populated with these very slowly moving landmarks. Quasi-static environments cannot be addressed through direct observations of the landmark and instead the motion of the landmark must be inferred over multiple observation cycles – teasing apart the uncertainty due to this very slow motion from the intrinsic observation noise.

In thermodynamics, quasi-static processes are characterized as progressing very slowly such that approximation as a series of equilibrium states is effective. The definition of *very slow* depends on the definition of the system – in particular its size and the time it takes to reach equilibrium. This concept can be extended to the understanding of very slow dynamic environments. Quasi-static dynamic environments are environments where consecutive observations capture a single equilibrium state, ie. they do not capture landmark dynamics. In the case of an environment whose landmarks move with constant velocity, consecutive observations of a quasi-static dynamic landmark would appear to be static. Just as in thermodynamics, quasi-static dynamic environments are characterized by an infinite series of these equilibrium states, where the landmark is seemingly static in the state, but over a longer period of time, the evolution of the system is clear.

It is useful to define a **quasi-static speed threshold**, v_{QS}, under which a landmark's motion can be considered quasi-static. Then, using prior knowledge of the **characteristic speed** of a landmark v_k, the decision can be made to apply a quasi-static SLAM method when $v_k < v_{QS}$.

Since a quasi-static environment is defined as one in which the motion of a landmark cannot be detected from two consecutive images of it, the minimum detectable speed is proportional to the minimum detectable change in position and inversely proportional to the time between the observations. In other words:

$$\text{minimum detectable speed} = \frac{\text{minimum detectable change in position}}{\text{time between observations}}. \tag{3.1}$$

Any landmark moving faster than the minimum detectable speed would have moved far enough between observations that the measured change in position is distinguishable from the noise on the position measurement. Any landmark moving slower than that can be considered quasi-static. In other words, the maximum speed to be considered quasi-static (v_{QS}) is also the minimum detectable speed. This gives the relationship between the quasi-static speed threshold v_{QS}, the minimum detectable change in position x_{min} and the time between observations t_{obs},

$$v_{QS} = \frac{x_{min}}{t_{obs}}. \tag{3.2}$$

To further specify v_{QS} in the case of constant translational velocity landmark (i.e. rotational velocity is zero), it is assumed that the UAV and landmark are both moving in the same direction, both parallel to the surface and the UAV's field of view is pointing directly perpendicular to the surface. Also, it is assumed that the first image is captured when the landmark enters the field-of-view (FOV) and the second image is captured just before the landmark leaves the FOV. Then, t_{obs} is given by,

$$t_{obs} = \frac{l_{FOV}}{(v_{UAV} - v_k)}, \tag{3.3}$$

where l_{FOV} is the length in the direction of UAV motion of the FOV and v_{UAV} is the speed of the UAV. The minimum detectable change in position, x_{min}, is equal to the uncertainty on the landmark position measurement. Rearranging Equations (3.2) and (3.3) for the minimum value of v_k that is detectable gives the equation for v_{QS},

$$v_{QS} = v_{UAV}\frac{x_{min}}{l_{FOV} + x_{min}}. \tag{3.4}$$

Finally, a landmark can be considered quasi-static if its characteristic speed follows the relationship:

$$v_k < v_{QS}. \tag{3.5}$$

Equation (3.4) makes it clear that, for the same characteristic landmark speed, a fast environment can be treated as a quasi-static environment by adjusting the following UAV or mission parameters:

- decrease the dimension of the FOV (or the altitude of the UAV);
- increase the speed of the UAV; or
- increase the uncertainty of the landmark position measurement.

This means that an environment is not inherently fast, slow or quasi-static. Rather those descriptions are only useful relative to the selected parameters of the mission (ex. UAV speed, altitude etc) and sensor(s) (ex. FOV, pixel size, noise in estimation etc.).

Note that the quasi-static threshold, v_{QS}, as defined in equation (3.2) can be applied to landmarks that have *both* or *either* constant translational speed and constant rotational speed by using the x_{min} in the appropriate units (i.e. metres for translational and radians for rotational speeds). However, t_{obs} can not be directly expressed in terms of the dimension of the field of view in the case of rotational speeds meaning equation (3.4) cannot be used for the case with non-zero, constant rotational speeds. The patterns listed above regarding the mission parameters' effect on the quasi-static threshold remain valid for the constant translational and rotational speeds (i.e. non-zero rotational speed).

The selected mission and sensor parameters that require icebergs to be treated as quasi-static are defined in Chapter 4. Subsequently, Chapter 6 designs a SLAM algorithm to estimate quasi-static KMMs.

Chapter 4 Experimental Set-Up

This chapter defines the set-up for simulations and experiments performed throughout the thesis. This includes the mission and testing environment (Section 4.1) inspired by the Canadian marine Arctic environment, and the quadrotor platform and sensors (Section 4.2). The scope of simulations and experiments (together: test cases) that evaluate the performance of the algorithms developed in this thesis are described in Section 4.3. In addition, a baseline case showing the performance of an established static SLAM algorithm is captured to validate the simulated environment relative to the experimental environment at model-scale (Section 4.4). Finally, static SLAM is applied to a dynamic environment (Section 4.5) to highlight the requirement for an approach that is not limited by the static assumption.

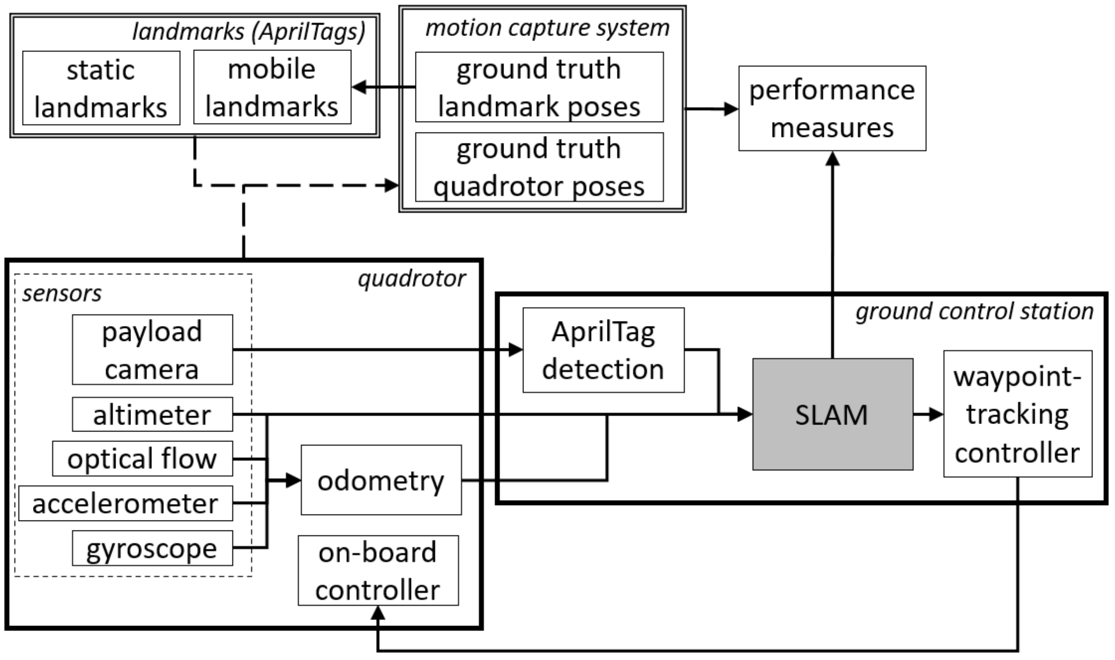


Figure 4.1: Elements used in experiments grouped by sub-system.

The experimental set-up, whether in the simulated or laboratory environment, includes several elements summarized in Figure 4.1 based on the sub-system they are part of. The thick bordered boxes show the platform and sensors (Section 4.2) necessary to test various SLAM algorithms (gray-filled box) as well as two supporting tools (AprilTag detection and a waypoint-tracking controller for the UAV). The double-outlined boxes are functions for the landmarks that form the map and ground truth capture system, they are external to the UAV and SLAM approach (Section 4.1).

4.1 Mission and Environment

While the approaches developed in this thesis are not specific to the type of autonomous vehicle, sensor suite, or environment, the inspirational case is of a UAV traversing a Canadian marine Arctic environment. This section details the mission and experiment parameters, as well as describes the fiducial tags used to represent landmarks, the mobile landmark implementation, and the simulated and laboratory flight spaces where the tests are performed. The mission and environment inspire the choice of platform (UAV) and drive the choice of appropriate sensors (described in Section 4.2).

4.1.1 Mission

The mission selected for experimentation is inspired by the Canadian marine Arctic environment, where a UAV flies over a region of interest, observes several landmarks that may be static, quasi-static or slowly-moving, to simultaneously localize itself within a map it is building of the observed landmarks. While the SLAM approaches developed in this thesis estimate both the position and orientation of the vehicle in three-dimensional space, to focus on error growth patterns without path complexities, the mission trajectory is reduced to a one-dimensional *pass*. A pass consists of a transit out to a waypoint some distance away from the start (take-off) point, then a return to the start point, in a nominally straight line that defines the X-axis. This choice does not result in any loss of generality as the factors constructed in the graph are not constrained in any way to the one-dimensional space. In fact, while the desired path will be one-dimensional, due to disturbances and uncertainties, the vehicle will inevitably respond with cross-track and altitude motions, in addition to the expected along-track motions.

In static environments, the SLAM loop closures constructed in the first pass have the largest impact to reduce the dead-reckoning drift localization error, while subsequent passes create more loop closures that can, to a lesser extent, continue to improve the estimate. In piecewise-deterministic environments that are disturbed by events, subsequent passes can improve the estimate, but they also mean more opportunities for events to disrupt the KMM. To study the estimation of the landmark KMM in an epoch (no events), a one-pass mission is suitable, while studying event detection and estimation of landmark KMMs in multiple epochs should use a two-pass mission with the event occurring *between* passes.

Mission Parameters: For such pass-based missions, the primary parameters that describe a particular mission are the pass length (l), UAV altitude (h), forward velocity of the UAV (v_{UAV}), and the SLAM pose update frequency (f). In the laboratory, the pass length is limited to 10 m and a reasonable altitude is 1-1.5 m (1.3 m was chosen). For a mission lasting approximately 1 minute, the UAV velocity should be 0.4 m/s along the X-axis. From experimentation with static SLAM and the quadrotor, the pose update frequency is selected to be 2 Hz. This was selected as a trade-off between increased computational time (at higher frequencies) and diminished waypoint tracking accuracy (at lower frequencies).

Landmark speed: Selecting the landmark speed for the mission is difficult due to the variety of iceberg size, shape and dynamics. In [42], Kimball and Rock determined the speed of the iceberg in their study to be 0.0918 m/s (translational) and -0.4644 rad/hr (-0.000129 rad/s). This translational speed is approximately 8 km/day. Assuming that their team selected a nearly-static iceberg that would remain in place long enough for them to perform the experiments, iceberg speeds up to 20 km/day (i.e. $v_{QS} = 20$ km/day) will be considered as a quasi-static environment in this thesis. Such speeds are not atypical.

In Equation (3.4), the threshold to treat landmark motions as quasi-static was defined relative to the speed of the UAV, the camera's FOV, and the minimum detectable change in a landmark's position. In Table 4.2, the accuracy for landmark pose measurements in the experiments was found to be 0.0611 m in the X-direction (the primary direction of motion). For $x_{min} = 0.0611$ m, $v_{UAV} = 0.4$ m/s and camera FOV $= 1.1$ m in the X-direction at an altitude of 1.3 m, speeds under 0.02 m/s can be considered quasi-static. For icebergs to be treated as quasi-static, the scale between the model-scale and the full-scale should be chosen such that 20 km/day in full-scale is geometrically and kinematically similar to 0.02 m/s at model-scale.

To study the estimation of motion models faster than the quasi-static regime, thus defining a **slow environment**, speeds of 40-80 km/day are used.

Mission Scales: The tests include both simulations and experiments, where simulations help explore performance in boundary cases that are anticipated to be challenging for the approach, while laboratory experiments demonstrate the robustness of the method to sensor and actuator noise, and other unknowns on actual systems. Two scales of tests are thus proposed: a full-scale one with a 100 m pass length (in simulations), and a model-scale one with a 10 m pass length (simulations and experiments). A 100 m pass is close to the limit for visible line of sight UAV operations. While the greatest use would be beyond line of sight, permissions to perform missions beyond line of sight is regulated. Buckingham π analysis was performed to keep geometrical and kinematic similarity between these two scales. In addition to pre-selecting the pass-lengths, the quadrotor speeds were also pre-selected to avoid high-pitch angles that interfere with an optimal landmark detection with the on-board downward-looking camera. In the full-scale tests, v_{UAV} is limited to 2 m/s which results in a maximum 5° vehicle pitch.

The full-scale test is chosen to have lengths 10× longer than the model-scale one, and to have times (durations) 2× longer than the model-scale. This means a 20 km/day (0.2 m/s) equivalent iceberg would have a full-scale speed in experiments of 0.1 m/s and a speed for model-scale experiments of 0.02 m/s. Also, the pose update rate at model-scale of 2.0 Hz (one measurement update every 0.5 s) is reduced to 1 Hz (one measurement update every 1 s) at full-scale. The summary of the mission parameters for the full- and model-scales are given in Table 4.1.

The scaling used for the environment is extended to the corrupting noise and covariances. For example, while at model-scale, the landmark position measurement is corrupted with zero-mean Gaussian noise with a standard deviation of 0.0611 m in the X-direction, at full-scale, it would be corrupted with a standard deviation of 0.611 m.

Table 4.1: Mission parameters for full- and model-scale experiments

parameter		**full-scale**	**model-scale**
pose update frequency (f)	[Hz]	1.0	2.0
pass length (l)	[m]	100	10
vehicle altitude (h)	[m]	13.0	1.3
vehicle velocity (v_{UAV})	[m/s]	2.0	0.4
landmark velocities (v_{ice}):			
quasi-static (20 km/day)	[m/s]	0.1	0.02
slowly-moving (80 km/day)	[m/s]	0.4	0.08

4.1.2 Fiducial Tag Landmarks

To focus development on the SLAM algorithm, the task to detect and identify landmarks is abstracted using the AprilTags fiducial tagging system [28]. An example of an AprilTag is shown in Figure 4.2. The AprilTag C++ library is easily integrated within the ROS middleware. The AprilTag detection node can detect and identify tags from either the actual or simulated camera images.

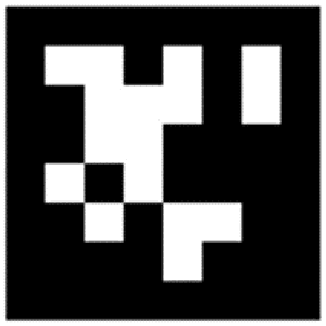


Figure 4.2: Example of a tag from the AprilTag fiducial tagging system [28] © 2011 IEEE.

The AprilTag system can discriminate between multiple tags in a single image, recognize their specific identification numbers and estimate their positions. The position estimate improves when the tag is perpendicular to the camera axis and when the tag is centred in the image. Accurate knowledge of camera parameters (such as focal length, principle point, and rectification matrix) is required for accurate pose estimation. In the simulated environment, the estimate is fairly accurate in all cases, while in actual measurements, when the quadrotor's camera captures a tag at a high pitch or roll angle, the orientation is not as accurately estimated. Zero-mean Gaussian noise with standard deviations specified in Table 4.2, is added to the simulated tag pose estimates to capture the observed experimental performance.

Table 4.2: Zero-mean Gaussian noise added to the tag pose measurements from the AprilTag detector to capture observed experimental measurement accuracy.

pose		standard deviation
x	[m]	0.0611
y	[m]	0.0467
z	[m]	0.0220
roll	[rad]	0.0651
pitch	[rad]	0.0713
yaw	[rad]	0.0201

Printed AprilTags of 10 cm width were glued to wood coasters and arranged on the laboratory floor. In the simulated environment, Gazebo box objects of width 10 cm (model-scale) or 1 m (full-scale) are generated, each with a different tag pattern applied to the top surface.

4.1.3 Mobile Landmarks

To implement mobile landmarks in the laboratory, LEGO MINDSTORMS NXT robots, Figure 4.3, provide the landmarks' locomotion. The `r2d2mipal` ROS package provides a driver [71] that controls the NXT motors' rotation rates over Bluetooth using a velocity controller based on [72], where the motion capture system provides the NXT pose feedback. The NXTs have a minimum speed of 0.01 m/s and are able to maintain a straight path over a 10 m distance with a deviation of less than 0.1 m.

For simulated mobile landmarks, a controller moves the Gazebo tagged objects at a constant velocity by publishing the desired (or reference) updated landmark position, calculated based on the Gazebo landmark ground truth pose.

4.1.4 Flight Space

The laboratory flight space is in the Oceans Tech Hub at Dalhousie University. It provides a long, controlled space without fluctuations in light or airflow that may

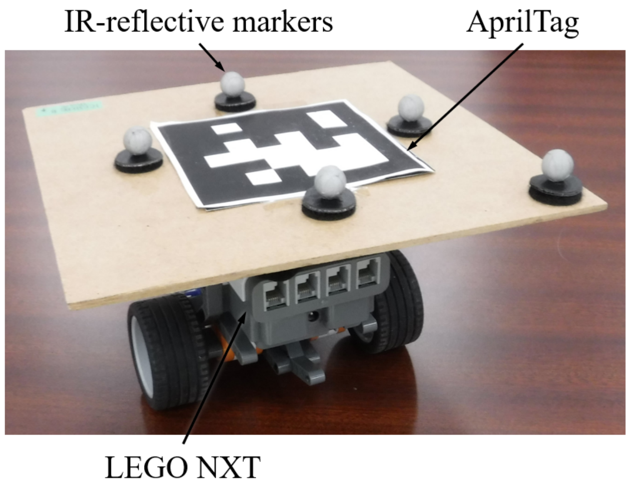


Figure 4.3: LEGO MINDSTORMS NXTs as model-scale iceberg analogue. NXTs provide locomotion to move the AprilTag landmarks in the laboratory flight space with IR-reflective markers for motion tracking.

affect UAV sensors or flight performance. The laboratory space is shown in Figure 4.4 with the quadrotor at the start (take-off) position and a total 21 landmarks laid out over a 10 m span, 19 of which are static (two are labelled for clarity) and 2 of which are mobile using NXTs. Note that the SLAM algorithm assumes all landmarks are mobile except the reference landmark located below the quadrotor at take-off. This reference landmark (always landmark 1) is assumed static throughout this thesis, however, a precisely known change in position is also acceptable.

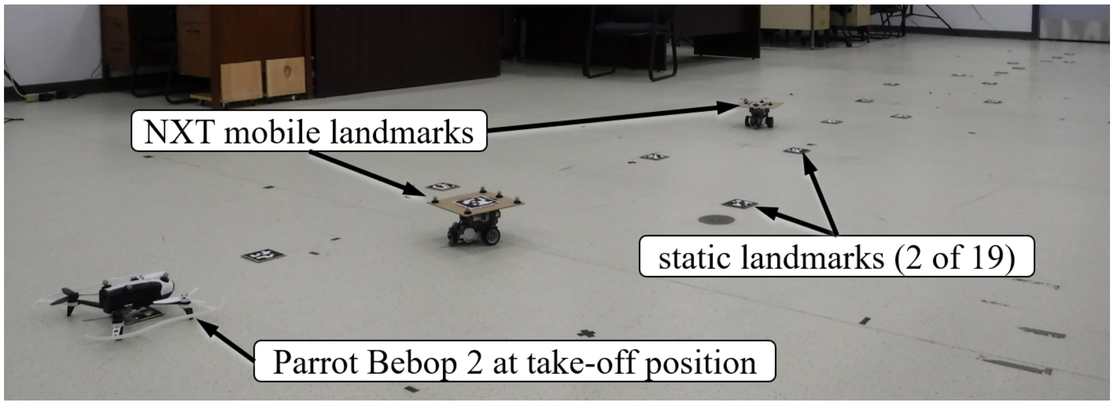

NXT mobile landmarks
static landmarks (2 of 19)
Parrot Bebop 2 at take-off position

Figure 4.4: Laboratory flight space used for simulation validation.

A motion capture system is also used in the laboratory set-up. The Motion Analysis Corporation motion capture system measures the UAV and landmarks' ground truth poses. Ground truth displacements are measured to within a fraction of a millimetre and attitudes to within 0.2° for each tracked object in the captured volume. It is capable of tracking multiple bodies with its eight identical infra-red (IR) cameras and, in the configuration used here, the captured volume is 12 m long × 4 m wide × 2.5 m high. Figure 4.5 shows one of the cameras as well as the Motion Analysis user interface.

(a) one of 8 IR-based motion capture cameras

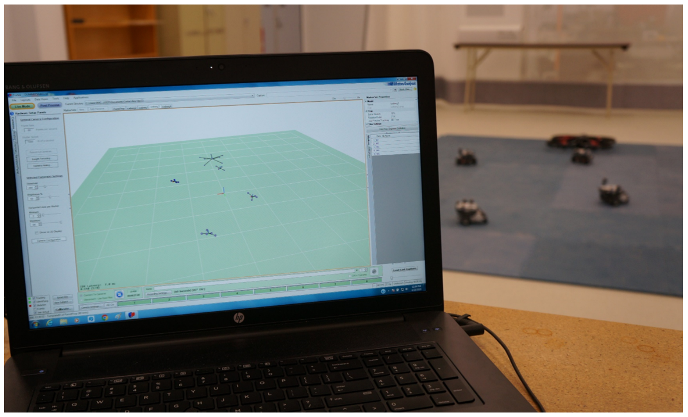

(b) motion capture system computer displays multiple bodies tracked in the flight space

Figure 4.5: Motion Analysis Corporation motion capture system as configured in the ISL in the Oceans Tech Hub.

Similarly, the simulated flight space is populated with a quadrotor, and static and dynamic landmarks. The Gazebo physics engine and Hector Quadrotor model (described in Section 4.2) are available as open source ROS packages (see Section 4.2.3). Figure 4.6 shows the Gazebo environment as well as the simulated camera output from the Hector model with one visible (tagged) landmark.

4.2 Platform and Sensors

This section describes the quadrotor platform's integrated payload sensor (visible wavelength optical camera) and vehicle sensor suite. The quadrotor's dynamics will also be discussed in this section. The quadrotor's body reference frame is shown in Figure 4.7 where the forward direction is the positive X-direction, an increase in altitude is a positive Z-direction and the positive Y-direction to port completes the right-handed reference frame. This figure also shows the reference positive roll, pitch

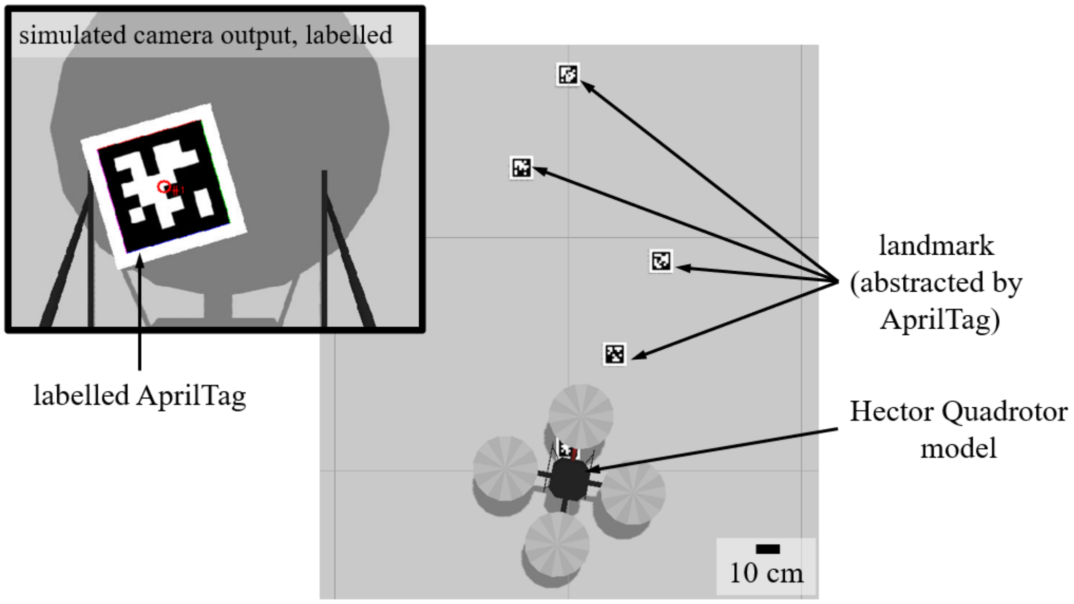

simulated camera output, labelled
labelled AprilTag
landmark (abstracted by AprilTag)
Hector Quadrotor model
10 cm

Figure 4.6: Simulated flight space environment using the Gazebo physics engine, Hector Quadrotor and landmarks (AprilTags). Inset shows the output from the quadrotor's camera after labelling by the AprilTag detection algorithm.

and yaw body rotation angles.

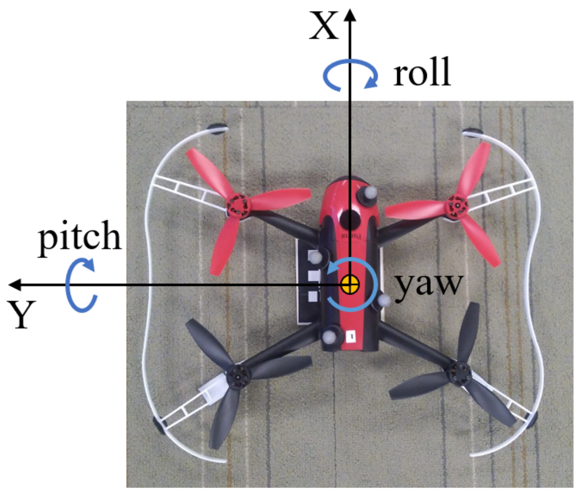


Figure 4.7: Quadrotor body reference frame, shown on Bebop quadrotor looking down in the negative Z-axis direction.

A quadrotor is a 6-DOF vehicle that is underactuated due to only inputs from four rotors (motors) in the horizontal plane. The impact of this underactuation is, for instance, a positive pitch angle (trim) to actuate a forward motion or a constant roll (list) to move laterally. Details on flight performance and control schemes for quadrotors are reviewed in [73] and [74] among other sources.

The following sections will describe the quadrotor used in the laboratory experiments (the Parrot Bebop 2) and the one used in simulations (the Hector Quadrotor).

4.2.1 Parrot Bebop 2 Quadrotor

The UAV used to validate the simulation environment is the Parrot Bebop 2 quadrotor shown in Figure 4.7. The Bebop's flight time is approximately 25 minutes per battery charge. Communications with its ground control station is through WiFi.

A. Navigation Sensors

The Bebop is integrated with an on-board stabilization controller based on an inertial measurement unit (IMU) (3-axis gyroscope and 3-axis accelerometer), compass (3-axis magnetometer), ultrasonic altimeter (under 4.9 meters), pressure sensor (above 4.9 meters) and an optical flow camera that only operates at low altitudes. For clarity, this camera will be referred to as the *optical flow camera* to distinguish it from the payload camera that captures images of the landmarks for SLAM. The Bebop also has a GPS chipset which, of course, does not work for indoor testing.

Odometry: Fusing the measurements from these sensors, the Bebop calculates and broadcasts an odometry estimate that includes its pose (translation and orientation) and velocity (translational and angular) relative to its take-off pose. Since the compass is disoriented indoors due to the presence of metal, the yaw angle estimate is inaccurate, and, consequently, the translation estimates are also inaccurate. However, the translational velocities and pitch and roll angle measurements are unaffected.

Optical Flow (Translational Velocities): The primary source for the translational velocities is the optical flow camera. Jaegle, Phillips and Daniilidis, demonstrated that optical flow noise models under optimal model parameters, with appropriate pre-filtering to remove outliers, follow a normal (Gaussian) distribution, while a Laplacian distribution is a better fit when outliers are present [75]. Since the Bebop filters the optical flow velocity estimates, a zero-mean Gaussian distribution model is used for the translational velocities.

Gyroscope: In the case of the roll and pitch, the gyroscope is the primary sensor, for which the Kalibr package defines a reasonable noise model [76]. This package uses a two-part noise model consisting of a slowly-varying sensor bias and an additive zero-mean Gaussian noise term. As the attitudes are filtered in the on-board Bebop controller, it is difficult to isolate the contribution of the bias. Instead, the assumed noise model is limited to zero-mean Gaussian noise.

Altimeter: The altitude of the Bebop is determined by fusing measurements from the on-board ultrasonic range finder and barometric pressure sensor. Since experiments were performed indoors and at low altitudes (well under the 4.9 m Bebop specification), the barometric pressure sensor should not have been used, yet black-box internal recordings indicated that it was being fused into the altitude measurement. While the noise model for a Gazebo simulated altimeter (as used in the Hector Quadrotor dynamics model [77]) uses a zero-mean Gaussian noise model to corrupt the altitude estimate, experimental measurements displayed both a zero-mean noise component and intermittent step-changes of 0.1 to 0.3 m, approximately every 10-30 s. This undesirable, intermittent step-change is attributed to the barometric pressure sensor being inappropriately fused with the ultrasonic sensor. Temperature had the strongest influence on the rate and magnitude of the barometric pressure step-changes. To reduce the impact, temperature changes were minimized in the laboratory and trials were run between breaks spanning several minutes, allowing the battery to cool down. Despite these interventions, the step-changes were observed in nearly all experimental trials.

Summary: The accuracy of the Bebop's odometry estimate was calculated by performing several test flights in the laboratory and comparing the odometry estimate with absolute position measurements from the motion capture system. The noise models of the elements of the odometry estimate are captured in Table 4.3.

B. Payload Sensor

The Bebop's payload sensor is a digitally stabilized, pan-tilt, 14 megapixel camera that for these experiments is pointed at its maximum pitch, 83° from the horizontal, towards the ground. The image streamed to the ground control station in near real-time is 856 $\times$ 480 pixels resolution with a camera horizontal FOV of 80°. For the

Table 4.3: Parrot Bebop 2 odometry zero-mean Gaussian noise model's standard deviations.

measure		standard deviation
translational velocity x	[m/s]	0.0294
translational velocity y	[m/s]	0.0141
altitude z	[m]	0.0759
roll angle	[rad]	0.0209
pitch angle	[rad]	0.0110
yaw rate	[rad/s]	0.02

experiments performed here, stabilization is disabled so that changes in the orientation of the image reference frame are tied to changes in the quadrotor orientation.

C. Dynamics and Controller

Researchers at Simon Fraser University (Burnaby, British Columbia) developed the `bebop_autonomy` driver [78] for the Bebop quadrotors which controls the UAV using the ROS (Robot Operating System) middleware (see Section 4.2.3). The driver provides the odometry estimate, the altimeter measurement and payload camera images to the ground control station. Notably, it does not broadcast raw gyroscope or accelerometer measurements nor images from the optical flow camera, meaning only the fused odometry estimate can be used for SLAM.

On-board control: The driver controls the UAV through a velocity command for the forward (X), lateral (Y), ascent (Z) and yaw (ψ) directions. UAV roll and pitch cannot be directly controlled through the driver. The on-board stabilization controller filters the pilot-commanded velocity input before actuation to smooth the flight performance. This happens whether the velocity input is commanded by a human pilot or an autonomous agent responsible for controlling the quadrotor's position (i.e. an autopilot).

Quadrotor Dynamics: Several on-board controller parameters can be adjusted to a pilot's preferences (for example, indoor/outdoor, smooth/aggressive or enabling GPS-automated behaviours). All default parameters for indoor flight were used with the exception of three settings that smoothed the flight given the low pose update rate (2 Hz). The maximum tilt angle was reduced to 5°, the maximum pitch/roll rotation rates were reduced to 300°/s and the maximum yaw rate was reduced to 60°/s. When the pilot sends a *hold* command (velocities in all directions are zeroed), the Bebop holds its position indoors (i.e. without GPS), at an altitude of 1 m, to within 0.15 m.

To assess the Bebop's ability to fly a straight path in the flight space, it was tasked with flying at an altitude of approximately 1.3 m over a length of 10 m using only the on-board odometry estimate. Over that distance, the quadrotor drifted in the positive Y-direction by 0.90 m. The cross-track drift primarily arises from the on-board controller's inclusion of the compass in its odometry filter.

Controller: A proportional-derivative (PD) waypoint-tracking velocity controller node was developed. It uses the estimated UAV position from the SLAM algorithm to calculate the velocity commands (in the X, Y, Z and ψ directions) to fly the quadrotor through a series of waypoints. A saturation limit for the maximum setpoint quadrotor velocity is applied to the calculated command. The experimentally-tuned control gains are listed in Table 4.4.

Table 4.4: Parrot Bebop 2 PD-controller gains

control gain	along-track (X)	cross-track (Y)	altitude (Z)	yaw (ψ)
proportional (k_p)	0.4	0.3	0.02	0.2
derivative (k_d)	0.18	0.2	0.0025	0.01

4.2.2 Hector Quadrotor Model

The Hector Quadrotor model [77] is used for simulations. It was developed and validated by the Technical University of Darmstadt (Darmstadt, Germany) and implemented with the ROS middleware [77]. The Hector Quadrotor model simulates the kinematics and dynamics of a quadrotor in response to velocity setpoint commands of the same form as the Bebop quadrotor's. It also includes sensor models, in this case a camera and altimeter, whose parameters can be adjusted to model the Bebop's sensors. Those parameters are then scaled for the full-scale simulation tests using the dimensional analysis described in section 4.1.1. The Gazebo physics-based engine is used to track the quadrotor's and landmarks' ground truth poses.

A. Navigation Sensors

Rather than individually simulating each of the navigation sensors, an odometry message was constructed for the Hector quadrotor to match the Bebop's odometry message. The ground truth measurements from the Gazebo physics engine are corrupted with zero-mean Gaussian noise with the standard deviations given in Table 4.3. In addition, to model the intermittent step changes observed in altitude, an offset of $o_{alt} \sim \mathcal{N}(0.2, 0.1^2)$ in meters at a time $t_{oalt} \sim \mathcal{N}(20, 10^2)$ in seconds. These intermittent step changes are only applied in the model-scale, as a full-scale experiment would occur outdoors where the barometer measurements would not cause this error.

B. Payload Sensor

Hector's default camera was adjusted to match the Bebop's payload camera (856 $\times$ 480 pixels resolution, 80° horizontal FOV, 83° pitch downwards) and AprilTag detection was also corrupted with noise as described in Table 4.2.

C. Dynamics and Controller

The default Hector Quadrotor was adjusted to provide similar a performance between the experiments and the simulations. This also applies to the platform dynamics and the waypoint-tracking controller.

Platform Dynamics: The Hector dynamics was used with their default settings, including a twist controller which translates command velocities (in the X, Y, Z and ψ directions) to motor commands. The mass, size and inertia of the Hector Quadrotor were matched to the Bebop based on the measures taken by [79], where the mass is 0.5 kg, the distance from the UAV centre of mass to the centre of each rotor is 0.12905 m, the moments of inertia with respect to the X and Y axes are both 0.00389 kg m^2 and the moment of inertia with respect to the Z axis is 0.0078 kg m^2.

Controller: The Bebop PD waypoint-tracking controller is applied to the simulated quadrotor with the control gains given in Table 4.5. To model the cross-track drift observed on the Bebop, an additional 0.02 m/s was sent to the quadrotor twist controller for the cross-track (Y-direction) velocity.

Table 4.5: Hector quadrotor simulator PD-controller gains

control gain	along-track (X)	cross-track (Y)	altitude (Z)	yaw (ψ)
proportional (k_p)	0.4	0.4	0.1	0.2
derivative (k_d)	0.18	0.18	0.01	0.01

4.2.3 Robotic Middleware

Information flow and communications is managed with the ROS middleware. ROS is a message-based infrastructure that connects independently developed functions [80]. Code is organized into functional blocks called nodes, while nodes are organized into packages. Nodes use a publish-subscribe architecture to asynchronously pass data structures called messages on topics. ROS is inherently decentralized and modular, meaning that a node that performs image processing can subscribe equally to a node that is simulating an image or to a node that is capturing an image from an actual camera. This allows a seamless transition from initial simulation-based development to real-world testing that is transparent to the functional code.

The ROS architecture allows nodes to be distributed across multiple platforms (computers). For development purposes, the nodes of the primary SLAM package are run on the ground control station computer while the autonomous system's flight computer sends raw data to the ground control station computer. Later, the final implementation could run all of the nodes on the flight computer with only health monitoring data shared with the ground control station computer.

The ROS middleware in this thesis is implemented on a machine running Ubuntu 14.04 (LTS) with ROS Indigo. This version of ROS uses Gazebo version 2.2.3. The ROS installation followed the recommendations on the ROS wiki. The following open-source packages were also installed:

- Hector Quadrotor (binary)
  `https://wiki.ros.org/hector_quadrotor`
- AprilTags with ROS wrapper (source)
  `https://wiki.ros.org/apriltags_ros`
- Tele-op Twist Keyboard (binary)
  `https://wiki.ros.org/teleop_twist_keyboard`

- gtsam 3.2.1 (source)
  `https://borg.cc.gatech.edu/download.html`

- Bebop Autonomy (source)
  `https://github.com/AutonomyLab/bebop_autonomy`

- r2d2mipal NXT Driver (source)
  `https://github.com/mipalgu/NXTdriver`

- Cortex Bridge (source)
  `https://github.com/unr-arl/cortex_ros_bridge`

- Rviz Covariance Plugin (source)
  `https://wiki.ros.org/rviz_plugin_covariance`

A ROS graph (visually) depicts the inter-relationships between the nodes and messages/topics within a ROS system. For the static SLAM case with a simulated quadrotor, the ROS graph is shown in Figure 4.8 where nodes are highlighted in light grey ovals and topics are shown in unfilled rectangles.

Nodes with thick black borders are nodes developed for this thesis: **/static_slam** implements the static SLAM algorithm and **/vel_pub_node** is the waypoint-tracking controller. Those with thin borders are from open-source packages. Nodes that publish messages to a topic have an arrow from the node to the topic, while nodes that subscribe to a topic have an arrow from the topic to the node. Debug nodes and topics that are not used are hidden from this graph for clarity.

The ROS graph makes the modularity of the system apparent. When the quadrotor and flight space are available for real-world testing, the Gazebo physics engine would not be necessary and the topics the `/gazebo` node publishes are published by the quadrotor's drivers instead. Further, the `/command/twist` topic published by the velocity controller would be subscribed to by the quadrotor's motor driver to close the control loop. This means the AprilTag detector node, the SLAM node and the velocity publisher node are unaffected by the transition from simulations to experiments.

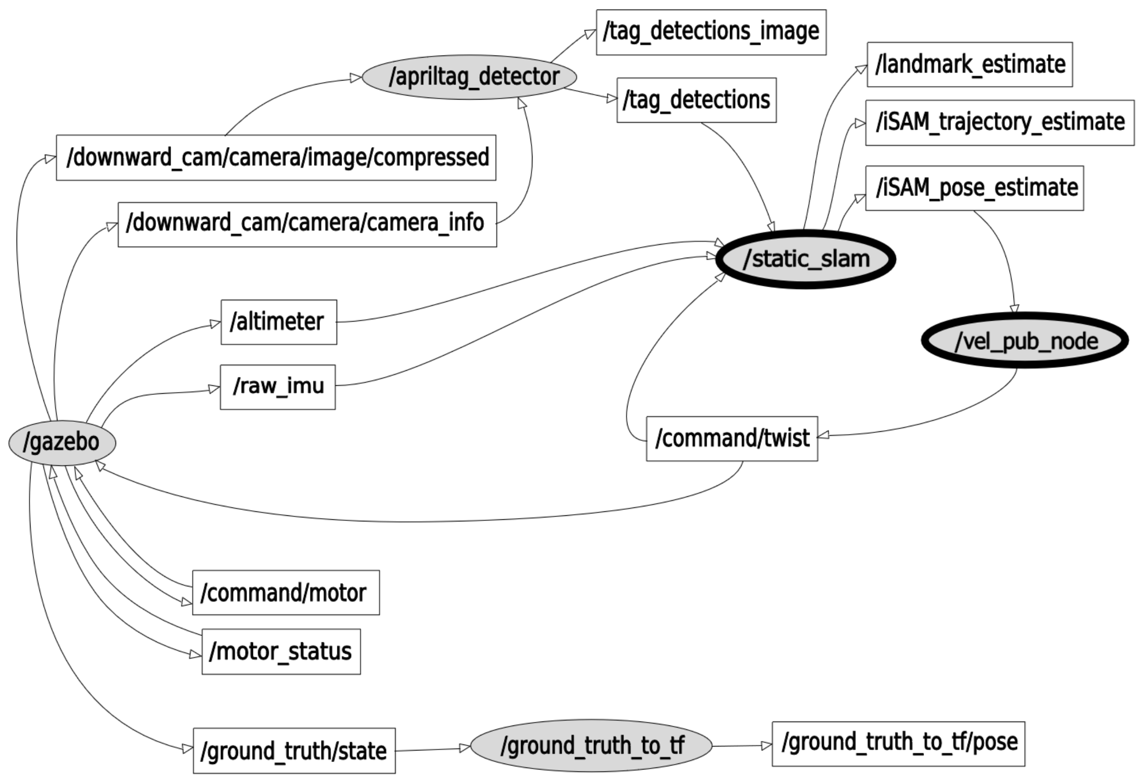


Figure 4.8: ROS graph for baseline static SLAM. Nodes shown as filled grey ovals and topics as unfilled rectangles. Thick outlined ovals are nodes developed for this thesis.

4.3 Summary of Test Cases

The proposed SLAM algorithms is evaluated with several performance tests. Each *test* studies a particular *case* (a specific mission, environment and scale), is executed as either a *simulation* or an *experiment* and comprises 10 repeat *trials* to quantify the consistency of the results. This is necessary given the additive zero-mean Gaussian noise. The taxonomy is summarized in Figure 4.9.

The cases of interest are summarized in Table 4.6 by the section number that discusses those results. A bold-faced font indicates both a simulation and an experiment is performed for the case, otherwise only a simulation is performed. Underlined cases are the *parent case* for the particular method, i.e. the case for which the method and any covariances or thresholds have been designed.

These tests are consistent with the cases of interest outlined in Section 4.1, with two mission types (one-pass and two-pass), two scales (full-scale and model-scale) and

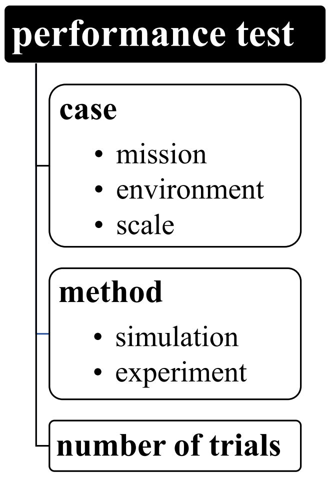


Figure 4.9: Taxonomy of performance tests in this thesis.

Table 4.6: Tests performed in the thesis organized by mission type, scale and environment variant with the section(s) that discusses those results.

mission	one-pass		two-pass	
scale	full-scale	model-scale	full-scale	model-scale
static	6.5.4	**4.3.3**		
slow	5.3.1	4.4, **5.3.2**		
quasi-static	5.3.3, 6.5.1	**6.5.2**	6.5.5	
anomalously fast	6.5.3			
one event			8.2.1	**8.2.2**
two events			8.2.3	

four environment types (static, slow, quasi-static and quasi-static with one event). Two other environments provide insight into the SLAM performance in Canadian marine Arctic environments. Firstly, an anomalously fast environment where the expected speeds are quasi-static yet one landmark travels at slow speed, can show if the quasi-static SLAM method is robust to imperfect prior knowledge. Secondly, a quasi-static environment with two events can show the ability to detect two different events that occur at the same time.

All of the test cases use the same base map which includes several landmarks over which the UAV performs either one or two passes. Figure 4.10 shows the true map of the static landmarks (red) and start positions of potentially mobile landmarks (black).

The landmarks are not numbered in any particular order, except that landmark 1 is always located at the take-off (start) position and is known to be static *a priori*.

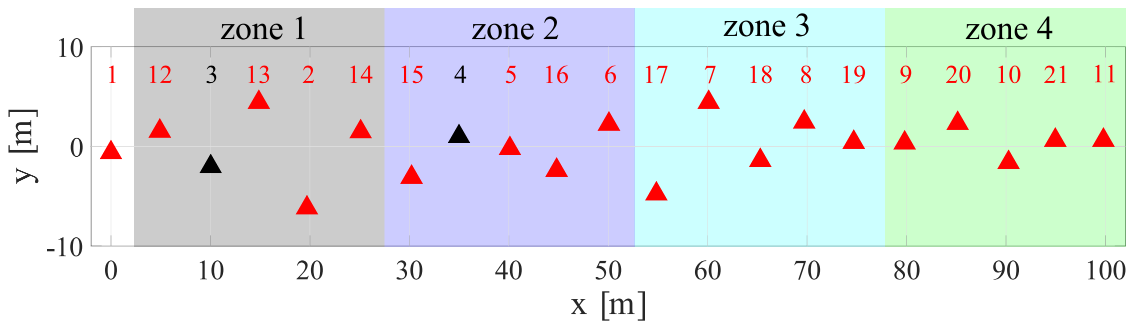


Figure 4.10: Ground truth map of landmarks for full-scale mission, with red triangles indicating static landmarks and black triangles indicating the start positions of mobile landmarks.

The ground truth map can be compared to the problem context diagram in Figure 3.1, with the start position similar to the take-off position on the ship and a sparse environment of landmarks (icebergs) two of which are mobile during the UAV's mission. The UAV's task is to perform a pass: take-off from the ship (at $x = 0$ m), travel in a straight line (along the X-axis) to a waypoint (at $x = 100$ m), then return to the ship for recovery. During this mission, the ship is assumed to remain stationary, or to accurately know its change in pose.

4.4 Simulation Environment Validation

To validate this simulated environment, a baseline static SLAM simulation and laboratory experiment is performed and compared.

4.4.1 Static SLAM

The baseline static SLAM algorithm selected, based on the literature review in Chapter 2, is pose graph SLAM using version 3.2.1 of the gtsam library and the iSAM graph optimizer [9]. An overview diagram of the (static) pose graph SLAM algorithm is shown in Figure 4.11.

The quadrotor payload camera captures images that are pre-processed with the AprilTag package from Olson [28]. The processed images are used to form the loop

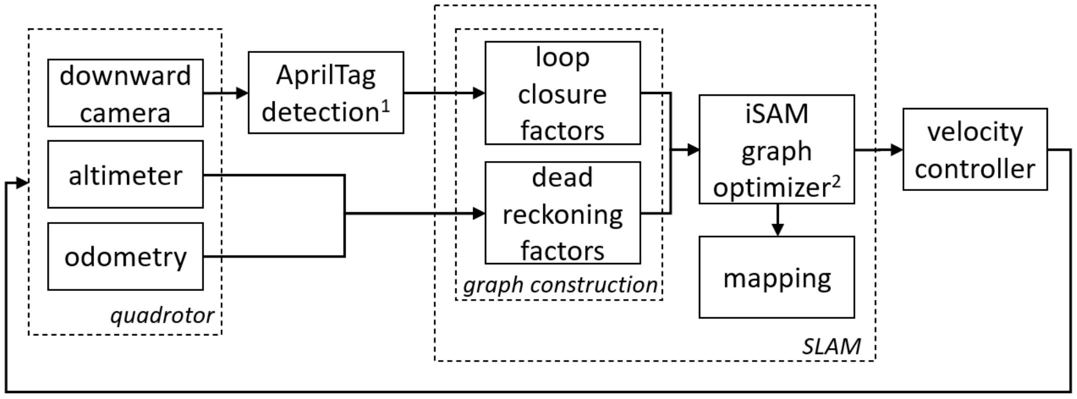


Figure 4.11: Algorithm for static pose graph SLAM. The algorithm uses two open source packages [1]AprilTags [28] and [2]iSAM from the gtsam library [9].

closure factors that link both sequential and non-sequential poses. The second set of factors are dead-reckoning ones that integrate the odometry measurements (translational velocities and gyroscope angles) over time to predict the vehicle pose. Together, these factors construct the graph structure which is passed to the iSAM graph optimizer [9]. The localization estimate is passed to the waypoint-tracking velocity controller to calculate the next control input to the vehicle, while the map is computed from the iSAM pose estimates and landmark observations. The update rate for the loop is tied to the desired pose update frequency for the mission.

4.4.2 SLAM Covariance Matrices

The most important parameters of a SLAM algorithm are its covariance matrices on the factors that form the graph. The covariance matrix represents the Gaussian noise on a factor, whether that be a unary factor that constrains one pose, like the prior, or a binary factor that constrains the geometrical transformation between two poses, like the odometry or loop closure factors.

A diagonal covariance matrix (variance on each element, with off-diagonals all zero) is assumed for all measurements. This implies that the elements of the uncertainty on the transformation are uncorrelated (ie. the uncertainty on the X element has no influence on the Z element uncertainty). For the odometry factors that connect quadrotor poses, the off-diagonal elements can be assumed small when the orientation angles are small (see linearization of quadrotor motion models, for example [74]).

The detection accuracy for AprilTags is similarly assumed to be uncorrelated for tags placed at near-zero tilt angles.

In general, large variances create flexible factors, meaning the optimizer can select a wider range of values for the pose(s), while small variances express a high degree of confidence in the measurement to create a smaller solution space. These matrices are described in this section and are summarized in Table 4.7.

Prior Factor and Covariance: The SLAM algorithm requires a prior factor to anchor its first pose to an inertial reference frame. Without this prior, there would not be a unique solution, since all other factors are relative (between poses). The prior is characterized by an initial robot pose, which is defined at $(x, y, z) = (0, 0, 0)$ and rotation angles all zero as well in the experiments here, and a covariance matrix specifying the certainty of this measurement. A larger covariance means the optimizer has more flexibility in where to connect the trajectory to the inertial reference frame, i.e. the first pose can deviate further from the prescribed initial value. The variances on the prior pose should be small to convey high certainty in these initial values and to constrain the trajectory tightly to the global reference frame.

Dead-Reckoning Covariance: The dead-reckoning factor links two consecutive poses by the geometrical transformation from the odometry measurement. The odometry accuracy is assumed to be zero-mean Gaussian (as discussed in Section 4.2), so the covariance for this translation is treated as constant. The constant covariance matrix is rotated based on the previous orientation angle estimate.

Underestimating these variances makes the resulting graph rely too heavily on the dead-reckoning estimate. However, overestimating these variances drives the graph to the loop closures and makes it more sensitive to incorrect data associations.

The variances populating the diagonal covariance matrix, as given in Table 4.3, are calculated based on the accuracy of the two Bebop odometry measurements that are combined to calculate the transformation. In the case of the attitudes and altitude, two measurements, with standard deviations, σ_1 and $\sigma_2 = \sigma$, are subtracted to yield a resulting measurement with a standard deviation of $\sigma_{2-1} = \sqrt{\sigma_1^2 + \sigma_2^2} = \sqrt{2\sigma^2} = \sqrt{2}\sigma$. In the case of the X and Y odometry that are based on the velocity measurements, the variance on the measured velocity is multiplied by the (constant) time step (the SLAM pose update rate) to determine the variance on the displacement.

Loop Closure Covariance: The loop closure factors are based on the AprilTag detection node output. This package estimates the position and yaw-orientation of landmarks with good accuracy, however the tilt (pitch and roll) of the landmarks is more uncertain. Loop closures are essential to removing drift from pose and trajectory estimates. If the covariances are too large, the loop closures are effectively unused in the estimate, while if they are too constrained (small) an incorrect data association or poorly localized landmark can lead to rapid divergence of the graph.

The variances used for the loop closure factors are calculated based on the landmark pose estimation accuracy in Table 4.2.

Table 4.7: Static SLAM covariance matrices described by the standard deviations on the diagonal.

		prior	**dead reckoning**	**loop closure**
roll	[rad]	0.01	$\sqrt{2}(0.0209) = 0.029557$	$\sqrt{2}(0.0651) = 0.092065$
pitch	[rad]	0.01	$\sqrt{2}(0.0110) = 0.015556$	$\sqrt{2}(0.0713) = 0.100833$
yaw	[rad]	0.05	$\sqrt{2}(0.02) = 0.028284$	$\sqrt{2}(0.0201) = 0.028426$
X	[m]	0.05	$(0.5)(0.0294) = 0.014700$	$\sqrt{2}(0.0611) = 0.086408$
Y	[m]	0.05	$(0.5)(0.0141) = 0.007050$	$\sqrt{2}(0.0467) = 0.066044$
Z	[m]	0.05	$\sqrt{2}(0.0759) = 0.107339$	$\sqrt{2}(0.0220) = 0.031113$

4.4.3 Performance Metrics

The SLAM performance is quantified through its accuracy of the UAV position, landmark map, and for non-static cases, landmark speed.

Localization and Mapping: The RMSE of the UAV position (x, y, z) estimates for the complete trajectory is calculated using ground truth measures at each iteration. The landmark map is found by transforming observations from the body-frame to the world-frame using the pose estimates, then averaging all observations of each static landmark. This map is then compared with the ground truth static landmark positions to calculate the RMSE. These metrics are consistent with traditional SLAM algorithms and are the primary measure of successful performance as without an accuracy trajectory, the UAV may be unable to complete its mission and safely recovered.

Landmark Speed: For slow and quasi-static environments, the true and estimated speeds in the X-Y plane of each landmark are compared using a bar chart, and the speed accuracy is reported as the RMSE separately for the static and moving landmarks. The landmark speed is necessary to predict the landmarks pose in the future. However, because the speeds are by definition slow or quasi-static, the impact of inaccurate speed estimates is smaller.

Boundary between Moving and Static Landmarks: While the SLAM algorithms in subsequent chapters treat all landmarks as potentially mobile, it is useful to define a boundary or threshold to distinguish moving landmarks from static ones for reporting performance metrics. The boundary for a moving landmark is defined as one standard deviation above the mean landmark speed, where these statistics are found using the estimates for all landmarks. Since the speeds in quasi-static and slow environments are by definition low, the ability to discern whether landmarks are moving or static, especially close to the UAV's take-off location, is more important than the precision of the speed estimate. This boundary provides an indication of the slowest speeds that can be detectable or discerned as non-static.

Performance Consistency: Each experiment is repeated 10 times to report the consistency of the performance. Although this is a small population to perform statistical analysis, a simulation study was performed on the static environment and after approximately 8 trials, the statistical measures (mean and standard deviation) converged. A Gaussian model of the statistics of these 10 trials is selected to match the Gaussian models in the underlying SLAM algorithm and sensor noise models. These statistics are reported in a table, while two figures (a trajectory and landmark map, and landmark speed bar chart) are plotted for the trial with the median-performance. An analysis of the measures in the table and the median-performance trial exposes trends between cases.

Reporting: For each case in Chapters 4, 5, 6, and 8 the results presented take the form of:

1. quantitative performance summary (table)
    - localization error (norm and x, y, z components)
    - mapping error (norm and x, y, z components)

- mean static landmark speed error
- mean dynamic landmark speed error
- mean speed boundary

2. planar (X-Y) map of UAV trajectory and landmark positions for median-performance trial (figure)

3. bar chart showing the estimated speed for each landmark, and the boundary between static and mobile landmarks for median-performance trial (figure)

Although the SLAM estimate computes the full 6-DOF poses, accuracies are reported only for the position, as the orientation is controlled close to zero for the selected missions. This is because the errors in the attitudes are bounded.

4.4.4 Validation Results

Using the described static SLAM approach (Figure 4.11), platforms and sensors (Section 4.2) and model-scale environment (Table 4.1), the one-pass, model-scale, static environment test is performed in both the laboratory and simulated environments. The test results are summarized in Table 4.8, with the trajectory and map of the median-performance trials of the experiment shown in Figure 4.12 and the simulation shown in Figure 4.13.

Table 4.8: Localization and mapping errors for baseline static SLAM in the experiment and the simulation.

metric		**experiment**	**simulation**
localization error (norm)	[m]	0.41 ± 0.15	0.46 ± 0.17
x error	[m]	0.14 ± 0.07	0.20 ± 0.08
y error	[m]	0.18 ± 0.07	0.14 ± 0.05
z error	[m]	0.28 ± 0.16	0.33 ± 0.20
mapping error (norm)	[m]	0.30 ± 0.13	0.54 ± 0.24
x error	[m]	0.09 ± 0.04	0.26 ± 0.16
y error	[m]	0.06 ± 0.03	0.18 ± 0.11
z error	[m]	0.26 ± 0.15	0.35 ± 0.25

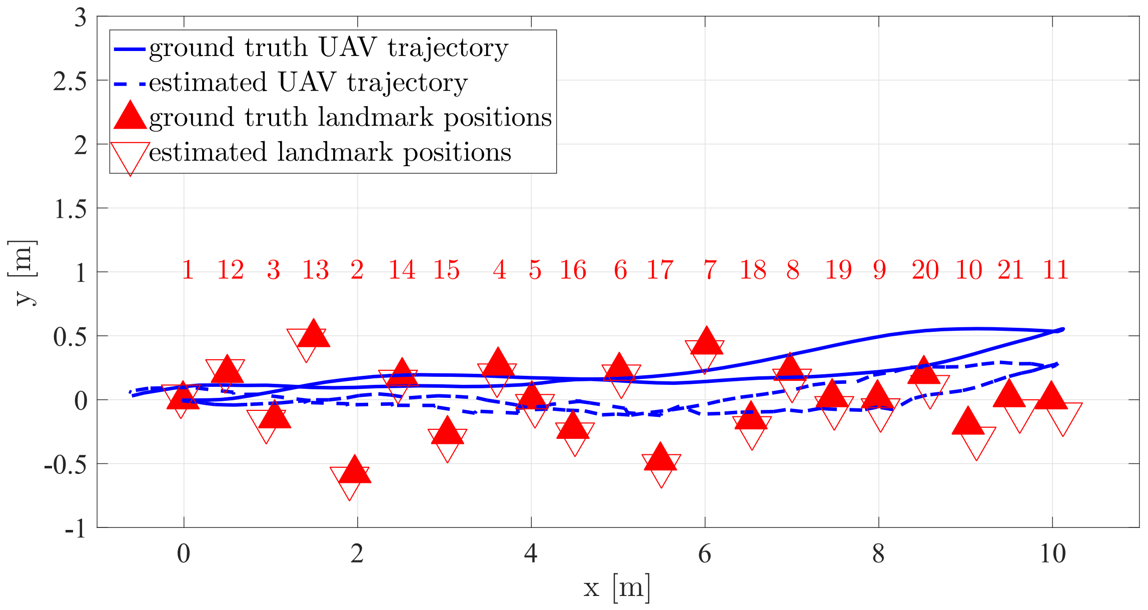


Figure 4.12: Experimental result: planar map of baseline static SLAM in model-scale, static environment.

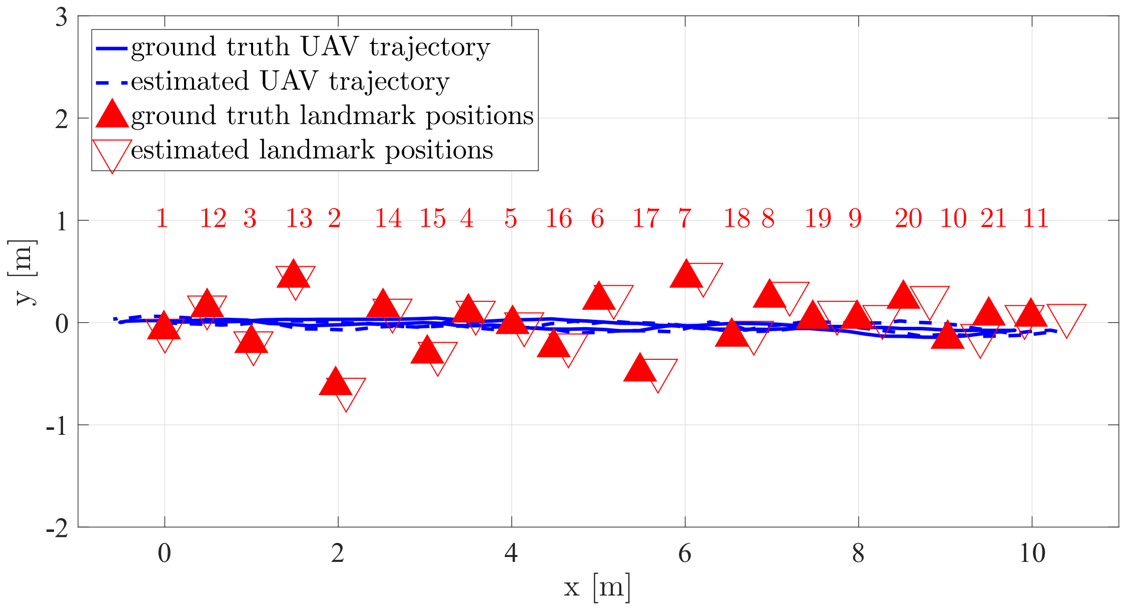


Figure 4.13: Simulation result: planar map of baseline static SLAM in model-scale, static environment.

A visual comparison of Figures 4.12 and 4.13 reveals two primary patterns. Firstly, the cross-track drift of the Bebop quadrotor is notable compared to the simulated vehicle, yet the SLAM algorithm is successfully able to estimate much of this drift from the desired trajectory to ensure an accurate map of the environment and trajectory estimate. Secondly, the landmark positions are over-estimated in the X-direction in both the simulation and the experiment. This along-track underestimation arises from a pitch error that drives a translation in the altitude (Z) rather than in the horizontal plane (X-Y). Due to the relatively large variance on the pitch for loop closure factors (0.1 rad), the loop closure factors are uncertain for this dimension, and unable to drive down the drift in the odometry pitch estimate. In simulation results, the X-direction localization error is larger than in the experimental results, which also drives the larger mapping error in the along-track direction, discussed below.

Turning to the quantified measures of performance in Table 4.8, the localization and mapping error are of similar magnitudes in simulations and experiments. The large altitude error is due primarily to the altimeter's intermittent step-changes discussed in Section 4.2. These step changes require the altitude measurement variance to be larger than the along-track and cross-track positions, resulting in more uncertainty on that element of the pose estimate. In addition, the altitude error also arises from the uncertainty in landmark pitch and roll estimation for loop closure factors that are unable to reduce the drift from the odometry pitch estimate.

On the mapping side, the error is statistically higher in the simulation than in the experiment, especially in the X-Y plane. The simulation models the worst case situation, where there are uncorrelated errors in all dimensions of the landmark estimate concurrently, while in the real system there is coupling between the contributions (for example, if the range estimate to the landmark has an error it propagates to the position of the landmark in the X-Y plane). The result is that the simulated variability in landmark accuracy calculated from a large set of measurements is conservatively larger than the error on any one particular landmark.

While the simulated platform and environment do not perfectly capture the laboratory system and environment, the simulation provides a reasonable model to assess the impact/sensitivity of the proposed SLAM algorithms (MBD-SLAM in Chapter 5, MBQS-SLAM in Chapter 6 and PDQS-SLAM in Chapter 8) on the localization and mapping performance.

4.5 Applying Static SLAM in Dynamic Environment

Before developing the proposed dynamic SLAM algorithm, it is useful to see how the baseline pose graph SLAM algorithm performs in the thesis' simulated dynamic environment. The simulation result in Figure 4.14 is for the case with two moving landmarks (landmarks labelled 3 and 4) travelling with different velocities (4 cm/s and 8 cm/s respectively). They are labelled using solid black triangles to mark their ground truth starting (left in figure) and ending (right in figure) positions. These speeds correspond to 40-80 km/day, i.e. the slow environment type for the one-pass, model-scale experiment. The trajectory and map of the median-performance trial of the static pose graph SLAM algorithm is shown in Figure 4.14.

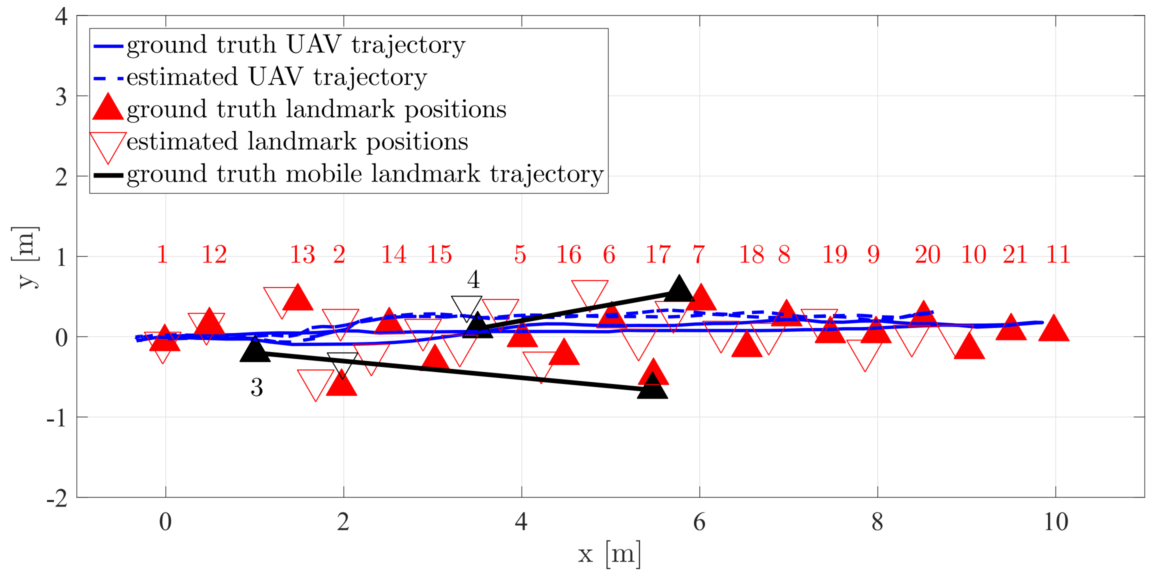


Figure 4.14: Simulation of static SLAM in a dynamic environment with two slow mobile landmarks (labelled 3 and 4), at 8 cm/s and 4 cm/s, respectively.

Figure 4.14 clearly demonstrates that the map generated, with the static environment assumption, is significantly less accurate. Although the quadrotor is able to complete its trajectory, the estimate of the trajectory deviates from the ground truth, so much so, that the quadrotor only estimates that it travelled 80% of the total distance. In fact, the localization error (norm) is 0.84 m (± 0.33 m) and the mapping error (norm) is 0.82 m (± 0.34 m); each much larger, and more variable between repeat trials, than the baseline case in Figure 4.13, where the localization error was 0.46 m (± 0.17 m) and the mapping error was 0.54 m (± 0.24 m).

To understand the performance of the baseline SLAM in a dynamic environment, it is interesting to follow the progression of the SLAM estimate over the quadrotor's mission. For the first half of the pass (while travelling from $x = 0$ m to $x = 10$ m), the graph consists predominantly of dead-reckoning factors. These factors are typically quite accurate early in the mission, then begin to deviate over time due to accumulation of sensor errors. As the quadrotor begins its return trip (from $x = 10$ m to $x = 0$ m), loop closures are added to the graph as more and more tags are re-observed. For static landmarks, these loop closures reduce the quadrotor's position estimate drift, however loop closures involving moving landmarks assign the motion of the landmark to the quadrotor's change in position.

Depending on both the balance of certainty between dead-reckoning and loop closure factors (quantified in the covariance matrices) and the location of these poorly-formed loop closures in the graph structure, the impact can be damaging, or even cause the trajectory solution to diverge. In this case, since most of the moving landmarks' motion is in the same direction as the positive along-track motion of the quadrotor, the optimizer holds landmarks 3 and 4 close to the positions where they were first observed (where the quadrotor's pose estimate had not yet accumulated much drift) and shortens the estimated pass length.

This simple case motivates the requirement to develop a SLAM algorithm that does not rely on a static map to provide an accurate localization solution in dynamic environments.

Chapter 5 Proposed Model-Based Dynamic SLAM

This chapter proposes a novel structure, the model-based dynamic factor (MBDF), with which to extend and apply pose graph SLAM to a spatio-temporally evolving environment. This is the first step towards relaxing the static environment assumption in traditional SLAM approaches.

This work follows from the concepts of landmark KMMs, piecewise-deterministic dynamic environments (Chapter 3) and the demonstrated need to not treat dynamic landmarks as static ones (Section 4.5). The proposed MBDFs are applicable to a single epoch of a piecewise-deterministic dynamic environment where the landmark KMM is a constant velocity model.

MBDFs are used in place of static loop closure factors in the traditional SLAM formulation. They replace the static landmark assumption with one where landmarks evolve with time, following a motion model whose parameters are estimated in real-time. While the landmark evolution implemented here is restricted to landmark KMM, and specifically constant velocity KMMs, this concept can also apply to the time evolution of other iceberg landmark features like size, shape, texture, color, or reflectivity to name a few.

After first defining the MBDFs and detailing two proposed KMM parameter estimation approaches in Section 5.1, this chapter proposes model-based dynamic SLAM (MBD-SLAM) as a proof-of-concept that KMM parameters can be inferred from a small number of landmark observations, while simultaneously estimating the UAV's trajectory in Section 5.2. MBD-SLAM is tailored to slow environments, as defined in Section 4.1, and its performance is shown for three cases:

1. one-pass, full-scale, slow environment, simulation (Section 5.3.1),
2. one-pass, model-scale, slow environment, simulation and experiment comparison (Section 5.3.2); and,
3. one-pass, full-scale, quasi-static environment, simulation (Section 5.3.3).

The first case is the mission for which MBD-SLAM is designed, while the second case can be compared with the performance of static SLAM in the slow environment (Section 4.5) to demonstrate the improved performance of MBD-SLAM. Finally, the third case captures a limitation of MBD-SLAM and motivates tailoring a SLAM approach to quasi-static environments in Chapter 6. The performance results of MBD-SLAM approach is discussed in Section 5.4.

5.1 Model-Based Dynamic Factors

In pose graph SLAM, loop closure factors model geometric constraints between vehicle poses by comparing measurements of a common landmark that is assumed to be static. To facilitate relaxing the static environment assumption, these loop closure factors must capture the landmark's dynamics. Specifically, the landmark KMM must be part of the factor's construction before applying the standard graph optimization. MBDFs allow the landmark KMM to integrate directly into the factor graph, so the graph is responsive to the environmental dynamics.

Figure 5.1 shows the pose graph representation in a dynamic environment where poses x_i and x_j are related through a *model-based dynamic factor* (MBDF) notated by $f(v_k, \Delta t)_{ij}$ where v_k is the velocity of landmark k that was observed at the two poses.

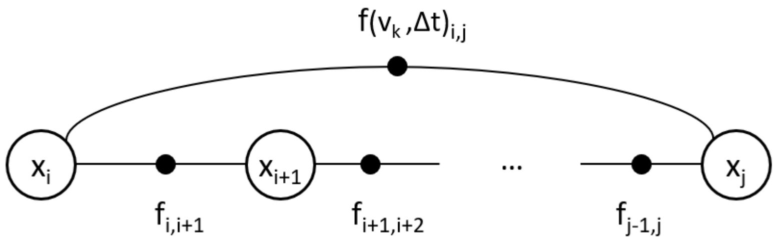


Figure 5.1: Factor graph representation of the proposed MBDF for pose-graph SLAM assuming a constant velocity landmark motion model [81] © 2019 IEEE.

Like static loop closure factors in Figure 2.2, MBDFs depend on two observations of a common landmark, however, they also require a model of the landmark transformation over time. Compared with dynamic pose and landmark graph SLAM, MBDFs for pose graph SLAM combine three factors from Figure 2.3 into one. Once the MBDF is constructed, the optimizer can, as usual, estimate the variables that minimize the error on each factor in the graph.

This graph can also be described as a cost function that is minimized to find the optimal trajectory (X^*) and landmark velocities (V^*), as follows:

$$X^*, V^* = \underset{x,v}{\arg\min} \sum_i ||f(x_i, u_i) - x_{i+1}||^2_{\Sigma_i} + \sum_{ijk} ||f(x_i, x_j, v_k) - z_{ij,k}||^2_{\Lambda_{ij}}. \tag{5.1}$$

In this formulation, the first term minimizes the error (Mahalanobis distance) on the dead-reckoned odometry factors and the second term minimizes the error on the observation-based loop closure factors. Here, u_i is the pose change in the body reference frame calculated from odometry, x_i, x_{i+1} and x_j are poses as in Figure 5.1, $z_{ij,k}$ is the observed distance between pose x_i and x_j calculated through the common landmark k, and Σ_i and Λ_{ij} are covariances on the respective factors.

In static SLAM, the difference between two observations z_{ik} and z_{jk} of the common landmark k is calculated as $z_{ij,k} = z_{ik} \ominus z_{jk}$, where $\ominus$ is the *reverse operator* [11] that finds the difference of the two observations while accounting for rotations. Since the landmark is assumed static, $z_{ij,k}$ is a measure of the transformation of the platform between the two observations. Hence, this measurement can be compared with the estimate of the measurement (h_{ij}), using values for x_i and x_j from the graph as $h_{ij} = x_i \ominus x_j$. Graph optimization then minimizes the error $e_{ij,k} = h_{ij} \ominus z_{ij,k}$ for all factors. The same process is repeated here taking into account the landmark KMM.

MBDF Construction

For the proposed MBDF, the measurement of the distance between poses, $z_{ij,k}$, is found as in the static case ($z_{ij,k} = z_{ik} \ominus z_{jk}$) since the velocity is not available as a direct measurement. The estimate h must therefore be modified to remove the contribution of the landmark's motion. This modified estimate of the static measurement is referred to as h_s and is found as $h_s = (x_i \ominus x_j) \ominus T_{k^i k^j} = h \ominus T_{k^i k^j}$. The factor's error is $e = h_s \ominus z_{ij,k}$.

This factor is visualized in Figure 5.2. Consider the two measurements of the landmark. The first measurement of the landmark, labelled T_{ik}, is a measure of the landmark's position at time t_i, when the robot was in position x_i, in the robot's body frame at x_i. The second measurement of the landmark, labelled T_{jk}, is a measure of

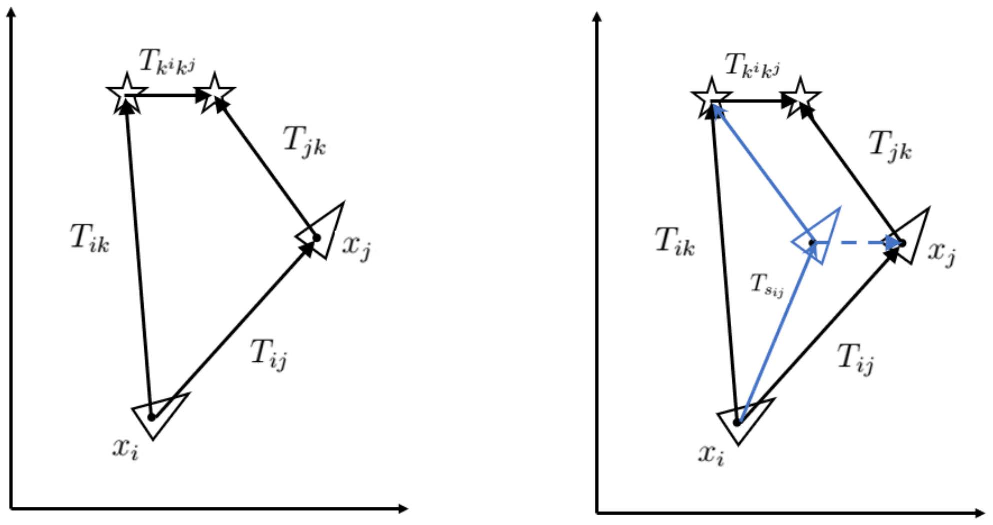


Figure 5.2: Geometric derivation of MBDFs. Components used to form MBDFs (left) and the use of a constructed point to transform the system to a comparable static representation in blue (right).

the landmark's position at time t_j, when the robot was in position x_j, in the robot's body frame at x_j. In the intervening time between t_i and t_j, the landmark follows its motion model, resulting in a transform $T_{k^ik^j}$. Thus, to compare the estimate of the measurement between x_i and x_j, the second measurement T_{jk} must be transformed through the inverse of $T_{k^ik^j}$ to be in the coordinate frame of the robot when at t_i. Removing the contribution of the landmark's motion results in the transform $T_{s_{ij}}$ which can be compared with the apparent transform between the robot's poses based on the measurements.

Constant Translational and Angular Velocity KMM

The transform $T_{k^ik^j}$ representing the motion model can be addressed as a change in position (Δp) and orientation ($\Delta\theta$) over a time step (Δt) for a landmark experiencing constant translational acceleration (a_b) and constant angular velocity (ω_b) measured in its body frame. This is given by Sittel, Muller and Burgard [82] as:

$$\Delta p = v_b \Delta t + 0.5(a_b + \omega_b \times v_b)\Delta t^2, \tag{5.2}$$

and

$$\Delta\theta_b = \omega_b \Delta t. \tag{5.3}$$

To transform this from body frame to the world frame for the transform $T_{k^ik^j}$, these expressions must be pre-multiplied with by the initial orientation of the body, R_b.

With this structure established, the next step is to estimate the landmark KMM parameters, the landmark velocity, using one of two proposed approaches: integrate the motion model into the graph optimization as a latent variable (LV) to be estimated simultaneously with the robot's poses (5.1.1), or to use expectation maximization (EM) to approximate the parameter's value (5.1.2). This chapter will explore the expected computational advantage of the EM method compared with the expected performance advantage of the LV method to assess their applicability.

5.1.1 Latent Variable Approach

An LV is one that is not directly observed, but is instead inferred. In the case of the landmark KMM, the parameters of the model (the translational velocity v_b, and angular velocity ω_b) are treated as LVs and inferred from the graph. Building these factors requires the time step Δt between poses, the initial rotation of the landmark R_b, and the two observations (pose and covariance) of the landmark as inputs. The Jacobian for this factor is determined from the numerical derivative function in the gtsam library.

This above method allows the evolution of the landmarks in 6-DOF (translational velocity in X, Y, Z, and angular velocities in roll, pitch and yaw). In the case of 2-D landmarks that exhibit primarily translational velocities in the (X-Y) plane and angular yaw rate, a prior factor can be placed over the velocity variables (v and ω) that strictly constrain the standard deviation of the Z, roll and pitch elements, while more loosely constraining the X, Y and yaw elements (the prior used in these experiments is given in Table 5.1).

5.1.2 Expectation Maximization Approach

Adding LVs to estimate the model parameters in the graph increases the graph dimension, and with it, the time to determine the optimal variable values. In 1998, Neal and Hinton defined an approximation approach, EM, to iteratively solve for the pose and landmark state estimates [83]. While Neal and Hinton fist used this to estimate the landmark positions in a static map, Graham and How later used this approach for

Robust SLAM in 2014 to iteratively estimate the poses and covariance matrix [84].

EM is applied to solve for the landmark KMM parameters (ie. velocity). The KMM parameters are estimated from observations and unoptimized poses in the expectation-step (E-step) then those values are used to form the loop closure factors and apply standard iSAM optimization [9] in the maximization-step (M-step). The EM algorithm, as proposed by Neal and Hinton [83], iterates between the E- and M-steps until the estimates of KMM parameters and poses convergence. Due to the high computational cost of each M-step, Graham and How [84] proposed a single-iteration approach. Their logic assumed that since the trajectory has many poses, the M-step (iSAM optimization) will be repeated many times over the mission. So rather than iterating until the estimate converges each time a new pose is added, the single-iteration approach can iterate only once when a new pose is added, and delay further iterations to subsequent pose additions, eventually driving the estimate towards the true value.

Weighted Least Squares Linear Regression

Linear regression is applied to estimate the landmark's KMM's parameters in the E-step. Following the constant velocity model proposed in Chapter 3, the slope (velocity) and intercept (initial position) for each landmark can be computed using ordinary least squares. However, since the confidence of each observation may vary, weighted least squares (WLS) is preferred. This is performed for each of the landmark's motion directions. In a general system this would be translational velocities in X, Y and Z and angular velocities in roll, pitch and yaw, while in a planar landmark case, this reduces to translational velocities in X and Y and angular rate in yaw (ψ).

Given a generic system $y = mx + b$ and n observations of the form (x_i, y_i) with standard deviation σ_i, the WLS estimate for m and b are:

$$\hat{m} = \frac{\sum w_i(x_i - \bar{x}_w)(y_i - \bar{y}_w)}{\sum w_i(x_i - \bar{x}_w)^2}, \tag{5.4}$$

and

$$\hat{b} = \bar{y}_w - \hat{m}\bar{x}_w, \tag{5.5}$$

where the $\bar{x}_w$ and $\bar{y}_w$ are the weighted means given as:

$$\bar{x}_w = \frac{\sum w_i x_i}{\sum w_i}, \tag{5.6}$$

and

$$\bar{y}_w = \frac{\sum w_i y_i}{\sum w_i}. \tag{5.7}$$

In the case of velocity estimation, m is the estimated velocity, b is an estimate of the position at $t = 0$, x is the observation time and y denotes the observed landmark position. The variances of each observation are inversely related to the weights used in the estimation:

$$w_i = \frac{1}{\sigma_i^2}. \tag{5.8}$$

Finally, the variance of the estimated velocity is:

$$Var(\hat{m}) = \frac{\sigma^2}{\sum w_i (x_i - \bar{x}_w)^2}. \tag{5.9}$$

The estimate of the landmark's velocity and variance for the E-step can be used to calculate the transform $T_{k^i k^j}$ and subsequently h_s, for the loop closure factors used to construct the graph for the M-step. While the LV approach adjusts the velocity estimates in the graph along-side the pose estimates, the EM method treats the velocity estimate outside of the graph, only updating the loop-closure factors with the first iteration of the velocity estimate calculated each time the landmark is observed.

Implications for Estimating with Few Observations

In implementing EM for parameter estimation, it is interesting to look at the limiting case of only a few observations. If a landmark has only been observed once, its velocity cannot be estimated and a default value must be used. The assumption of a static landmark is tied to the expectation that most landmarks will be static, quasi-static or slow in this application. If a landmark has been observed exactly two times, the estimate of the velocity is calculated by attributing all changes in a landmark's measured position to its velocity. This means that the uncertainty in the vehicle's pose and the landmark velocity cannot be distinguished. Only when there are several measurements, and multiple iterations of the EM cycle, can the uncertainty in the landmark's motion and the uncertainty in the quadrotor's position be separated, yielding the mapping and localization accuracy benefits of MBD-SLAM over traditional static SLAM.

Computational Implications of EM Approach

The EM approach to estimate the landmark KMM parameters has the advantage of reduced computational complexity, as the graph size is not scaled by the number of landmarks. However, at each SLAM update, it requires re-calculating all loop closure factors for each landmark that is observed. Since so many factors are changed at each iteration, especially on the return trip of the mission, the computational gains of incremental optimizers, like iSAM, over full-graph optimizers are diminished.

5.2 Proposed Model-Based Dynamic SLAM Algorithm

Two approaches to MBD-SLAM were proposed. In both approaches, a quadrotor uses [1]AprilTag-abstracted landmarks [28] to perform pose-graph SLAM based on the [2]iSAM optimizer [9], while following waypoints using a velocity controller. LV-MBD-SLAM, as defined in Section 5.1.1, is shown in Figure 5.3. Meanwhile Figure 5.4 shows EM-MBD-SLAM, which uses an approximation as defined in Section 5.1.2 to reduce the graph size, and consequently the computational load.

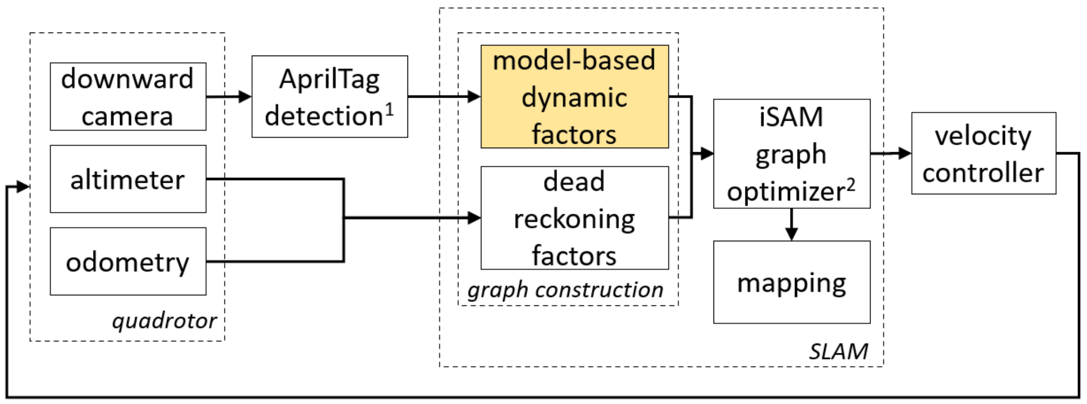


Figure 5.3: Proposed LV implementation of the MBD-SLAM algorithm. The yellow block indicates the novel MBDFs that allow the graph to estimate the landmark velocities.

Both proposed MBD-SLAM approaches learn the landmark velocities by allowing the factor graph to adapt to the spatio-temporally evolving environment. As with the baseline static SLAM in Figure 4.11, a quadrotor with navigation sensors that report the altitude and odometry and a downward facing payload camera, uses landmarks abstracted with the AprilTag fiducial tagging system to perform SLAM. The SLAM

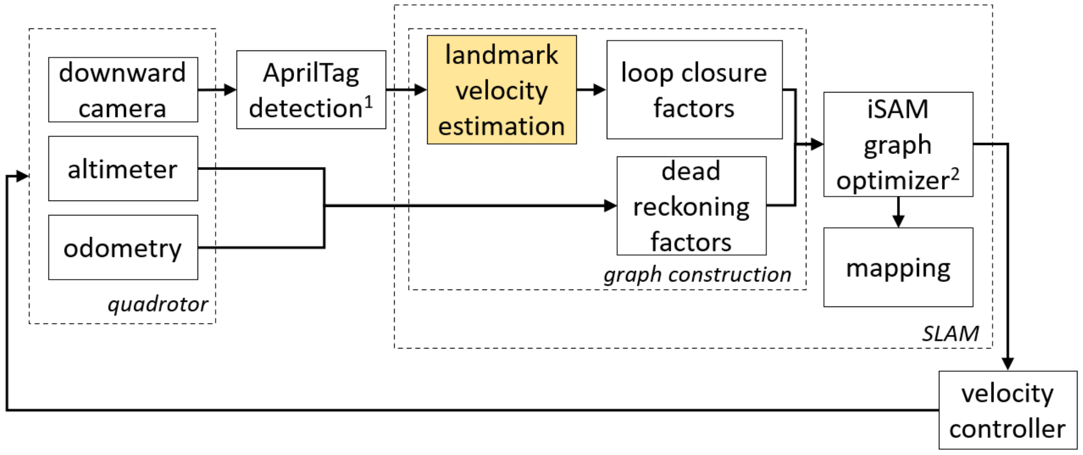


Figure 5.4: Proposed EM approximation for the MBD-SLAM algorithm. The yellow block indicates the proposed landmark velocity estimation step of the EM algorithm. Figure adapted from [81].

pose estimate for the quadrotor is used by the velocity controller to provide the inputs necessary to follow the set waypoints. However, for MBD-SLAM, the standard static loop closure factors are replaced (MBDFs for the LV approach) or adjusted (using the landmark velocity estimation from the EM approach). In Figures 5.3 and 5.4, these new blocks are filled in yellow.

With the LV approach, the observations of landmarks are directly used to build the MBDFs. At each iteration, the number of variables (nodes) in the graph grows by one for the new pose and one for each previously unseen landmark's velocity. The number of factors also grows by one for the new dead-reckoning factor and by the number of loop closures that are formed.

With the EM approach, all observations of each landmark are first used to calculate the landmark velocity using WLS linear regression, before that velocity estimate is used to adjust the loop closure factor by the transform $T_{k^ik^j}$. In the EM approach, the graph grows by only one variable (the new pose), one dead-reckoning factor and the number of loop closure factors, at every iteration. This means the LV-approach's graph grows more quickly, especially when more landmarks exist in the environment. Larger graphs translate directly to longer computation times due to the computationally expensive graph optimization step [20].

Although this Chapter focuses on landmarks that evolve according to a KMM, the WLS regression and parameter estimation approach can be modified to estimate the parameters of any motion model that is meaningful to the landmark evolution. For example, this method can be extended to other types of evolving landmarks such as those with changing size, shape, or reflectivity. Further, either estimation approach for the MBD-SLAM algorithm can be used with different sensors and platforms, and for various missions and environments, so long as an odometry and landmark pose measurement is available and an appropriate landmark KMM can be selected.

5.3 Results

To understand the performance of the MBD-SLAM proposed in this chapter, three cases are performed using the two velocity estimation approaches: LV and EM. All cases assume the landmarks move in a 2D plane, thus, their velocities are only estimated for X, Y and ψ. The UAV's task in all cases is to take-off from the start position at $(x, y, \psi) = (0, 0, 0)$, travel to the right (positive X-direction) until reaching a waypoint then return to the start position. The mission and environment parameters of the cases are established in Table 4.1. The three cases are as follows:

- **Case 1:** one-pass, full-scale, slow environment, simulation (Section 5.3.1)
  Case 1 is the parent case for the MBD-SLAM, and in addition to demonstrating the successful performance in terms of the metrics established in Section 4.4.3, benchmarks for the execution time of one SLAM iteration are compared in Case 1 to assess if the theoretical computation improvement of the EM approach over the LV approach is achieved.

- **Case 2:** one-pass, model-scale, slow environment, simulation and experiment (5.3.2)
  Case 2 allows comparison between simulated and experimental results, as well as comparison with the results of applying static SLAM from Section 4.5 to demonstrate the improvement when the static assumption is relaxed through MBDFs.

- **Case 3:** one-pass, full-scale, quasi-static environment, simulation (Section 5.3.3)

> The improvement shown in Case 2 provides justification for proceeding with this approach to the limiting case of quasi-static environments, however, Case 3 shows the MBDF's limitations in quasi-static environment, and motivates the work of the next chapter.

The covariance matrices used in MBD-SLAM match the static SLAM covariance matrices in Table 4.7. The only exception is the yaw term of the dead-reckoning factor which, when reduced by an order of magnitude (from 0.03 rad to 0.003 rad), resulted in improved performance in velocity estimation accuracy in the full-scale tests. This change did not produce a noticeable effect in the model-scale tests, and is thus ascribed to the exponentially larger cross-track uncertainty associated with a yaw variance over 100 m compared to 10 m. This is used in all tests in Chapters 5, 6, 7 and 8.

Further, the LV approach requires a prior for the velocity of each landmark to restrict the velocity estimates to planar (X, Y, ψ) motion only. This is done by placing a tight variance on elements that are not estimated (Z, roll and pitch) and a loose variance on elements that are estimated (X, Y, ψ). The standard deviations on the diagonal of that covariance matrix are given in Table 5.1. The system sensitivity is such that specifying the order of magnitude of the standard deviation is sufficient.

Table 5.1: LV MBDF covariance matrix prior described by the standard deviations on the diagonal for the model-scale. Suitable for planar landmark motion in model-scale, slow environment.

LV velocity prior		**standard deviations**
roll	[rad/s]	0.0001
pitch	[rad/s]	0.0001
yaw	[rad/s]	0.1
X	[m/s]	0.01
Y	[m/s]	0.01
Z	[m/s]	0.001

5.3.1 Case 1: Full-Scale Simulation

The performance of MBD-SLAM using the two approaches – LV and EM – is shown below. First, the computational effort of the two methods are compared to determine whether the EM-MBD-SLAM is in fact more efficient. Next, the localization, mapping and velocity estimation test results are summarized in Table 5.2 and the localization and mapping performance of the median-performance trial is shown in Figure 5.6 for LV-MBD-SLAM, and Figure 5.7 for EM-MBD-SLAM.

Case 1 Computational Effort Result

An important consideration for autonomous systems is the ability to perform the algorithm on-board in near real-time. For that reason, the EM approximation was proposed for MBDFs. The execution time of the two approaches is compared in Figure 5.5 to assess whether that theoretical reduction in complexity resulted in reduced computational effort.

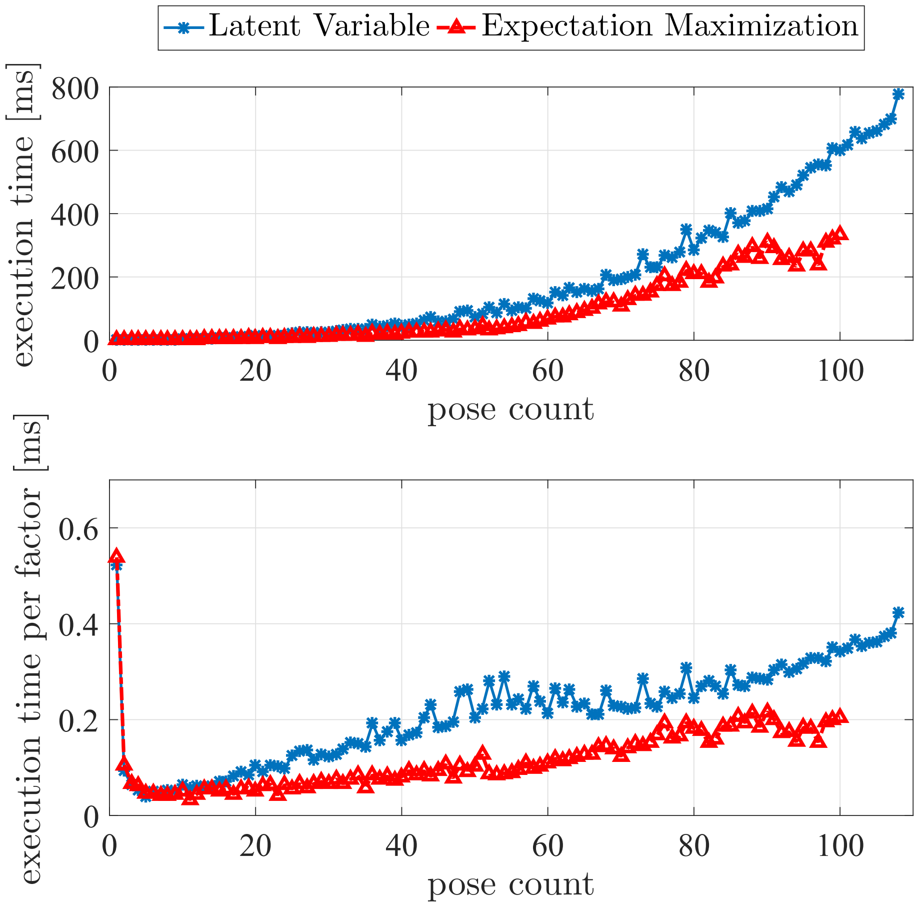
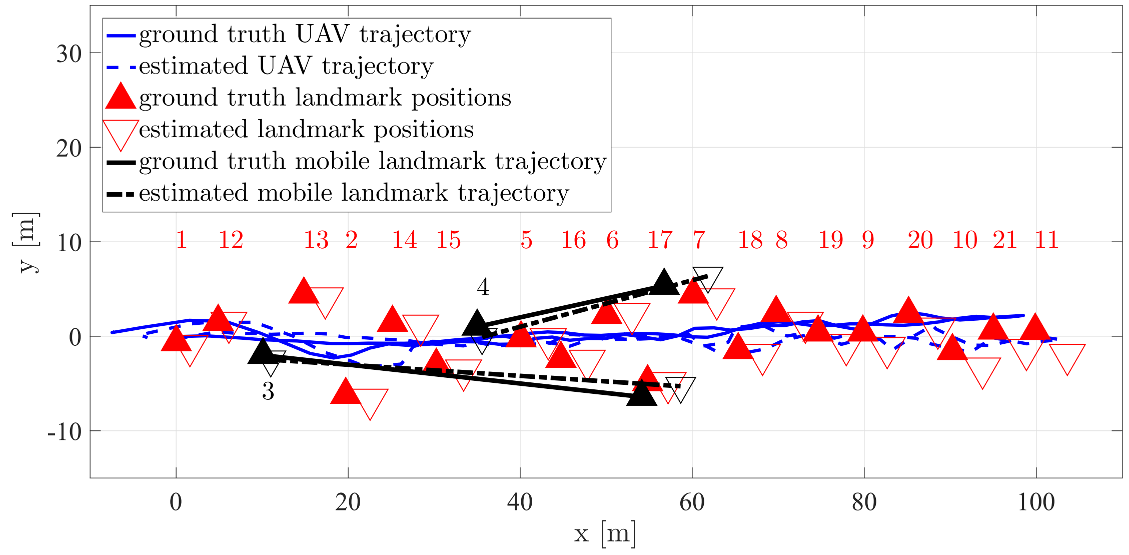


Figure 5.5: Comparison of execution times for one cycle of LV- and EM-MBD-SLAM.

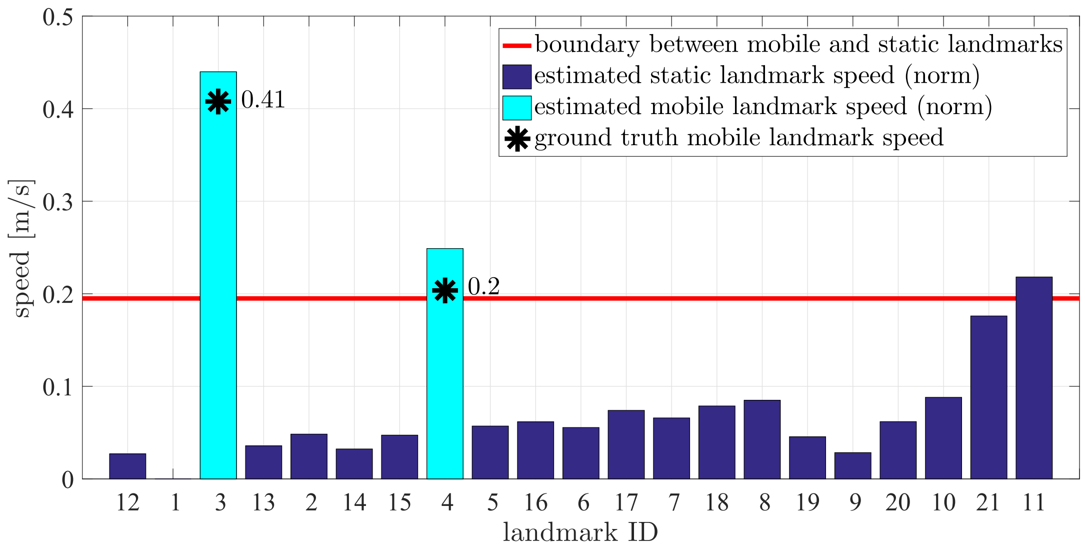
The execution time is calculated for one cycle which starts when an image is received via the AprilTag detector. In most cases, two landmarks are viewed in one image, but there may be no landmarks or several landmarks in an image depending on the configuration of the moving landmarks and the quadrotor's pose. The cycle ends when the new pose estimate is published to the quadrotor's velocity controller. The maximum value of the execution time for one cycle and the maximum execution time per factor in the graph are reported.

While the execution time grows with the graph size for both cases, because the LV approach has more variables to estimate, it is more computationally expensive to optimize. In this case, only 20 landmark velocities (each in 3-DOF) are estimated (recall, 21 landmarks, but landmark 1 is assigned to be static while all others are treated as potentially moving) while over 100 poses (each in 6-DOF) are being estimated so the impact is more evident later in the mission when many factors use those velocities. Note that the exact number of poses is different in each trial due to the slightly different paths taken by the UAV in response to the pose estimates. For example, if the pose is underestimated as the UAV approaches a waypoint, the UAV will continue moving until it is estimated to be within a tolerance distance of the waypoint thereby requiring one or more extra poses to complete the pass. This difference does not change the general trend of the execution time per cycle of EM or LV.

For the execution time per factor in the graph, the LV approach initially grows more quickly during the first half of the mission, then stabilizes before growing again. That point of stability corresponds with the quadrotor hovering near $x = 100$ m prior to returning to the start point on a reciprocal heading on the same path. This is an expected behaviour as at that position it sees one landmark (AprilTag 11) and adds new odometry factors and several loop closure factors through AprilTag 11 without having a significant impact on the rest of the graph. At this moment, the graph grows quickly in number of factors, with a relatively small cost to optimize the incremental change in the graph. Meanwhile, the execution time per factor for the EM approach is approximately linear, as expected for iSAM [9].



(a) planar map

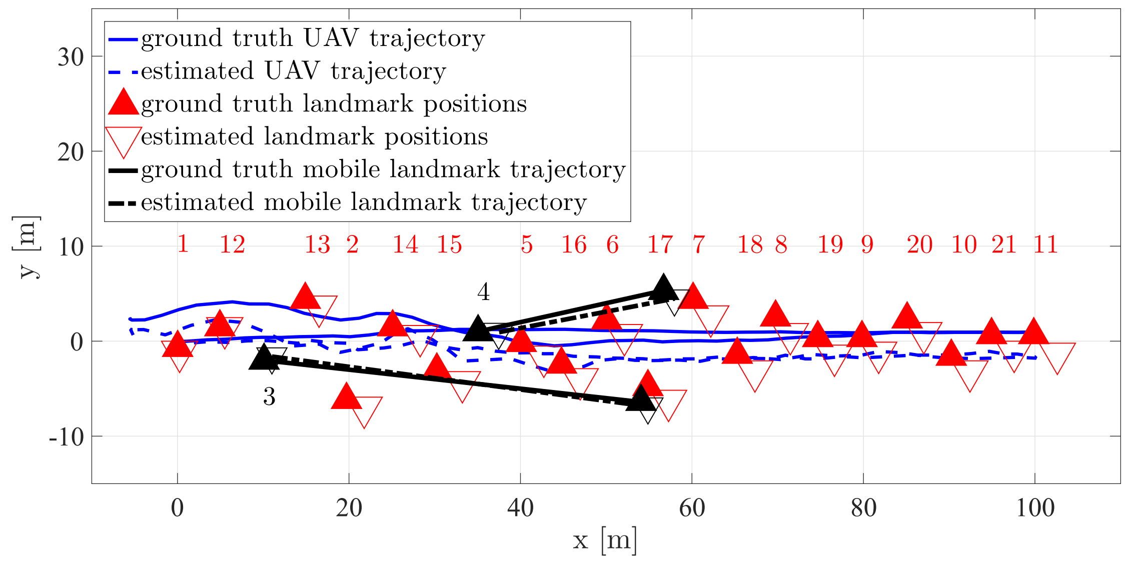


(b) speed estimates

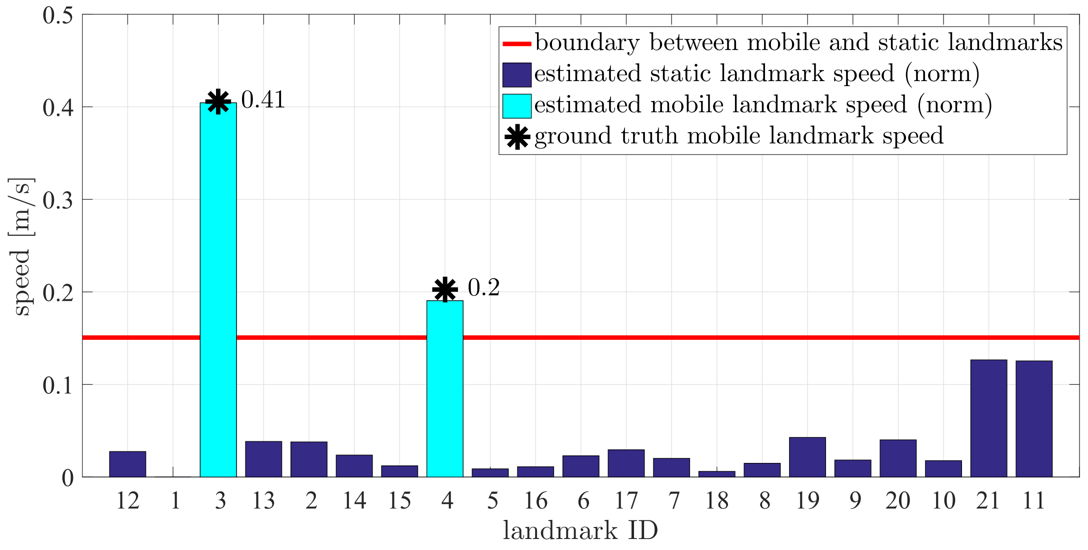
Figure 5.6: Case 1 simulation of LV-MBD-SLAM in full-scale, slow environment.

Case 1 Localization and Mapping Results

Figure 5.6a shows the median-performance trial of the one-pass, full-scale, slow environment LV-MBD-SLAM as a map. Here, the quadrotor's trajectory in the X-direction is overestimated resulting in the landmark positions also being over-estimated. The cross-track error on both the trajectory and the map grows as the quadrotor travels further from the start location. The velocities are mostly well estimated as shown in Figure 5.6b, with landmarks far from the take-off location, where the localization uncertainty is highest, having the worst velocity estimation accuracy. The mean speed boundary between the static and mobile landmarks is 0.19 m/s which is just lower than the defined slow speed (0.2 m/s), which means the LV-MBD-SLAM might struggle to distinguish slowly moving landmarks from the static landmarks in some trials.



(a) planar map



(b) speed estimates

Figure 5.7: Case 1 simulation of EM-MBD-SLAM in full-scale, slow environment.

Table 5.2: Case 1 (full-scale, slow) simulation performance summary for LV- (Figure 5.6) and EM- (Figure 5.7) MBD-SLAM.

metric	**units**	**LV**	**EM**
localization error (norm)	[m]	4.09 ± 1.12	4.22 ± 1.78
x error	[m]	2.40 ± 1.01	1.96 ± 1.46
y error	[m]	1.52 ± 0.93	1.19 ± 0.74
z error	[m]	2.25 ± 0.99	2.75 ± 1.41
mapping error (norm)	[m]	3.87 ± 1.23	4.50 ± 2.77
x error	[m]	1.78 ± 1.09	2.22 ± 1.74
y error	[m]	1.48 ± 1.29	1.21 ± 0.68
z error	[m]	2.41 ± 1.09	3.22 ± 2.19
mean static landmark speed error	[m/s]	0.0637 ± 0.0177	0.0581 ± 0.0211
mean dynamic landmark speed error	[m/s]	0.0444 ± 0.0218	0.0286 ± 0.0187
mean speed boundary	[m/s]	0.1878 ± 0.0235	0.1720 ± 0.0233
maximum execution time	[ms]	779.23	332.10
maximum execution time per factor	[ms]	0.4226	0.2162

The median performance case of the one-pass, full-scale, slow environment EM-MBD-SLAM algorithm map in Figure 5.7a shows an overestimation of the along-track (X) positions of the landmarks, as well as a notable cross-track error on the trajectory. The speeds for the landmarks in Figure 5.7b are well estimated. Not surprisingly, landmarks far from the start location have the highest error, though the estimated speeds (approximately 0.13 m/s in the median-performance trial) remain below the mean speed boundary (0.17 m/s) for the 10 trials. This mean speed boundary is between the defined slow speed (0.2 m/s) and the quasi-static speed (0.1 m/s) which indicates the EM-MBD-SLAM approach would be able to distinguish slow landmarks, but not quasi-static landmarks, from the static environment.

Case 1 Discussion

The performance of both the LV and EM approaches show dramatically improved localization and mapping compared to applying static SLAM in the dynamic environment. Not only are the moving landmarks well estimated, the trajectory and map of static landmarks match the ground truth better.

The mapping and localization performance of these two cases appear quite similar when comparing Figures 5.6 and 5.7, and the localization and mapping errors are statistically similar. Comparing the localization and mapping errors to the path length of 100 m, the localization error is between 4-5% for both cases and between 3-5% for the mapping error. Much of these errors are in the Z-direction which matches the modelled noise of the altimeter. This performance aligns with the static SLAM performance (Table 4.8) in the static environment.

The spread of mapping and localization performance (the standard deviations) for the EM approach is larger than the LV approach for most measures. Due to the use of a single-iteration of linear regression in the EM approach, the uncertainty on a given measurement has a larger impact on the estimated velocity (and subsequently the mapping and localization estimates) than if that uncertain measurement were optimized (until convergence) in the graph. This means the performance variation over 10 trials is expected (and observed) to be larger for the EM approach than the LV approach.

The speed estimates of the landmarks shown in Figures 5.6b and 5.7b are statistically similar based on their standard deviations for the two cases, though the LV approach tended to overestimate the speed of static landmarks slightly more than the EM approach. The threshold to distinguish between moving and static landmarks was similar for the two approaches (approximately 0.18 m/s, equivalent to 36 km/day), as was the mean static speed estimate (approximately 0.06 m/s, equivalent to 12 km/day).

The most notable difference between the two approaches is the execution time shown in Figure 5.5. The EM-MBD-SLAM does provide a clear improvement in execution time for each cycle as well as execution time per factor particularly as the number of poses grows. Note that MBD-SLAM does not utilize node or edge removal techniques that could bound the graph execution time. Given the growth rates of the execution time for either approach, it is clear that if the mission were longer with the same SLAM iteration rate, node or edge removal would be necessary to maintain an acceptable pose update rate for the UAV waypoint-tracking controller.

5.3.2 Case 2: Model-Scale Simulation and Experiment

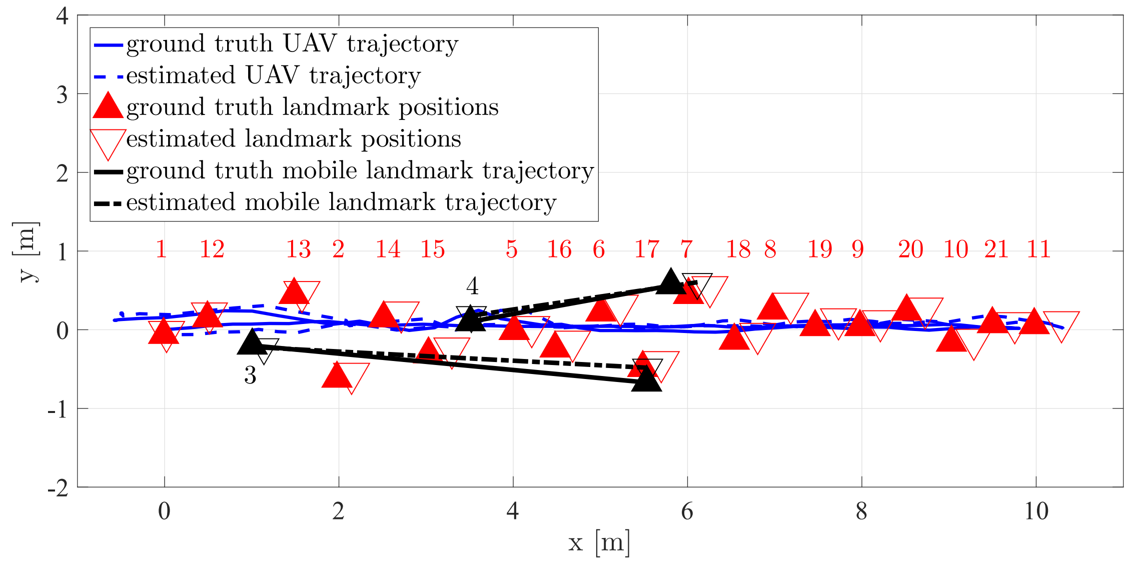
The second case demonstrates the performance of MBD-SLAM in the model-scale, slow environment in a simulation and an experiment. The LV approach results are summarized in Table 5.3 with the median-performance simulation trial shown in Figure 5.8 and experimental trial is shown in in Figure 5.9. The EM approach results are summarized in Table 5.4 with the median-performance simulation trial shown in Figure 5.10 and experimental trial shown in Figure 5.11. These four tests allow comparison of MBD-SLAM with static SLAM applied to slow environments (Section 4.5) as well as ensuring MBD-SLAM does not perform more poorly in experiments than in simulations.

Case 2 LV-MBD-SLAM Results

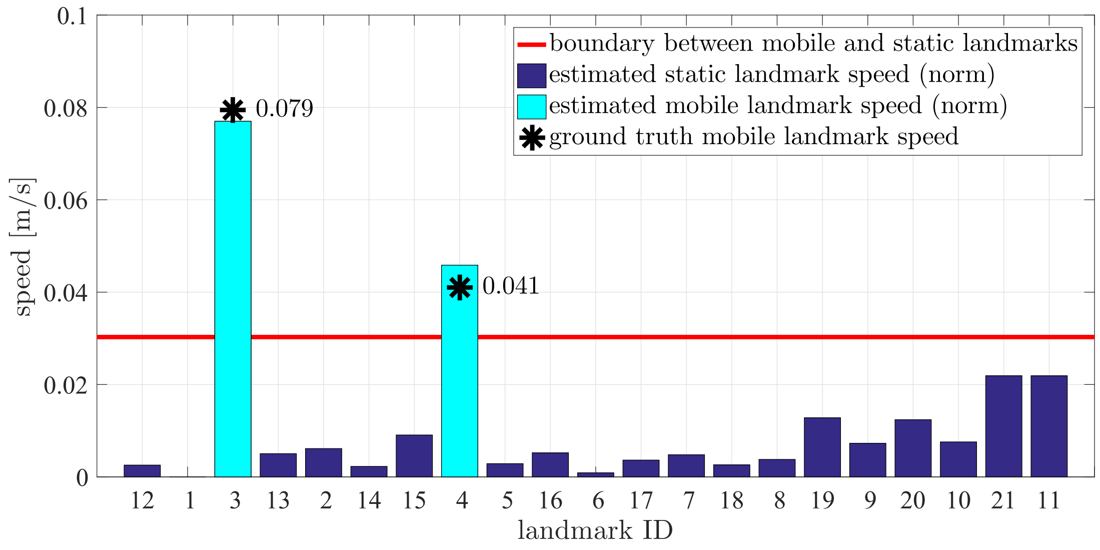
Table 5.3: Case 2 (model-scale, slow) simulation (Figure 5.8) and experiment (Figure 5.9) performance summary for LV-MBD-SLAM.

metric	**units**	**simulation**	**experiment**
localization error (norm)	[m]	0.34 ± 0.07	0.37 ± 0.11
x error	[m]	0.21 ± 0.07	0.20 ± 0.07
y error	[m]	0.11 ± 0.05	0.17 ± 0.08
z error	[m]	0.18 ± 0.09	0.19 ± 0.08
mapping error (norm)	[m]	0.34 ± 0.07	0.28 ± 0.09
x error	[m]	0.22 ± 0.05	0.13 ± 0.05
y error	[m]	0.11 ± 0.05	0.12 ± 0.04
z error	[m]	0.20 ± 0.11	0.18 ± 0.08
mean static landmark speed error	[m/s]	0.0104 ± 0.0047	0.0060 ± 0.0017
mean dynamic landmark speed error	[m/s]	0.0071 ± 0.0089	0.0101 ± 0.0075
mean speed boundary	[m/s]	0.0332 ± 0.0072	0.0259 ± 0.0045

Figure 5.8a shows the median performance trial of the one-pass, model-scale, slow environment LV-MBD-SLAM simulation as a map. Here, the quadrotor's trajectory in the X-direction is overestimated resulting in the landmark positions also being over-estimated. The cross-track error in both the trajectory and the map are not significant. The velocities are well estimated as shown in Figure 5.8b, with landmarks far from the take-off location, where the localization uncertainty is highest, having



(a) planar map

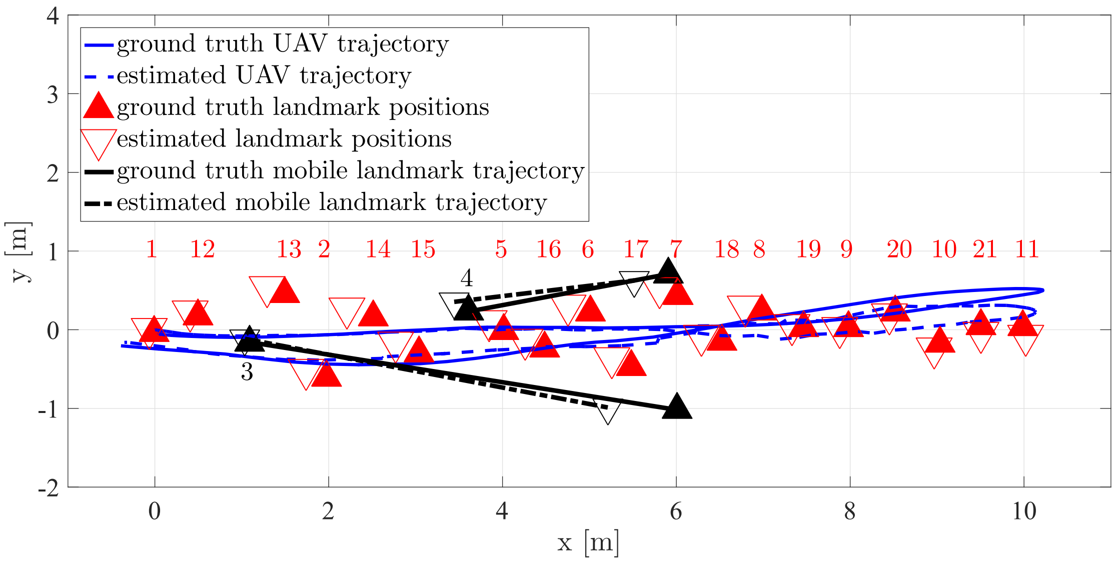


(b) speed estimates

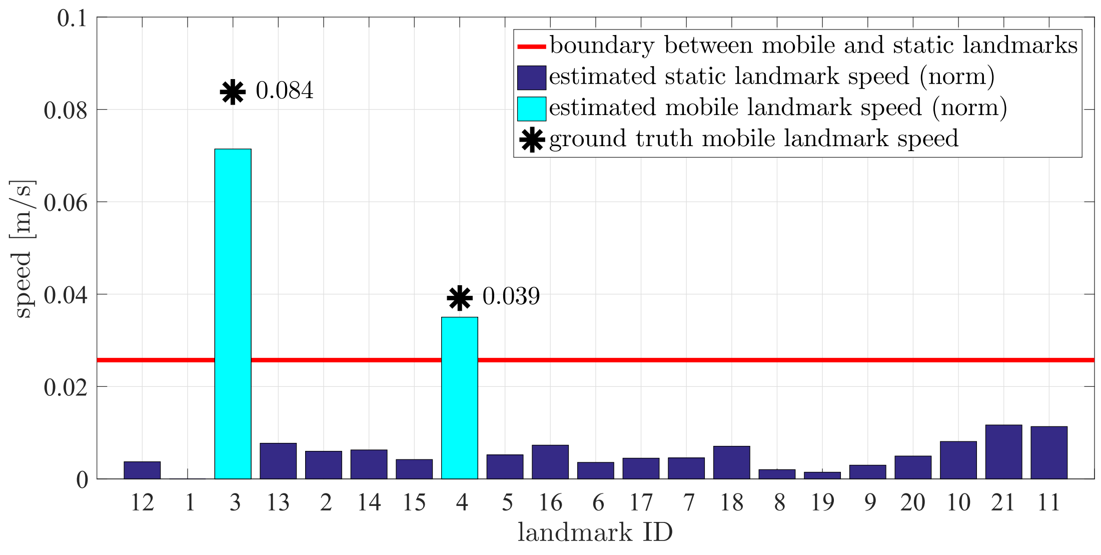
Figure 5.8: Case 2 simulation of LV-MBD-SLAM in model-scale, slow environment.

the worst velocity estimation accuracy. The speed boundary between the static and mobile landmarks is on average 0.033 m/s for the ten trials (see Table 5.3), which is about halfway between the defined slow speed (0.04 m/s) and quasi-static speed (0.02 m/s), which means the LV-MBD-SLAM can distinguish slow landmarks, but not quasi-static landmarks, from static ones.

Figure 5.9a shows the median performance case of the one-pass, model-scale, slow



(a) planar map



(b) speed estimates

Figure 5.9: Case 2 experiment of LV-MBD-SLAM in model-scale, slow environment.

environment LV-MBD-SLAM experiment as a map. Unlike the simulation experiment, the plotted case shows an underestimation of the quadrotor's trajectory in the X-direction along with underestimation of the mobile landmark velocities and a shift towards the origin of the static tag positions. The cross-track drift of the quadrotor is notable in the ground truth trajectory and while it is well estimated for the first 70% of the mission, poses further from the take-off point, where the uncertain attitudes mean pose uncertainty is larger, show higher cross-track error. The velocities of mobile landmarks are under-estimated but are close to their correct values (0.072 m/s estimated compared to 0.084 m/s for the faster mobile landmark) as shown in Figure 5.9b, while static landmark speeds are well estimated. The speed boundary in the experiment is slightly lower than in simulation at 0.026 m/s on average for the 10 trials, meaning again the LV-MBD-SLAM can distinguish slow landmarks, but not quasi-static landmarks, from the static ones.

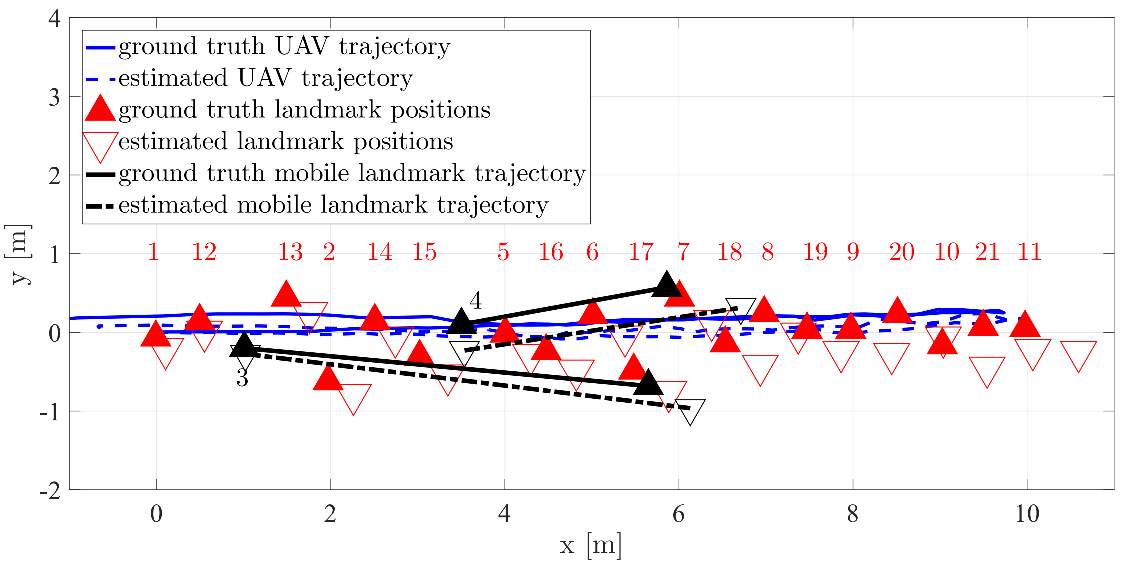
Case 2 LV-MBD-SLAM Discussion

The simulation and experimental performance in terms of mapping and localization accuracy on average showed similar magnitudes and trends between the X, Y and Z components for LV-MBD-SLAM at model-scale. The speed boundary between static and mobile landmarks was better (lower) in the experiment (0.026 m/s) than in the simulation (0.033 m/s) but both are sufficient to discern slow landmarks, but not quasi-static landmarks. Although the median cases that were plotted showed a discrepancy in the cross-track accuracy and under-estimation/over-estimation of the UAV poses in the along-track, on average the cross-track and along-track localization errors were not significantly different across the 10 repeat trials (Table 5.3).

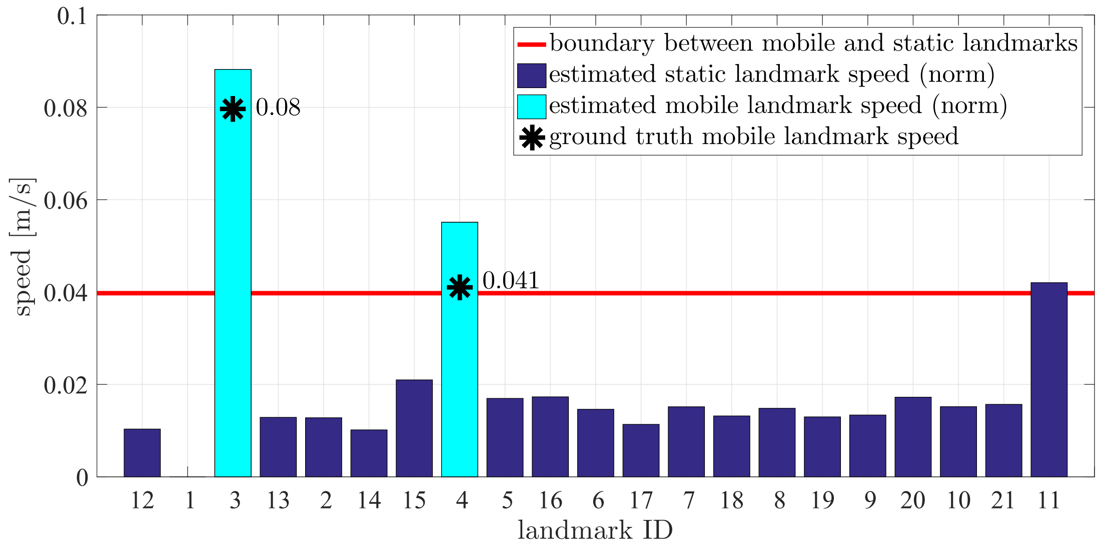
The performance of the LV approach in the model-scale tests (both simulation and experiment) show the same trends as the full-scale test in Case 1. The localization and mapping errors are between 3-4% relative to the 10 m pass length, the same error (as a percentage) as the full-scale. The boundary between static and dynamic landmarks is equivalent to approximately 26-33 km/day, which is somewhat better than the full-scale (36 km/day), but both scales were able to accurately distinguish slow landmarks from static landmarks. In the model-scale, the LV approach is better able to estimate the static landmark speed at an equivalent speed of 6-10 km/day (full-scale at 12 km/day), due to the smaller UAV attitude errors growth over 10 m.

Overall, the proposed LV-MBD-SLAM approach is an effective method to estimate the trajectory and map in a slow environment, as the mapping and localization performance in the slow environment are similar to the baseline static SLAM performance in a static environment. However, speed estimation for static landmarks is less accurate than desired in that several static landmarks would be mis-identified as mobile ones.

Case 2 EM-MBD-SLAM Results



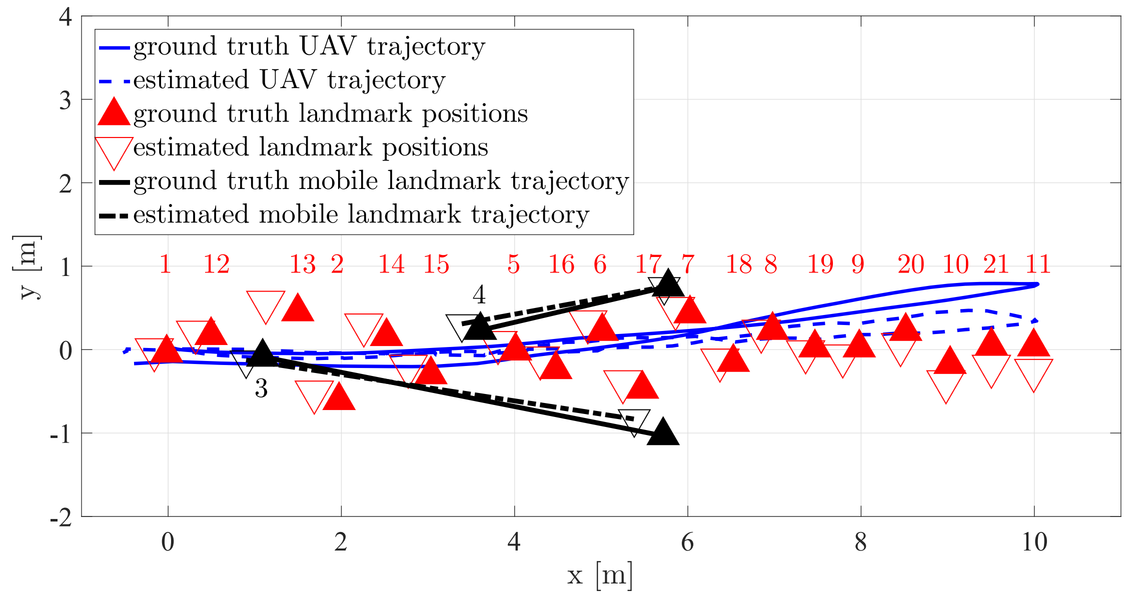
(a) planar map



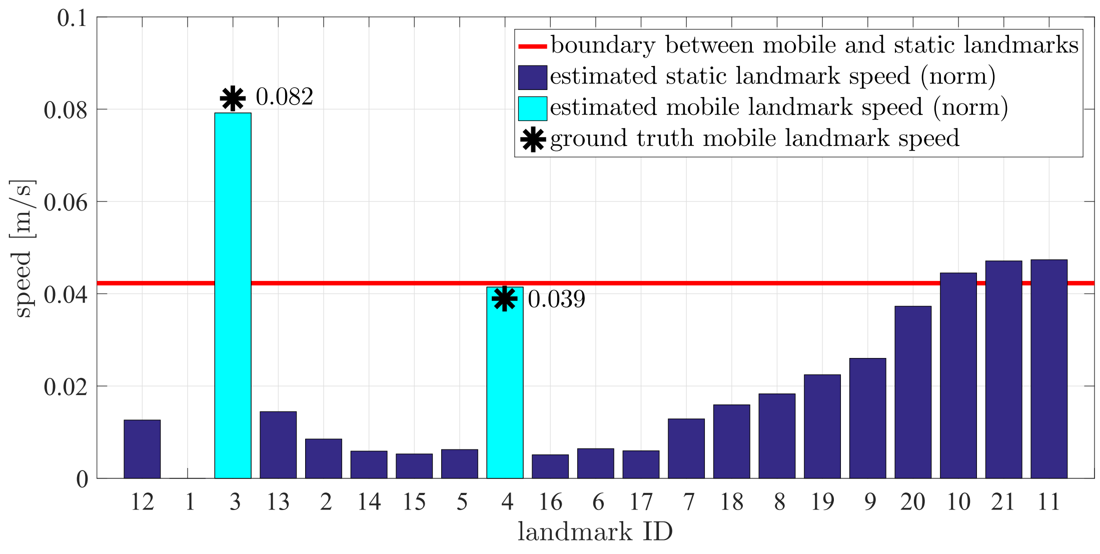
(b) speed estimates

Figure 5.10: Case 2 simulation of EM-MBD-SLAM in model-scale, slow environment.

Figure 5.10a shows the median performance case of the one-pass, model-scale, slow environment EM-MBD-SLAM simulation as a map. As with the median LV-MBD-SLAM simulation of this case, the quadrotor's trajectory in the X-direction is overestimated resulting in the landmark positions also being over-estimated. The cross-track error on both the trajectory and the map are small but notable, especially for the mobile landmarks. The velocities are well estimated as shown in Figure 5.10b.



(a) planar map



(b) speed estimates

Figure 5.11: Case 2 experiment of EM-MBD-SLAM in model-scale, slow environment.

The velocities tended to be over-estimated initially, with more observations driving the velocity estimate closer to the true speeds. The mean speed boundary between the static and mobile landmarks is 0.038 m/s which is close to the defined slow speed (0.04 m/s). This means the EM-MBD-SLAM implementation would be able to distinguish slow landmarks from the static landmarks in most cases.

Figure 5.11a shows the median performance case of the one-pass, model-scale,

Table 5.4: Case 2 (model-scale, slow) simulation (Figure 5.10) and experiment (Figure 5.11) performance summary for EM-MBD-SLAM.

metric	**units**	**simulation**	**experiment**
localization error (norm)	[m]	0.46 ± 0.17	0.35 ± 0.07
x error	[m]	0.20 ± 0.08	0.23 ± 0.04
y error	[m]	0.14 ± 0.05	0.13 ± 0.05
z error	[m]	0.33 ± 0.20	0.14 ± 0.06
mapping error (norm)	[m]	0.54 ± 0.24	0.29 ± 0.06
x error	[m]	0.26 ± 0.16	0.17 ± 0.04
y error	[m]	0.18 ± 0.11	0.11 ± 0.03
z error	[m]	0.35 ± 0.25	0.15 ± 0.07
mean static landmark speed error	[m/s]	0.0126 ± 0.0043	0.0163 ± 0.0046
mean dynamic landmark speed error	[m/s]	0.0061 ± 0.0054	0.0059 ± 0.0048
mean speed boundary	[m/s]	0.0382 ± 0.0066	0.0405 ± 0.0074

slow environment EM-MBD-SLAM experiment as a map. The cross-track drift of the quadrotor is notable in the ground truth trajectory and while it is well estimated for the first 70% of the mission, poses further from the take-off point, where the uncertain attitudes mean pose uncertainty is larger, shows higher cross-track error. The velocities of mobile landmarks are well estimated with an average speed error of 0.006 m/s (6 km/day), as shown in Figure 5.9b, while static landmarks are well estimated with low speeds. The mean speed boundary in the experiment is higher than in the simulation at 0.040 m/s meaning again the EM-MBD-SLAM would struggle to distinguish slow landmarks from static landmarks, despite mapping and localization accuracies similar to the baseline static SLAM case.

Case 2 EM-MBD-SLAM Discussion

The mapping and localization accuracies, summarized in Table 5.4, show similar performance in the X and Y components for EM-MBD-SLAM at model-scale between the simulation and experiment. The Z-component was much larger and more variable in the simulation than in the experiment. This is likely attributed to a day when the altimeter performance in the experiment was more stable than the day the altimeter was characterized. In terms of speed estimation, the velocity estimation for mobile landmarks was similar between the simulation and experiment, however, the static landmark speed estimation was worse (larger speeds) in the experiment (0.0163 ± 0.0046 m/s) than in the simulation (0.0126 ± 0.0043 m/s). This increased the boundary between the static and dynamic landmarks, making discrimination between mobile and static landmarks more difficult in the experiment than in the simulation.

The high boundary between mobile and static landmarks does not occur in the full-scale results for the EM approach (Case 1), nor is it as pronounced in the LV approach at either scale. At model-scale, the WLS regression at the core of the EM-MBD-SLAM approach is challenged both by a small number of observations per landmark, as well as a large cross-track drift of the quadrotor (relative to the distance travelled). Since for small numbers of observations, WLS attributes the localization error of the quadrotor to motion of landmarks (as discussed in Section 5.1.2), it is reasonable to expect the EM approach to be more sensitive to the model-scale's cross-track drift than the LV approach when estimating the KMM parameters (velocity).

As with the LV approach, the localization and mapping accuracies of the EM approach show the same trends in the full-scale experiment in Case 1 and the model-scale experiments in Case 2. The mapping and localization errors relative to the pass length are between 3-5%, with the most significant difference being the smaller Z component error in the model-scale experiment. While the velocity estimation accuracies of the static and mobile landmarks are similar between Case 1 and Case 2, the estimate of the boundary between static and mobile landmarks for the full-scale (34 km/day equivalent) is better than the model-scale (38-40 km/day) as the model-scale would struggle to discern slow landmarks (40 km/day) from static landmarks. Although correctly recognizing whether a landmark is static or mobile is important for navigating a dynamic environment, the localization accuracy is the primary requirement of the MBD-SLAM approach. The localization and mapping accuracy of the EM-MBD-SLAM approach is similar to the baseline static SLAM.

Case 2 Discussion

As in the large scale (Case 1), the performance of the EM-MBD-SLAM and LV-MBD-SLAM is generally similar in the model-scale simulations and experiments.

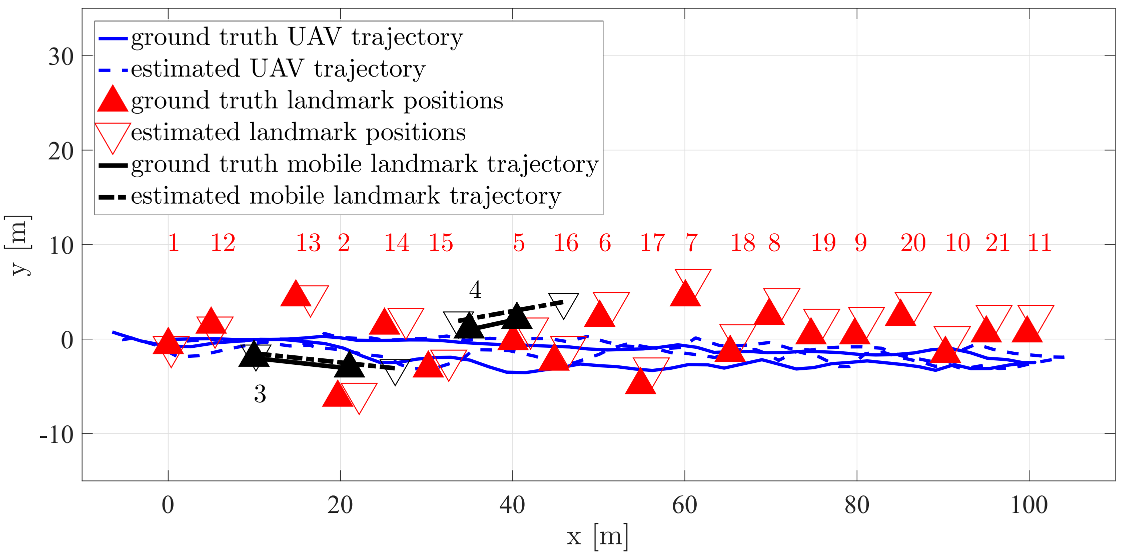
The primary difference in performance is seen in the speed boundary between the mobile and static landmarks for the experiment. In that measure, the LV approach performed significantly better, due the limitation of the EM approach when there are a small number of observations, as previously discussed.

Although the accuracy of the speed estimate is less important to the quadrotor's operation than the localization and mapping accuracy, it is interesting to note that the error on the static landmark speed estimate is higher for the EM approach compared with the LV approach in the model-scale. This is primarily due to the large cross-track speeds estimated for static landmarks numbered 10, 21 and 11 located farthest from the take-off location (x = 9-10 m). While the LV approach re-estimates the velocities of all landmarks at every pose-update iteration, the EM approach only re-estimates the velocities of the landmarks that are observed in that pose-update iteration. This means that when using the EM approach, the velocity estimates of AprilTags 10, 21 and 11 are not re-estimated in response to the loop closure factors constructed as the quadrotor returns to the start position. Since those loop closures are critical to reducing the dead-reckoning error that accrues over time, there is more uncertainty in estimating velocities of landmarks far from the start position using EM-MBD-SLAM approach. Additional observations of these landmarks during additional passes through the environment would trigger updating their velocity estimates, thus improving the velocity estimate's accuracy.

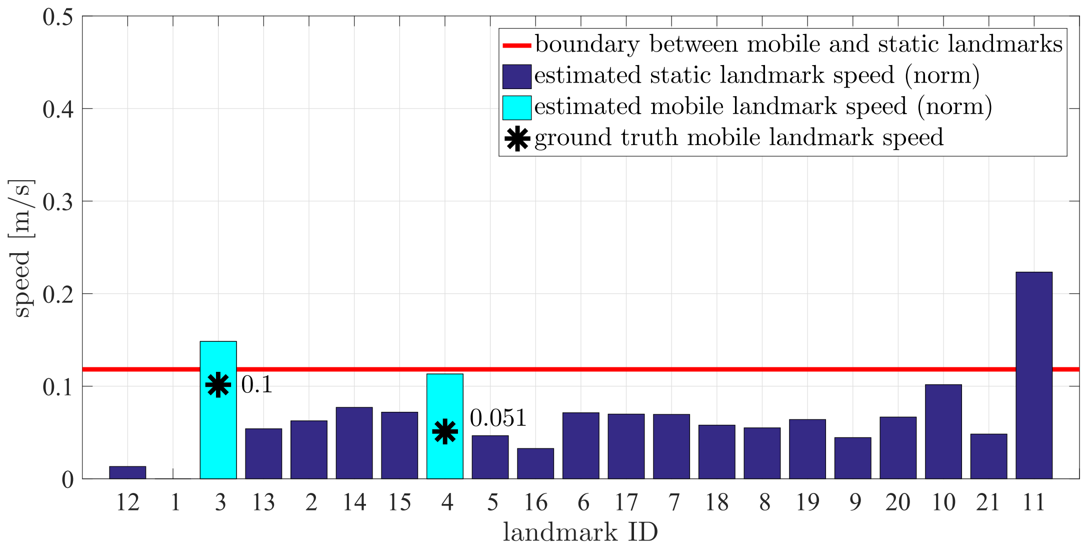
5.3.3 Case 3: Full-Scale Simulation with Quasi-Static Landmarks

The third and final case challenges the MBD-SLAM performance with a quasi-static environment. Cases 1 and 2 showed that the EM-MBD-SLAM and LV-MBD-SLAM were able to discern slow landmarks from static landmarks, but the boundary between static and mobile landmarks was above the quasi-static threshold. This case implements the MBD-SLAM in a quasi-static environment for the full-scale, one-pass experiment to quantify the performance in that challenging environment. The test results are summarized in Table 5.5 and the localization and mapping performance of the median-performance trial is shown in Figure 5.12 for LV-MBD-SLAM, and Figure 5.13 for EM-MBD-SLAM. This case shows that MBD-SLAM continues to perform well in terms of localization and mapping accuracy, but struggles to estimate quasi-static landmark speeds.

Case 3 Localization and Mapping Results



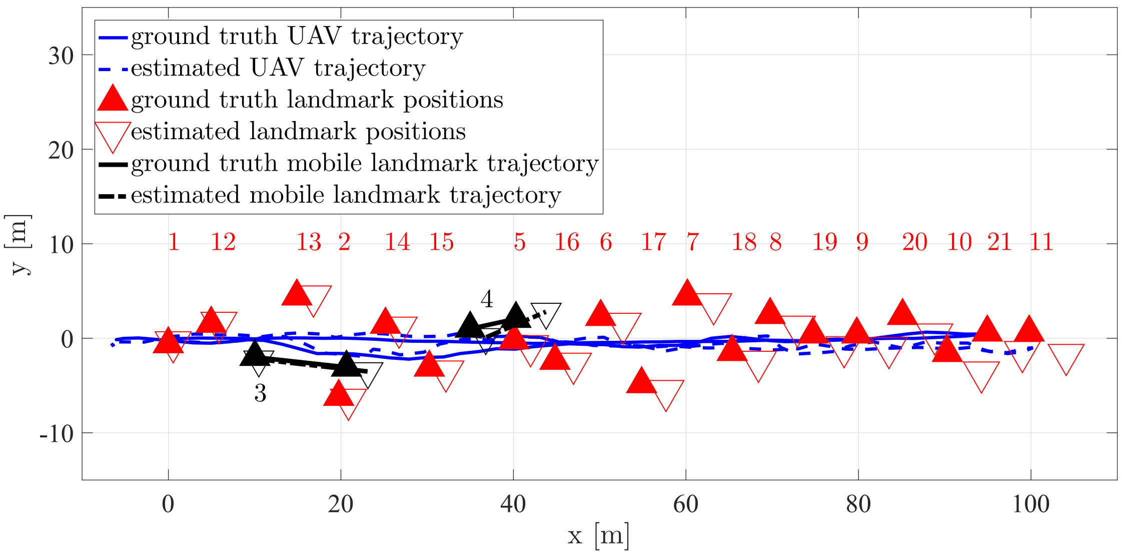
(a) planar map



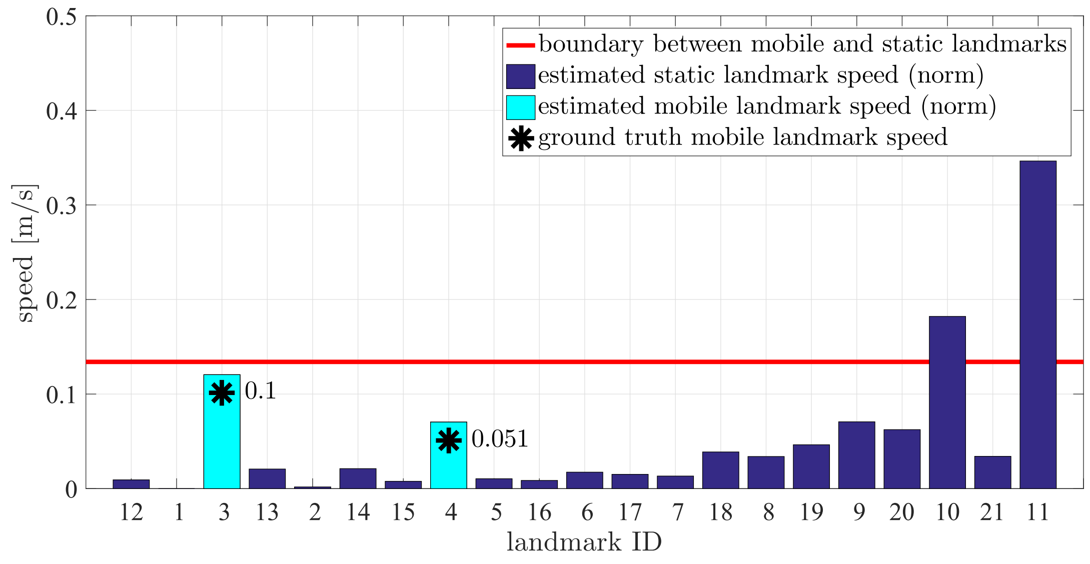
(b) speed estimates

Figure 5.12: Case 3 simulation of LV-MBD-SLAM in full-scale, quasi-static environment.

The map of the median-performance trial of the LV-MBD-SLAM approach applied to the one-pass, full-scale quasi-static environment is shown in Figure 5.12a. As in Case 1 (slow environment), the quadrotor's trajectory and landmark positions are



(a) planar map



(b) speed estimates

Figure 5.13: Case 3 simulation of EM-MBD-SLAM in full-scale, quasi-static environment.

over-estimated in the X-direction. The velocity estimates of the tags are also over-estimated as shown in Figure 5.12b. This is both for the mobile landmarks as well as for the estimate of the static landmarks' speed. The boundary between the static and mobile landmarks is on average 25 km/day, which is much lower than Case 1 (36 km/day), but still unable to discern the quasi-static landmarks from the static ones.

Table 5.5: Case 3 (full-scale, quasi-static) simulation performance summary for LV- (Figure 5.12) and EM- (Figure 5.13) MBD-SLAM.

metric	**units**	**LV**	**EM**
localization error (norm)	[m]	4.56 ± 1.18	4.21 ± 2.76
x error	[m]	2.45 ± 1.02	2.77 ± 2.37
y error	[m]	1.51 ± 1.00	1.00 ± 0.55
z error	[m]	2.70 ± 1.54	2.11 ± 1.30
mapping error (norm)	[m]	4.39 ± 1.14	3.71 ± 1.73
x error	[m]	2.14 ± 0.91	2.27 ± 2.01
y error	[m]	1.22 ± 0.85	0.81 ± 0.38
z error	[m]	3.00 ± 1.73	2.22 ± 1.13
mean static landmark speed error	[m/s]	0.0688 ± 0.0192	0.0683 ± 0.0276
mean dynamic landmark speed error	[m/s]	0.0400 ± 0.0169	0.0351 ± 0.0309
mean speed boundary	[m/s]	0.1265 ± 0.0301	0.1243 ± 0.0484

Figure 5.13a shows the performance of EM-MBD-SLAM for the one-pass, full-scale, quasi-static environment case as a map, while the speed estimates are shown in Figure 5.13b. As in the slow-environment, the trajectory in the X-direction and the landmark speed estimates are over-estimated. While the localization and mapping accuracies are similar (approximately 3-4% error) as are the static and mobile landmark speed errors, the boundary between the static and mobile landmark speeds is on average 25 km/day for the quasi-static environment, compared with 34 km/day for the slow environment. The pattern of estimation accuracy of landmark speed deteriorating for landmarks farther from the start position is seen again in the quasi-static environment.

Case 3 Discussion

The overall performance of the MBD-SLAM algorithm using both the EM and LV approaches in the quasi-static environment show similar performance as in the slow environment. In particular, despite neither method being particularly designed for the quasi-static landmark speeds, the localization and mapping errors are similar (3-5% of the 100 m pass length) not only to the slow environment in Case 1, but also to the baseline static SLAM performance shown in Section 4.4.4. The speed boundary was lower for the quasi-static environment compared to the slow environment, indicating that the MBD-SLAM algorithm was trending towards better estimating the quasi-static speeds and static speed, however, the boundary (25 km/day) remained above the threshold for quasi-static motion for this system and mission (20 km/day). This indicates that MBD-SLAM is not well suited to the quasi-static environment, but is able to cope sufficiently well that the quadrotor is able to complete its mission with reasonable localization and mapping accuracy. This shows that integrating the velocity estimate – even if the quasi-static speed estimate accuracy is not acceptable – improves the localization and mapping performance of a UAV in slow dynamic and quasi-static environments compared to the traditional static SLAM.

5.4 Summary of Results and Discussion

The MBDFs and MBD-SLAM algorithm proposed in this chapter were shown to enable a 6-DOF quadrotor to localize itself in an unfamiliar and GPS-denied environment with slowly moving landmarks. Although these landmarks defy the static assumption that is required in traditional SLAM algorithms, MBD-SLAM is able to accommodate their motion due to the MBDFs that relax the static assumption. Note that the scope of this chapter was limited to operations within a single epoch of a piecewise-deterministic environment, and the parameters of MBD-SLAM were designed for the slow environment.

For MBD-SLAM, the landmarks are attributed a deterministic, constant velocity KMM, with the parameters of the model estimated in one of two ways: the LV approach, or the EM approach. Neither of these approaches require prior estimates of characteristic speeds or transit directions of the landmarks.

Two cases were studied to demonstrate MBD-SLAM in slow environments. Success here meant that the localization accuracy and static landmark mapping accuracy were statistically similar to the performance of the baseline static SLAM in a static environment (Section 4.4.4).

- Case 1 was the parent case for which MBD-SLAM was designed. This one-pass, full-scale simulation experiment showed that the UAV was able to successfully complete its mission while estimating landmark KMM parameters in near real-time. This ability to estimate the environment's changes is critical to missions in slow or quasi-static environments over long periods of time when treating the map as static fails, such as navigating through marine Arctic ice flows.

- Case 2 varied the scale of the experiment to be able to compare the simulation to experimental performance and to relate the performance to the model-scale baseline static SLAM implementation in Section 4.4.4. This case was also one-pass through a slow environment and showed that the mapping and localization performance using either the LV-MBD-SLAM or the EM-MBD-SLAM algorithms was significantly better than static SLAM applied to the slow dynamic environment in Section 4.5 and was comparable to static SLAM applied in the static environment (Section 4.4.4). This proves the hypothesis that incorporating landmark KMM's into the loop closure factors enables a UAV to operate in a slow dynamic environment that the UAV would otherwise be unable to persistently map.

Expectation maximization approximation performance

The LV and EM approaches each had their limitations, but both were able to successfully estimate the UAV's trajectory and produce a map, despite the presence of mobile landmarks. Initially, the EM approach was selected as an approximation method to reduce the computational time required by the LV approach to simultaneously estimate the KMM parameters alongside the trajectory in the graph. While this was successful in that the EM method did require less time to execute a pose iteration cycle, it was also shown that the performance of the EM did not suffer significantly compared to the LV approach. Two main drawbacks of the EM approach were the increased variation in performance between trials (the standard deviation across 10 repeated trials was larger) and the larger speeds estimated for static landmarks in the model-scale experiments compared to the LV approach. Neither of these disqualify the EM approach. Further, the speed estimate limitation can be mitigated by performing multiple passes through the environment. In general, LV-MBD-SLAM should be applied when the accurate estimation of landmark speeds for landmarks far from the start position is critical to the operator (or ship navigator) and the trajectory is a single one-dimensional pass, while EM-MBD-SLAM should be applied when on-board processing is limited.

MBD-SLAM and quasi-static landmarks

In Cases 1 and 2, MBD-SLAM was shown able to distinguish slow landmarks from static ones, however, the threshold used to make this distinction would have struggled had the mobile landmarks been quasi-static. For that reason, a third case was selected to quantify the performance of MBD-SLAM for this challenging case. This one-pass, full-scale, quasi-static landmark simulation experiment showed that despite MBD-SLAM being unable to distinguish quasi-static landmarks from static ones and poorly estimating the speeds of mobile landmarks, MBD-SLAM remained successful in the primary goal of allowing a UAV to localize itself and map a dynamic environment. Again the localization and mapping accuracies were comparable to the baseline static SLAM implementation in the static environment, despite the MBD-SLAM not being designed for quasi-static landmarks.

5.4.1 Concluding Remarks

MBD-SLAM relaxes the static assumption of traditional SLAM by including landmark KMMs in the factor graph construction. Both LV-MBD-SLAM and the EM-MBD-SLAM approximation were shown to have similar performance in the slow environment as the baseline static SLAM in a static environment. In the quasi-static environment, the localization and mapping performance was again comparable to the baseline static SLAM, however the speeds of the landmarks were over-estimated.

While these experiments were inspired by a mission to navigate marine Arctic environments, there are parallel cases in which MBD-SLAM can be used. The constant velocity KMM is just one example of a state transition model that could define the spatio-temporal evolution of a landmark. Any change in position, orientation, size, colour, reflectivity or other characteristic, that can be described by a deterministic, analytical model can have its parameters estimated in the graph alongside the vehicle's trajectory.

Given the results of the three cases in this chapter, **MBD-SLAM can be recommended over traditional SLAM algorithms for both slow and quasi-static environments, where vehicle localization accuracy is the primary requirement**. Improving the ability to accurately estimate landmark speeds in quasi-static environments is the focus of the next chapter.

Chapter 6 Proposed Model-Based Quasi-Static SLAM

This chapter extends the MBD-SLAM algorithm proposed in Chapter 5 to quasi-static landmarks. Specifically, this chapter defines the *quasi-static assumption* (QSA) that is required for *model-based quasi-static factors* (MBQSF), and proposes the *model-based quasi-static pose-graph SLAM* (MBQS-SLAM) algorithm. This algorithm is specifically designed for the quasi-static environments described in Chapter 3, but acceptable performance is also shown for landmarks that are all static and anomalously fast.

In the previous chapter, MBDFs were proposed to integrate the landmarks' kinematic model into the SLAM pose graph. The resulting MBD-SLAM algorithm enabled a UAV to successfully localize itself and produce a map of a dynamic environment populated with both static and mobile landmarks. Mobile landmarks were attributed a constant velocity KMM with parameters optimized in the SLAM pose graph or estimated using WLS linear regression. These KMMs are applied during a single epoch of a landmark's more general piecewise-deterministic motion model, meaning it is assumed that no events disrupt the landmark's KMM.

The primary difference between MBD-SLAM and MBQS-SLAM is the way that the landmark's KMM parameters (velocity) are treated in the graph. In MBD-SLAM the velocity is a latent variable, while in MBQS-SLAM, the graph treats the velocity as observable. Similar to the use of state observers (e.g. Luenberger observer) in control theory, an estimate of the velocity is used as a measure of the hidden state. The graph in MBQS-SLAM then simultaneously minimizes the error on the UAV's trajectory and the landmarks' velocities, rather than leaving the landmarks' velocities unconstrained (free variables) in MBD-SLAM.

By definition, in quasi-static environments, the velocity cannot be inferred from consecutive camera frames, thus, this chapter will test the hypothesis that (1) the WLS estimate of the quasi-static mobile landmarks converges after several observations, (2) the WLS estimate can be used as the measurement after it converges, and

(3) until that convergence, the velocity measure can be assumed zero with a standard deviation based on a characteristic landmark speed. This third element of the hypothesis will be called the QSA as it should only be assumed when the landmark motion is quasi-static. This hypothesis forms the proposed MBQS-SLAM algorithm. It will be shown that this approach has comparable localization and mapping performance to the baseline static SLAM applied to a static environment and is able to more accurately discern and estimate quasi-static landmarks from static landmarks than the MBD-SLAM proposed in Chapter 5.

This chapter starts by establishing the theory and assumptions to apply MBD-SLAM to quasi-static environments. First, the MBDFs from Chapter 5 are extended to include landmark speed measures (which were previously unobservable), forming MBQSFs (Section 6.1). Next, the convergence of the WLS estimate in the EM algorithm defined in Chapter 5 is studied, to determine if and, if so, when the WLS estimate is an effective measure of landmark velocity (Section 6.2). This study responds to element (1) of the hypothesis and is critical to using the WLS algorithm as an observer for the hidden velocity variable in quasi-static environments. Third, the QSA is defined and the associated parameters are discussed (Section 6.3). These elements are combined into the MBQS-SLAM algorithm (Section 6.4).

MBQS-SLAM is demonstrated through five cases, starting with the parent case that is most representative of mobile icebergs in marine Arctic environments. This shows that the QSA enables MBQS-SLAM to overcome the performance limitations of traditional static and the proposed MBD-SLAM in quasi-static environments. Next, the model-scale case is demonstrated in both a simulation and an experiment to determine that there is no degradation in performance in moving from the simulated to experimental environment. This is followed by three tests that explore the limiting cases of MBQS-SLAM, specifically: a slow moving landmark, a fully-static environment, and a two-pass mission. In summary, the five cases are:

1. one-pass, full-scale, quasi-static environment, simulation (Section 6.5.1) (analogous to Case 3 from Chapter 5),

2. one-pass, model-scale, quasi-static environment, simulation and experiment (Section 6.5.2),

3. one-pass, full-scale, anomalously fast (one landmark quasi-static, one landmark slow) environment, simulation (Section 6.5.3)

4. one-pass, full-scale, fully-static environment, simulation (Section 6.5.4)

5. two-pass, full-scale, quasi-static environment, simulation (Section 6.5.5)

While Cases 3 and 4 present the algorithm with the extremes of landmark motion in a quasi-static environment, Case 5 is chosen to quantify how the amount of time a vehicle remains on station, persistently mapping the region, impacts performance. Additional passes provide observations that span a longer duration and provide further opportunities to iterate landmark velocities that are only updated when the landmarks are in the camera's FOV. While in a static environment, this would neither improve nor degrade the localization and mapping accuracy, in a quasi-static environment, more passes should improve the performance. However, longer durations on-station increase the likelihood of an event disrupting the landmark's KMM. Case 5 bridges the gap between the one-pass mission of Case 1 and the two-pass missions, with events, that will be examined in Chapters 7 and 8.

This chapter concludes with a summary of the results and discussion on the applicability and limitations of the proposed MBQS-SLAM algorithm in Section 6.6.

6.1 Proposed Model-Based Quasi-Static Factors

The MBDFs proposed in Chapter 5 incorporate the landmarks' KMMs into the graph structure allowing the pose graph optimizer to successfully estimate both the UAV trajectory and the landmark velocities. In MBDFs, the error function to be minimized is related to the trajectory, while the velocities are free to take any values that minimizes the error on the UAV's trajectory. This means the velocity estimates are initially larger than the true value before approaching the true value as more observations are made. Given the static and very slow speeds of a quasi-static environment, constraining the velocity term to be within the quasi-static regime is beneficial.

The derivation of the MBQSF extends from the derivation of the MBDF in Chapter 5. The MBQSF error E concatenates the error e of the translation between poses

x_i and x_j, with the error e_v on the landmark velocity v_k. In MBQSF, the measurement h has two elements: h_s, as defined in the MBDF, and h_v, which is the graph's estimate of the velocity, $h_v = v_k$. The error for this factor is given by:

$$E = \begin{bmatrix} e \\ e_v \end{bmatrix} = \begin{bmatrix} h_s \ominus z_{ij,k} \\ h_v \ominus z_{vk} \end{bmatrix}, \tag{6.1}$$

where $z_{ij,k}$ is the measurement of the transform between poses x_i and x_j through landmark k, and z_{vk} is the measurement of velocity for landmark k. As in MBDFs, these measurements are probabilistic, meaning they are represented by a probability distribution (Gaussian) with a mean value and a standard deviation to capture the uncertainty of the measurement. This makes this version of the factor useful both when there is a direct velocity measurement (as is possible for fast landmarks), but also when the velocity can be constrained to a bounded region (as for quasi-static landmarks).

These proposed MBQSFs require selection or calculation of an appropriate landmark velocity measurement. Based on the success of EM-MBD-SLAM, the velocity calculated using WLS will be used. However, Chapter 5 showed that the WLS velocity does not converge to a stable value until several observations are made. Section 6.2 explores the convergence behaviour of the WLS velocity to determine a threshold beyond which the WLS velocity can be used, while Section 6.3 proposes a velocity measurement to be used before that convergence occurs.

6.2 Velocity Estimation Convergence

Section 5.1.2 defined EM-MBD-SLAM, which applies WLS regression to estimate the landmarks' velocity in the expectation step, then the graph optimizer, iSAM, is applied for the maximization step. In this Section, the evolution of the WLS velocity estimate is studied to assess its convergence. The standard deviation on the WLS velocity estimate (σ_{WLS}) is a measure of the confidence in the estimate. When the standard deviation becomes small, the estimate has incorporated enough observations that the velocity estimate can be considered as converged. The performance of SLAM using the WLS velocity estimate in quasi-static environments will be assessed in Section 6.5.

The WLS standard deviation is calculated after each landmark observation. The same full-scale, quasi-static environment as in Case 3 of Chapter 5 is used, except that two mobile landmarks *both* move the equivalent of 20 km/day. Motion is again restricted to the horizontal plane, where two elements of the translational velocity (v_x and v_y) and one of angular velocity (ω_z) are each estimated using WLS resulting in three standard deviations (σ_{vx}, σ_{vy}, $\sigma_{\omega z}$). The norm of the translational velocity standard deviation ($\sigma_v = \sqrt{\sigma_{vx}^2 + \sigma_{vy}^2}$) and the standard deviation on the yaw rate, are monitored as measures of convergence. While cross-validation tests (i.e. repeatedly using a different subset of the observations to calculate the standard deviations) would provide a quantification of the robustness and sensitivity of the WLS convergence, they would also require many more observations (passes) than performed in this thesis. Instead, this study provides a visual understanding of the WLS estimate's convergence on the observation pattern typical of trials performed in this thesis. Three variants of Case 1 are tested: (1) baseline one-pass, (2) one-pass with UAV at half-speed, and (3) two-pass. These variants determine first whether the evolution of σ_{WLS} is smoothly decreasing and second, whether the time between observations or the number of observations drive the decrease in σ_{WLS}.

For analysis, the environment is divided into four zones, numbered as shown in Figure 6.1. Each zone contains five landmarks. Black triangles indicate static landmarks while red triangles indicate mobile landmarks. Landmark 1 is known static a priori and is not included in the WLS convergence analysis. Zones are numbered from closest to the start position (zone 1: landmarks 12, 3, 14, 2, 14) to furthest from the start position (zone 4: landmarks 9, 20, 10, 21, 11). These zones are approximately aligned with the patterns seen in velocity estimation performance in Chapter 5. To assess whether landmark motions impact the convergence, the velocity estimates' standard deviations for the two mobile landmarks are plotted in red.

As in Case 1, a full-scale, quasi-static simulation will be performed using EM-MBD-SLAM. To clarify the language in the analysis in this section, observations of each landmark will be grouped into *batches* as shown in Figure 6.2. One batch of observations consists of several *consecutive* observations of the same landmark. For example, as the UAV travels from $x = 0$ m to $x = 100$ m, it passes through a region in which it can observe the landmark at one or more poses. At these poses, the first

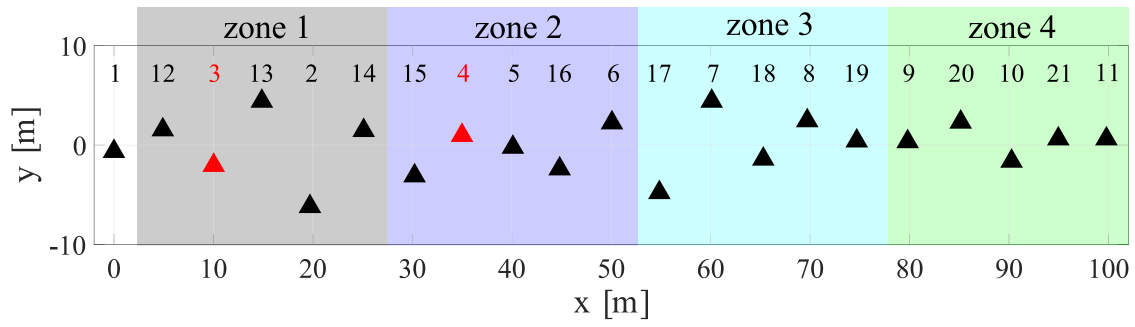


Figure 6.1: Map used for WLS convergence simulation study.

batch of observations of a landmark are captured. Then as the UAV returns along a reciprocal heading from $x = 100$ m to $x = 0$ m, a second batch of observations are captured. If a two-pass mission were performed, a total of four batches of observations would be collected for each landmark. In the limiting case where the observation frequency is such that only one observation is captured of each landmark as the UAV passes over it, the batch would contain only a single observation.

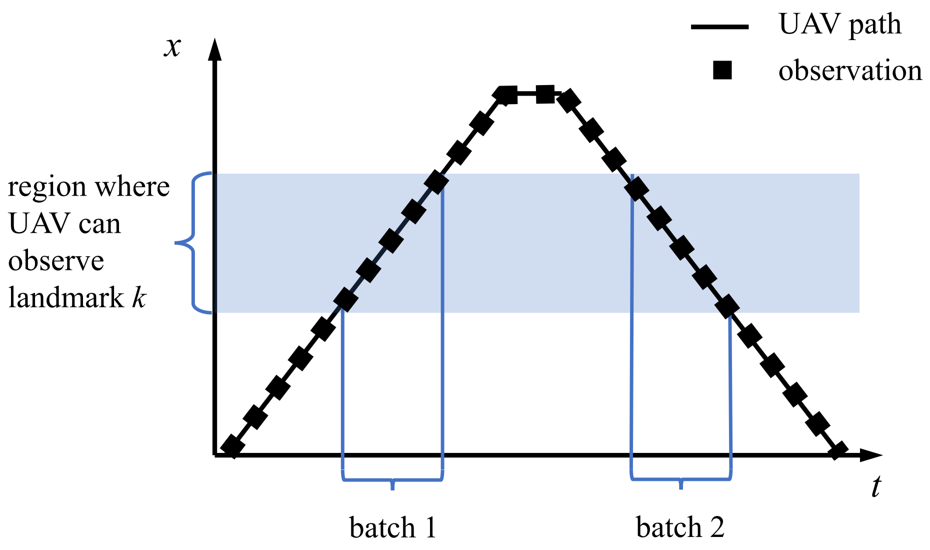


Figure 6.2: Definition of batches of observations used in analysis of WLS convergence.

Variant 1: One-pass (baseline) WLS convergence

The baseline test for studying WLS convergence is the one-pass, full-scale, quasi-static environment is shown in Figure 6.3. The overall trend is a consistent decrease in σ_{WLS} for each landmark as the number of observations used to calculate the velocity estimate using WLS increases. Halfway through the mission, after the 6th or 7th observation, there is a notable drop in the landmark's standard deviation. The 6th or 7th observation corresponds to the first observation of the landmark on the quadrotor's return trip (second batch). This drop is more distinct for landmarks in Zones 1 and 2 than in Zone 4.

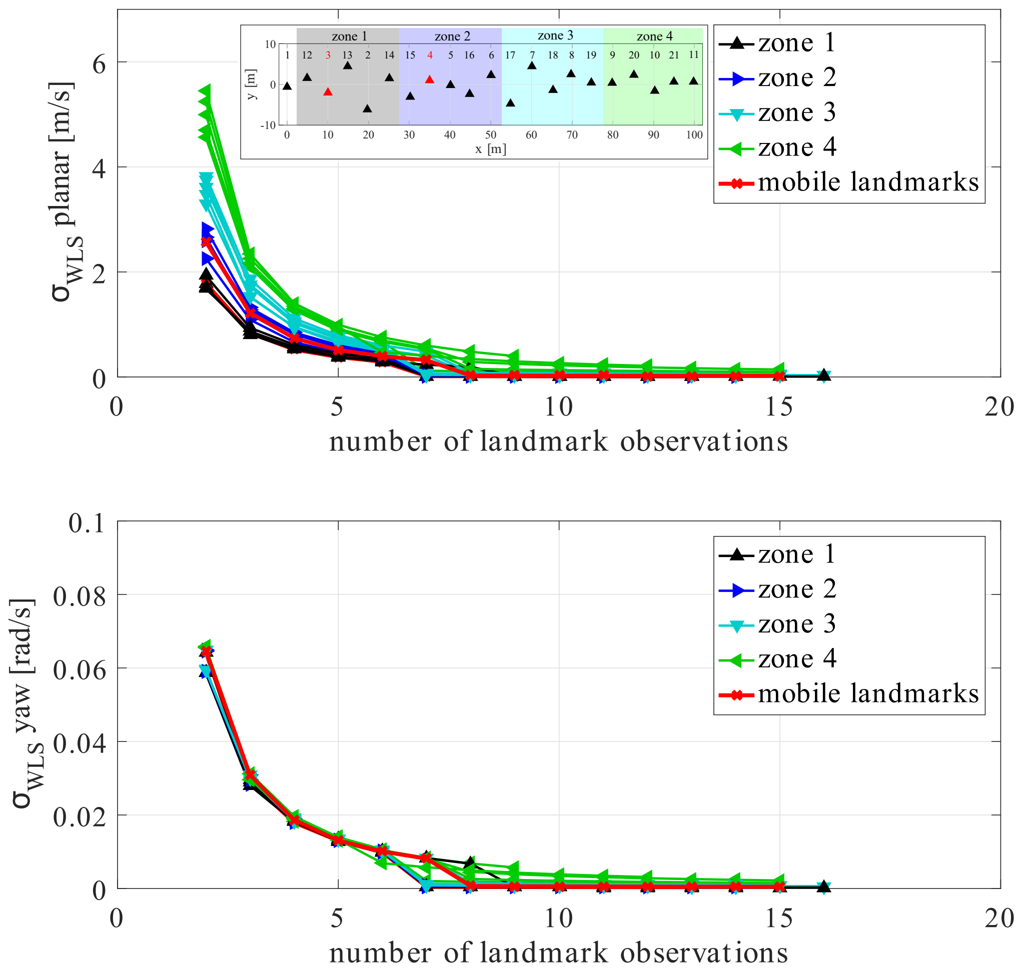


Figure 6.3: Standard deviation evolution for the translational (top) and angular (bottom) velocity estimates from WLS regression for Variant 1 (baseline).

Consider the observation sequence for the static landmark 15 as the quadrotor performs its mission. As the quadrotor enters Zone 2 (Figure 6.1), the first image with landmark 15 is captured. Given the quadrotor's pose update rate, the camera's FOV and the quadrotor's speed, landmark 15 is captured in five more consecutive images (ie. six consecutive poses observe landmark 15 in the first batch). As the quadrotor continues its mission, the FOV no longer captures landmark 15, until the quadrotor completes the outbound trajectory and performs its return journey. On the return trip, as the quadrotor passes through Zone 2, it captures a seventh image of landmark 15 as well as seven more consecutive images before landmark 15 is outside the FOV (concluding the second batch). These fifteen observations provide fifteen position estimates of landmark 15 over the two minute mission. The time between the two batches is longest for landmarks in Zone 1, and essentially zero for landmarks in Zone 4.

When calculated incrementally, as each new position estimate arrives, the largest impact on the landmark velocity estimate's standard deviation occurs when the first image of a batch is captured. This is because the time between images within a batch is too short to detect landmark motion in a quasi-static environment, while the time between batches is long enough to detect quasi-static motion.

This test showed that the standard deviation of the WLS velocity estimate (σ_{WLS}) smoothly decreases as more observations are made for all landmarks (static/mobile and all zones). It shows that the largest changes in standard deviation are when there are very few observations of a landmark, but once σ_{WLS} stabilizes only the first observation of a new batch has a notable impact on σ_{WLS}. Zone 4 showed the largest σ_{WLS}, while zone 1 showed the smallest σ_{WLS}, which is consistent with the expected impact of the quadrotor's larger pose uncertainty far from the start position. Zone 1 also showed the strongest impact of the first observation of the second batch while zone 4 showed the smallest impact with landmark 11 showing nearly no notable drop between batches. This implies that while some minimum number of observations is needed to have a converged velocity estimate, the most critical factor in the convergence is the time between batches of observations.

Variant 2: One-pass, half-speed UAV WLS convergence

Figure 6.4 shows the convergence for a quadrotor travelling at half the nominal speed (1.0 m/s). This case doubles the number of observations for each landmark as the quadrotor's camera FOV takes twice as much time to pass over the landmark. If the number of observations is critical, σ_{WLS} should be smaller for this variant than the previous one and the notable drop in σ_{WLS} should again occur after 6-7 observations. If the time is the critical factor, the drop should occur after the first observation in the second batch of observations, and the converged value of σ_{WLS} should be the same as in the first variant.

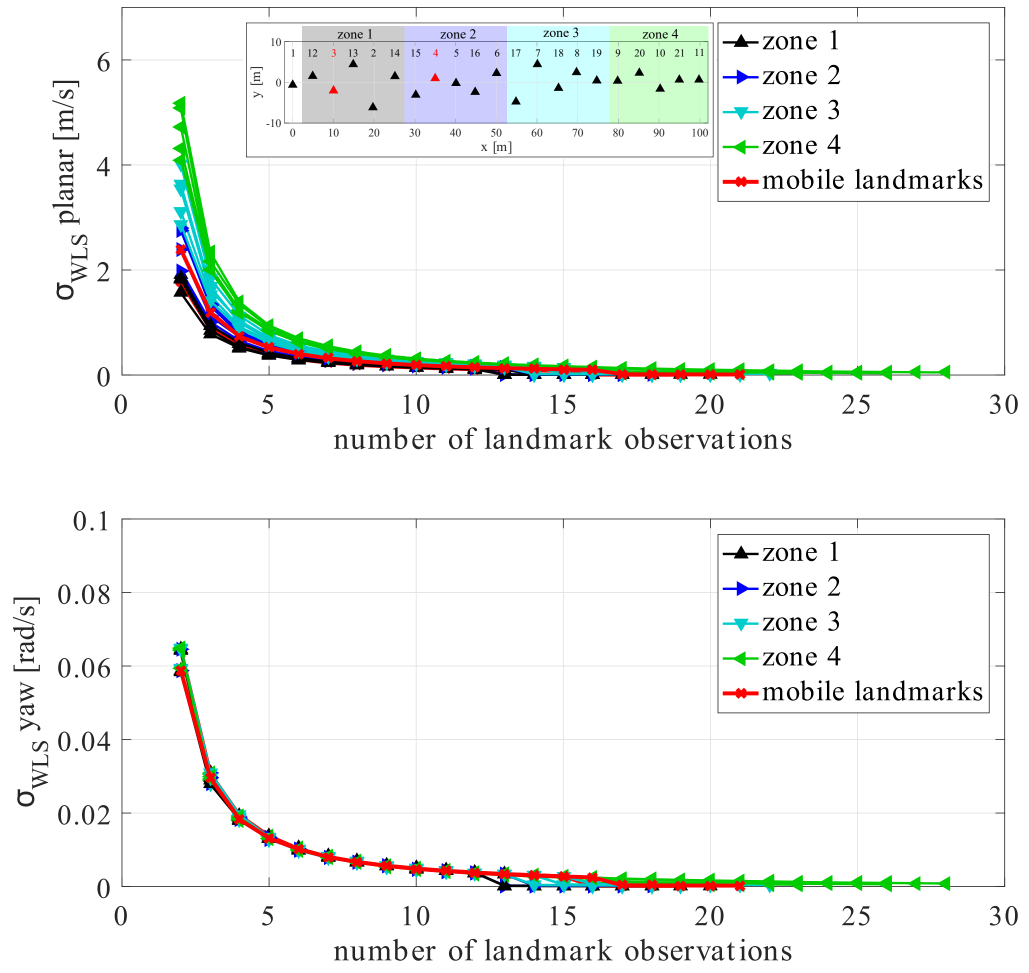


Figure 6.4: Standard deviation evolution of translational (top) and angular (bottom) velocity estimates from WLS regression, for Variant 2 (UAV travelling at half its nominal speed).

In Figure 6.4, there is a smaller, but still discernible, drop in the WLS velocity estimate's standard deviation at the 13th-16th observation of each landmark which, as in Case 1, corresponds to the first observation of the landmark on the quadrotor's return trip. The converged value of σ_{WLS} is smaller than Variant 1, but not by the amount that a doubling in number of observations would expect. Instead, this is more likely attributed to the fact that mission parameters of this variant (the half-speed quadrotor and the same landmark speed) no longer represent a quasi-static environment, as the time between the first and last observation in a batch is long enough to detect the landmark's motion. This is also supported by the smaller intensity of drop in σ_{WLS} as the later observations in the first batch have already contributed enough information for the estimate to stabilize without the need for the next batch.

This variant confirms the previous conclusion that while there is a minimum number of observations that are required for the WLS algorithm (approximately 4-5), σ_{WLS} converges when the time between observations within the set used for WLS estimation is large enough for quasi-static motion to be detected.

Variant 3: Two-pass, half pose update rate WLS convergence

To further explore the difference between the impact of number of observations and time between observations, the third variant doubles the distance the quadrotor travels (two-passes) and halves the pose update rate (ie. doubles the time between consecutive observations). This means the same number of observations are made per landmark as in Case 1, as shown in Figure 6.5. Confirming that the time between observations in the set of observations used for WLS estimation would mean the σ_{WLS} will converge at the same time as in the first variant (half the number of observations). This variant captures four batches of observations of each landmark (two batches per pass) and can also indicate if there is a velocity convergence benefit achieved by performing additional passes through the environment.

The third variant shows σ_{WLS} for most landmarks in zones 1, 2 and 3 converging after four observations, where the fourth observation is the first in the second batch of observations. For these landmarks, after this point σ_{WLS} does not show additional drops associated with the third or fourth batches for either static and mobile landmarks.

Landmarks in zone 4 show convergence after the sixth or seventh observation, which corresponds to the first observation in the third batch (the first observation

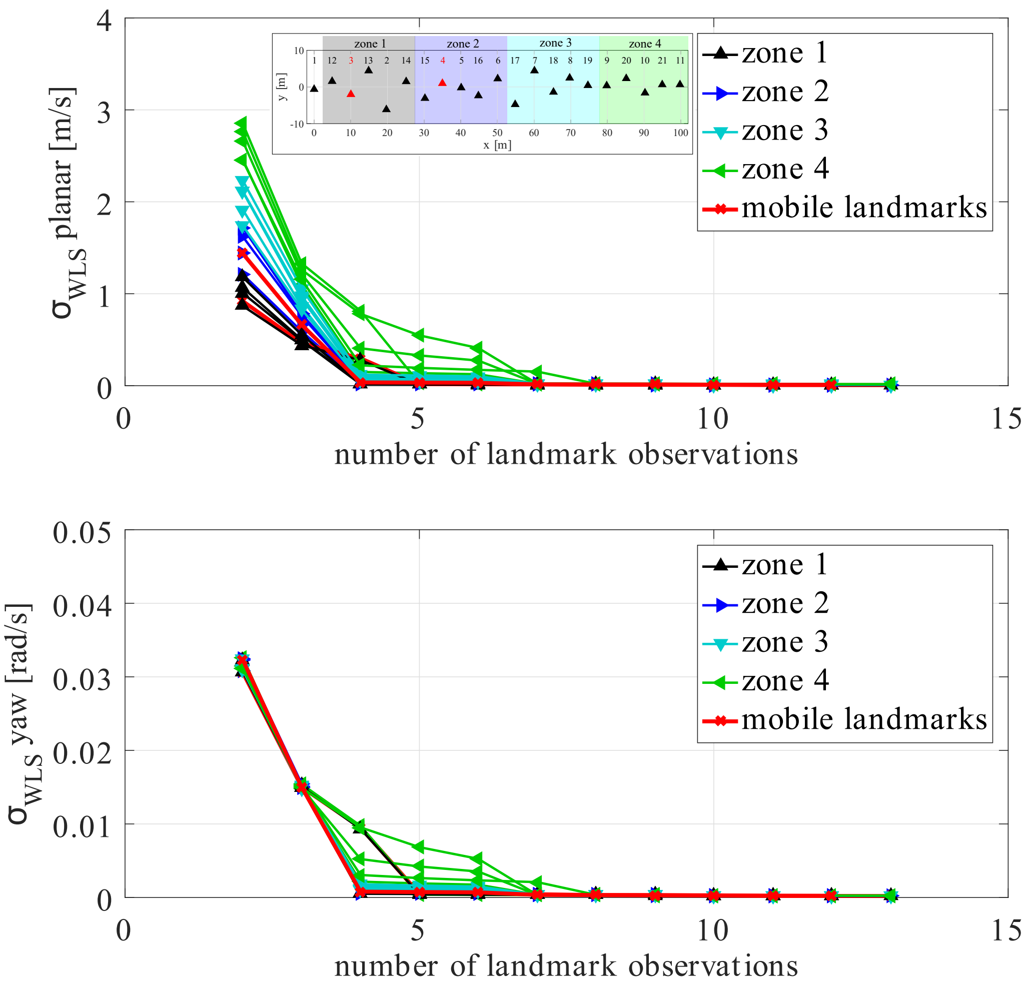


Figure 6.5: Standard deviation evolution for the translational (top) and angular (bottom) velocity estimates from WLS regression, for Variant 3 (two passes, half pose update rate).

of the second pass). As described in the typical observation pattern above, the time between observation batches 1 and 2 for landmarks in zone 4 is essentially zero, meaning that the observations of the first two batches do not span a sufficiently long duration to estimate a quasi-static speed. However, once the third batch begins, the elapsed time is sufficient for the velocity estimate to converge. This aligns with the discussion of results in Chapter 5 where velocity estimates for landmarks far from the start point were overestimated, while those closer to the start position were well estimated.

WLS Convergence study conclusions

The consistently decreasing convergence behaviour of σ_{WLS}, and the consistent converged value of σ_{WLS} across the three variants indicates that when the magnitude of the standard deviation is small, the velocity estimate can be treated as converged and used in the MBDF. Using the magnitude of σ_{WLS} removes the reliance on the number of landmarks and the time between observations whose thresholds would vary depending on the specific mission variant that was tested. The next section will discuss what should be done until the WLS velocity converges, and how to select the threshold of σ_{WLS} for a particular mission.

6.3 Proposed Quasi-Static Assumption

By definition, in a quasi-static environment, there is a set of static and mobile landmarks, where mobile landmarks move so slowly that two consecutive camera images cannot detect a change in the landmark's position, as defined in Section 3.4. In this chapter, Section 6.1 defined MBQSFs that extended the MBDFs from Chapter 5 to include an error term related to the landmark velocity. Then Section 6.2 showed that the WLS velocity estimate smoothly converges after observations span a sufficient duration to detect quasi-static motion. However, the MBQSFs require a velocity measurement for factors created before the WLS estimate converges. An assumption is proposed to provide that measurement.

Definition *quasi-static assumption (QSA)*:
If the landmarks' characteristic speed (υ_k) and the selected mission parameters are such that the environment is best described as quasi-static, as in Equation (3.4), the landmarks' velocity measurement shall be assumed zero, with a standard deviation of σ_{QSA}, until the WLS velocity estimate converges.

Until the WLS estimate converges, this assumption causes the graph to simultaneously minimize the error on the quadrotor trajectory, as well as the magnitude of the landmark speed. Minimizing the magnitude of landmark speed essentially treats the landmark as static with the certainty of it being static captured in σ_{QSA}. This means that mobile landmark speeds are likely to be underestimated, particularly if their speed is larger than the characteristic speed (υ_k).

The selection of σ_{QSA} should be related to the landmarks' characteristic velocity.

For the planar motion, constant velocity case, it is recommended that σ_{QSA} be set to the slowest characteristic landmark speed of interest. For this thesis, in the case of the marine Arctic environment described under the heading 'Landmark Speed' in Section 4.1.1 with a combination of static landmarks and planar motion landmarks with expected translational velocities of 10-20 km/day and angular yaw rates of 1 rotation every 10 minutes, σ_{QSA} is:

$$\sigma_{QSA} = \begin{bmatrix} \sigma_{QSA_x} \\ \sigma_{QSA_y} \\ \sigma_{QSA_z} \\ \sigma_{QSA_\phi} \\ \sigma_{QSA_\theta} \\ \sigma_{QSA_\psi} \end{bmatrix} = \begin{bmatrix} 10 \text{ km/day} \\ 10 \text{ km/day} \\ 2 \text{ km/day} \\ 0.02 \text{ RPM} \\ 0.02 \text{ RPM} \\ 0.10 \text{ RPM} \end{bmatrix}. \tag{6.2}$$

This σ_{QSA} is designed for a case when only planar velocities are expected, however, non-zero values for the z, roll and pitch components allow the factors to have some flexibility to landmarks that have velocity components outside the plane (such as a small roll rate). If more is known about the motion model this should be used to inform the selection of σ_{QSA}.

By selecting an appropriate value for σ_{QSA}, the graph allows the landmark velocity to vary within reasonable bounds, preventing the trajectory from potentially diverging as a result of large velocity estimates. This variance is preferable to treating the landmarks as strictly static, as that would result in the performance shown in Section 4.5. In addition, the parameter σ_{QSA} indicates the operator's confidence in the (prior) belief that the landmark is static. When in-situ information becomes available, generating a velocity estimate in which there is more confidence than the QSA, the QSA should no longer be applied to that landmark.

Threshold from QSA to WLS estimate

The choice of σ_{QSA} also defines the point to transition from using the QSA to the WLS estimate. When the confidence in the WLS estimate is better than the QSA covariance (broadly, $\sigma_{WLS} < \sigma_{QSA}$), the WLS estimate can be treated as the landmark velocity measurement. More specifically, the transition to using the WLS velocity estimate should occur when:

$$\sqrt{\sigma_{WLS_x}^2 + \sigma_{WLS_y}^2 + \sigma_{WLS_z}^2} < \sqrt{\sigma_{QSA_x}^2 + \sigma_{QSA_y}^2 + \sigma_{QSA_z}^2}, \tag{6.3}$$

and,

$$\sqrt{\sigma_{WLS_\phi}^2 + \sigma_{WLS_\theta}^2 + \sigma_{WLS_\psi}^2} < \sqrt{\sigma_{QSA_\phi}^2 + \sigma_{QSA_\theta}^2 + \sigma_{QSA_\psi}^2}, \tag{6.4}$$

are true. Note that for a given set of mission parameters that are typically not variable during a mission, σ_{QSA_v} and σ_{QSA_ω} are constants, while σ_{WLS_v} and σ_{WLS_ω} decrease as more observations are made of the landmark. Thus, the QSA will be used at the beginning of a mission, with landmarks transitioning to using their WLS velocity estimate as the mission progresses. Given the smoothly decaying σ_{WLS} evolution, unless there was an incorrect data association (for example due to an event), landmarks would never transition from using the WLS estimate back to using the QSA. This transition threshold will be used in the MBQS-SLAM algorithm defined in the next section.

6.4 Model-Based Quasi-Static SLAM Algorithm

The MBQS-SLAM algorithm is summarized in Figure 6.6. Much of the algorithm is the same as the baseline static SLAM (Figure 4.11), where a quadrotor performs pose-graph SLAM by filtering camera images using the ^{1}AprilTag detection algorithm [28] and the ^{2}iSAM optimizer [9], while following waypoints. However, there are two new blocks indicated in yellow.

The MBQSFs (B) are loop closure factors that simultaneously minimize the trajectory and the landmark velocity errors, as described in Section 6.1. The calculation of the landmark velocity measure (A) is similar to the WLS velocity estimate in EM-MBD-SLAM, but before the WLS velocity is used to build the MBQSFs, convergence is checked (Section 6.2). If the WLS velocity has not converged, the QSA is applied (Section 6.3). The proposed process to calculate the landmark velocity measure is defined in Figure 6.7.

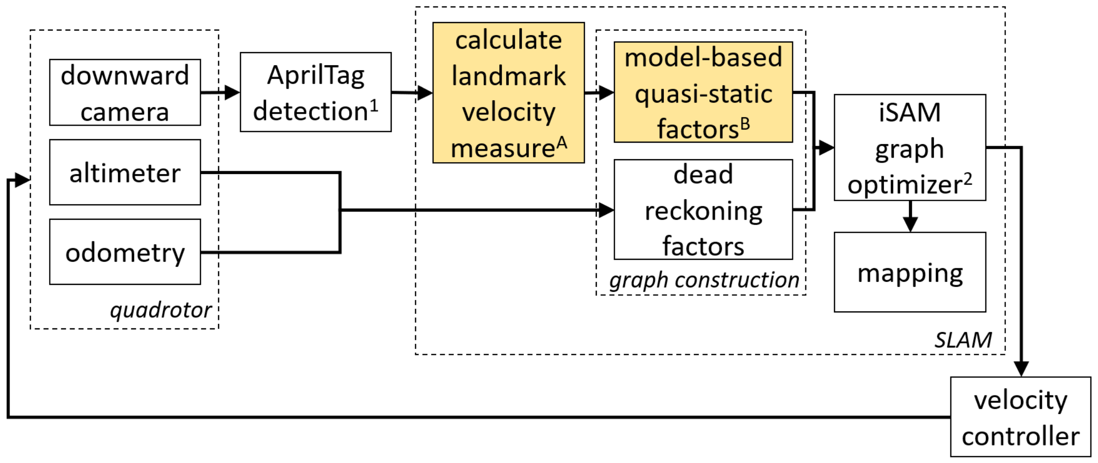


Figure 6.6: Proposed MBQS-SLAM algorithm. The yellow blocks indicate (A) the proposed calculation of the landmark velocity measurement using either the quasi-static assumption (QSA) or the WLS regression estimate, and (B) the novel MBQSFs.

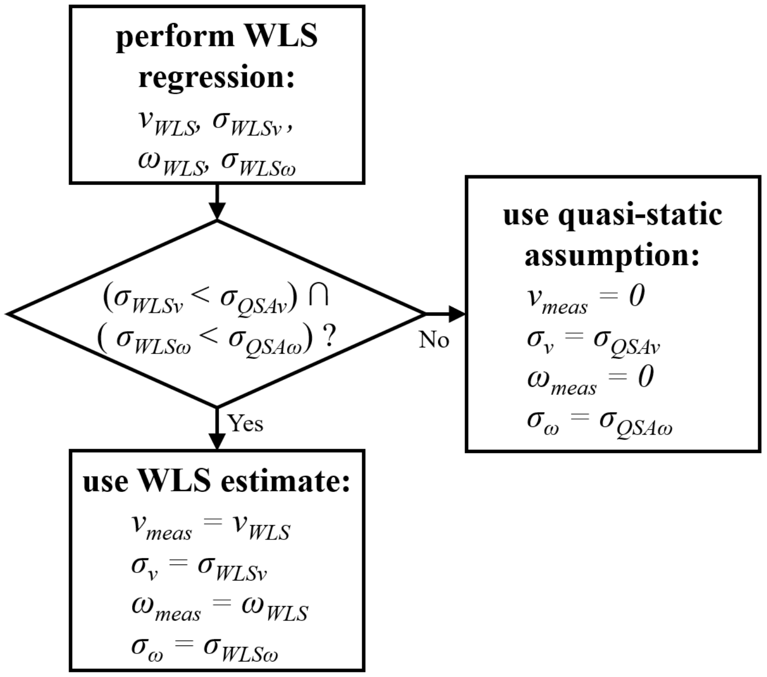

perform WLS regression:
v_{WLS}, σ_{WLSv}, ω_{WLS}, $\sigma_{WLS\omega}$
$(\sigma_{WLSv} < \sigma_{QSAv}) \cap (\sigma_{WLS\omega} < \sigma_{QSA\omega})$?
No
Yes
use quasi-static assumption:
$v_{meas} = 0$
$\sigma_v = \sigma_{QSAv}$
$\omega_{meas} = 0$
$\sigma_\omega = \sigma_{QSA\omega}$
use WLS estimate:
$v_{meas} = v_{WLS}$
$\sigma_v = \sigma_{WLSv}$
$\omega_{meas} = \omega_{WLS}$
$\sigma_\omega = \sigma_{WLS\omega}$

Figure 6.7: Proposed algorithm to calculate the landmark velocity measure to use in the MBQSFs.

6.5 Results

Five cases are selected to demonstrate the MBQS-SLAM performance. The summary of test cases in Table 4.6 includes five cases to test the MBQS-SLAM algorithm. The mission parameters for the full- and model-scale tests are given in Table 4.1. All cases restrict landmark motion to the plane (X, Y, ψ). The five cases are:

- **Case 1:** one-pass, full-scale, quasi-static environment, simulation (6.5.1)
  This represents the nominal mission for which MBQS-SLAM has been designed and shows the improvement of MBQS-SLAM compared to the performance of MBD-SLAM for the same Case in Section 5.3.3.

- **Case 2:** one-pass, model-scale, quasi-static environment, simulation and experiment (6.5.2)
  This case will be compared to the full-scale Case 1 and the baseline static SLAM from Section 4.4.4 to demonstrate the continued localization and mapping success in a more complex environment.

- **Case 3:** one-pass, full-scale, anomalously fast (one landmark quasi-static, one landmark slow) environment, simulation (6.5.3)
  Case 3 challenges the algorithm by applying MBQS-SLAM in an environment with one anomalously fast landmark (the speed of one landmark is slow, while both are expected to be quasi-static) to determine if the QSA harms the performance when landmarks are unexpectedly fast.

- **Case 4:** one-pass, full-scale, static environment, simulation (6.5.4)
  At the other extreme, Case 4 challenges MBQS-SLAM with an entirely static environment.

- **Case 5:** two-pass, full-scale, quasi-static environment, simulation (6.5.5)
  The two-pass case builds towards the more general dynamic environments discussed in Chapters 7 and 8, where events disturb landmark motion.

The covariance matrix used for the dead-reckoning factors match the one used in the baseline static SLAM and MBD-SLAM as given in Table 4.7. The MBQSF covariance matrix comprises 12 elements. The first six elements are related to the quadrotor's relative pose and match the values for the MBD-SLAM defined in Section 5.3. The last six elements are related to the velocity of the landmark – three for the translational velocity and three for the angular velocity – and are the same as the corresponding σ_v (in [m/s]) or σ_ω (in [rad/s]) element. The value of σ_v and σ_ω are the pre-selected QSA values until the WLS velocity converges, at which point they are equal to the WLS regression calculated standard deviations, as shown in Figure 6.7. The pre-selected values for σ_{QSA} for the model-scale environment are given in Table 6.1.

Table 6.1: QSA standard deviation (σ_{QSA}) on translational and angular velocity elements, specified for the model-scale case.

element	**unit**	σ_{QSA}
roll rate	[rad/s]	0.002
pitch rate	[rad/s]	0.002
yaw rate	[rad/s]	0.01
X speed	[m/s]	0.01
Y speed	[m/s]	0.01
Z speed	[m/s]	0.005

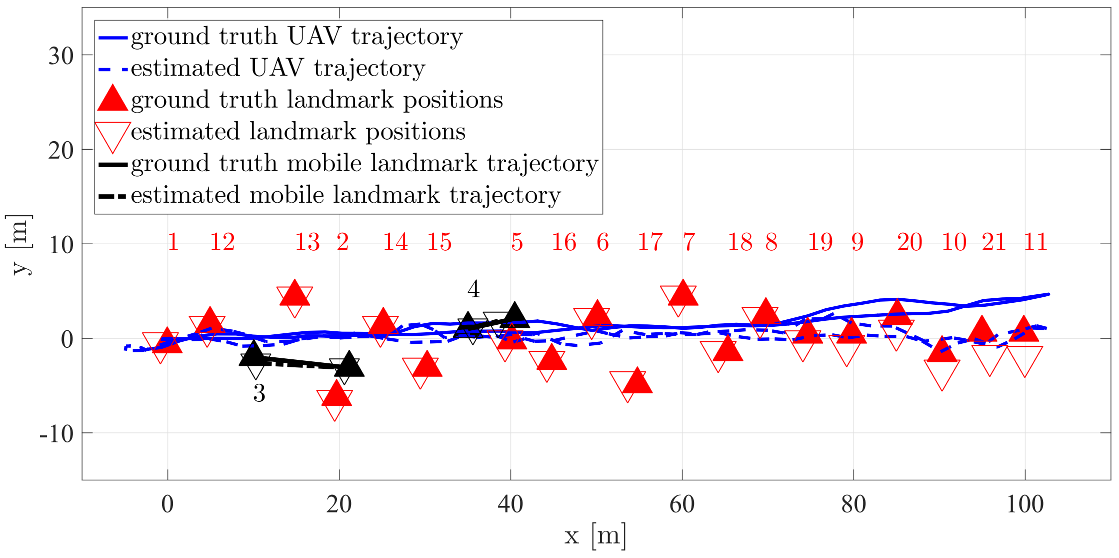
The selection of σ_{QSA} should exploit any prior knowledge of the environment. For this section, the covariance matrix is held constant for all five cases to assess how imprecise selection of σ_{QSA} impacts the performance when landmarks are faster (Case 3) or slower (Case 4) than expected.

6.5.1 Case 1: Full-Scale with Quasi-Static Landmarks

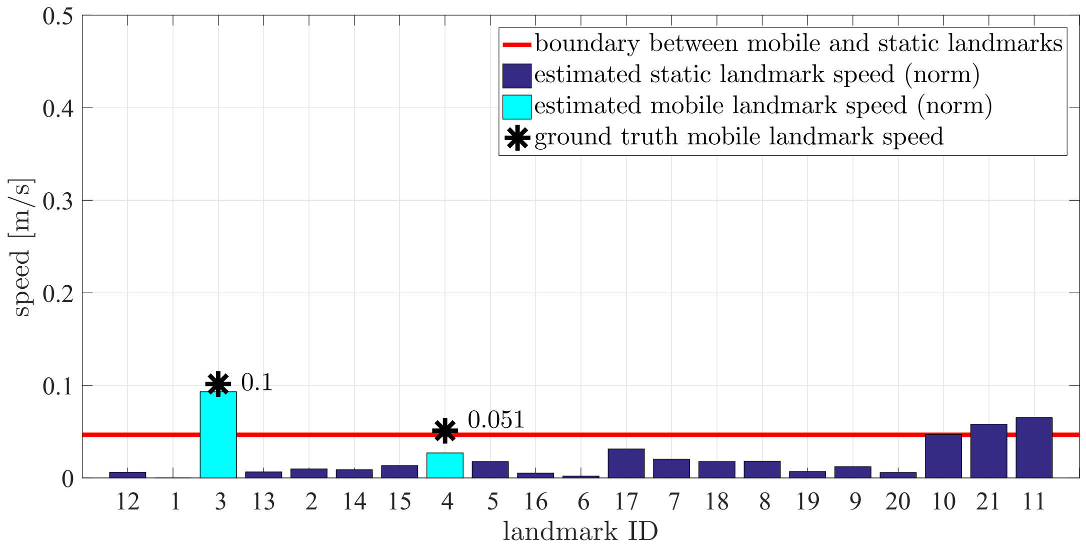
The performance of the proposed MBQS-SLAM algorithm in the full-scale, quasi-static environment for a one-pass mission is summarized in Table 6.2 and the median-performance trial is shown in Figure 6.8. This first case is the parent case for which MBQS-SLAM was designed with two mobile landmarks moving at the equivalent of 10 and 20 km/day (ie. quasi-static motion).

This case is the same as the third case for MBD-SLAM in Section 5.3.3. Recall that the localization and mapping accuracy of MBD-SLAM was similar to the baseline static SLAM performed in a static environment in Chapter 4, however the velocity estimates resulted in a boundary that was unable to discern mobile landmarks from static ones. MBQS-SLAM was designed to maintain the localization and mapping accuracy, while improving the velocity estimation accuracy and the ability to discern quasi-static motion from static landmarks.

Case 1 Localization and Mapping Results



(a) planar map



(b) speed estimates

Figure 6.8: Case 1 simulation of MBQS-SLAM in full-scale, quasi-static environment.

The planar map of the trajectory and landmark positions in Figure 6.8a shows little error in estimating landmark positions, particularly in the along-track direction. However, Table 6.2 indicates that the along-track error on the mapping is on average larger than the cross-track error. As noted previously, the cross-track error on the trajectory tends to grow as the UAV travels further from its start position, while the along-track error arises from over- or under-estimating the landmark speeds in the

Table 6.2: Case 1 (one-pass, full-scale, quasi-static) simulation performance summary for MBQS-SLAM (Figure 6.8).

metric	**unit**	**simulation**
localization error (norm)	[m]	4.42 ± 1.40
x error	[m]	2.41 ± 1.30
y error	[m]	2.16 ± 1.98
z error	[m]	1.75 ± 0.68
mapping error (norm)	[m]	4.05 ± 1.20
x error	[m]	2.50 ± 1.71
y error	[m]	1.50 ± 1.23
z error	[m]	1.89 ± 0.79
mean static landmark speed error	[m/s]	0.0476 ± 0.0149
mean dynamic landmark speed error	[m/s]	0.0299 ± 0.0189
mean speed boundary	[m/s]	0.0847 ± 0.0225

X-direction.

In terms of speed, the mean speed estimated for static landmarks was 0.048 m/s (equivalent to 9.6 km/day), while the boundary used to distinguish between the static and mobile landmarks was a speed of 0.085 m/s (17 km/day). Looking through the results of the ten trials also shows that while typically MBD-SLAM over-estimated the speeds of mobile landmarks, MBQS-SLAM tends to under-estimate their speeds as expected. This results in MBQS-SLAM performing more similarly to static SLAM during the early parts of a mission and performing more similarly to MBD-SLAM in the later parts of the mission. This is postulated to be a more successful behaviour outcome than the MBD-SLAM's over-estimated velocities for quasi-static environments, as quasi-static environments are expected to be well-mapped using static approaches for short-duration missions, until the displacement of landmarks becomes large (as discussed in Section 3.4).

Case 1 Discussion

The localization and mapping accuracies of the MBDQS-SLAM (4-5% relative to the 100 m pass-length) are statistically similar to the baseline static SLAM algorithm applied in a static environment (3-6%). Since the localization performance of the

UAV is the primary success criterion, the MBQS-SLAM performs acceptably in the quasi-static environment. The next question is whether it performs better than the MBD-SLAM of Section 5.3.3.

While the localization and mapping performance are similar to that shown in Case 3 for the LV- and EM-MBD-SLAM (Section 5.3.3), there is notable improvement in the velocity estimation accuracy. While MBD-SLAM showed a mean speed boundary of 25 km/day (equivalent), meaning quasi-static landmarks (10-20 km/day) could not be distinguished, for MBQS-SLAM, the boundary was 17 km/day. This means that the slowest landmarks of interest (10 km/day) could not be distinguished from the static landmarks, while the motion of 20 km/day landmarks can be distinguished. This is also apparent in the reduced mean static landmark speed estimate for MBQS-SLAM (14 km/day for MBD-SLAM, and 9.6 km/day for MBQS-SLAM).

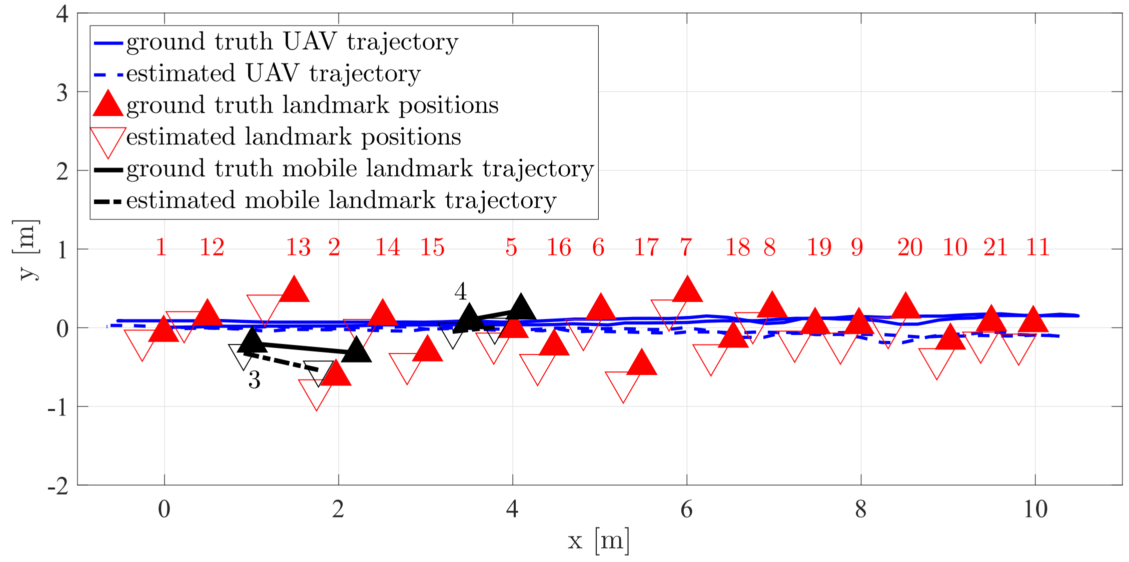
As expected, the velocities estimated using MBQS-SLAM are lower than their true values. This is because the QSA minimizes the magnitude of the landmarks' velocity until the WLS estimate converges. Thus, compared with MBD-SLAM which selects any value for the velocity that minimizes the UAV localization error, MBQS-SLAM chooses velocities close to zero initially. This underestimation can be controlled by selecting an appropriate σ_{QSA} as it governs the magnitude of the velocity estimate in the graph.

6.5.2 Case 2: Model-Scale with Quasi-Static Landmarks

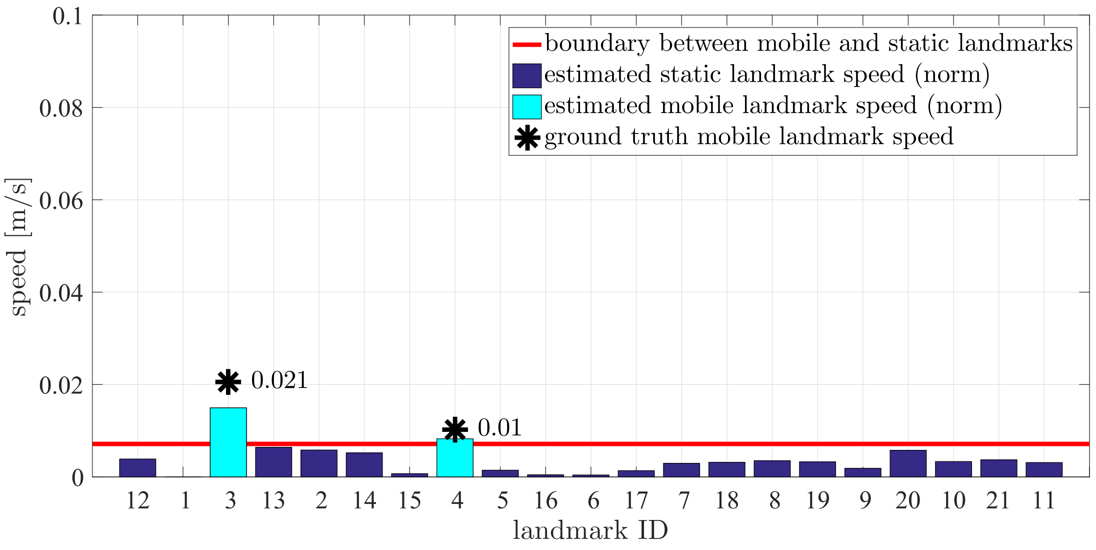
Next, MBQS-SLAM is tested in simulations and experiments at model-scale, as described in Chapter 4. Performance is summarized in Table 6.3. The simulation median-performance trial is shown in Figure 6.9 and the experiment median-performance trial is shown in Figure 6.10.

Case 2 Localization and Mapping Results

The planar map of trajectory and landmark positions for the trial with the median performance in simulation is plotted in Figure 6.9a. It shows a growing cross-track error as the UAV moves away from the start position, as well as a 0.1-0.2 m off-set in the along-track position of all landmarks and the poses in the UAV's trajectory. The localization and mapping error on average is approximately 5% of the 10 m pass



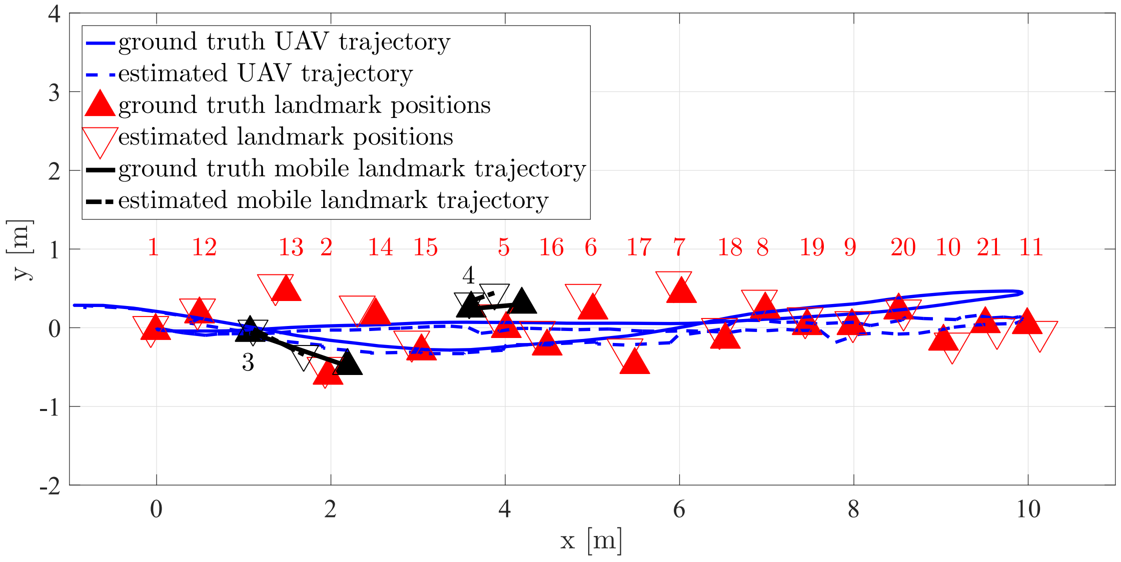
(a) planar map



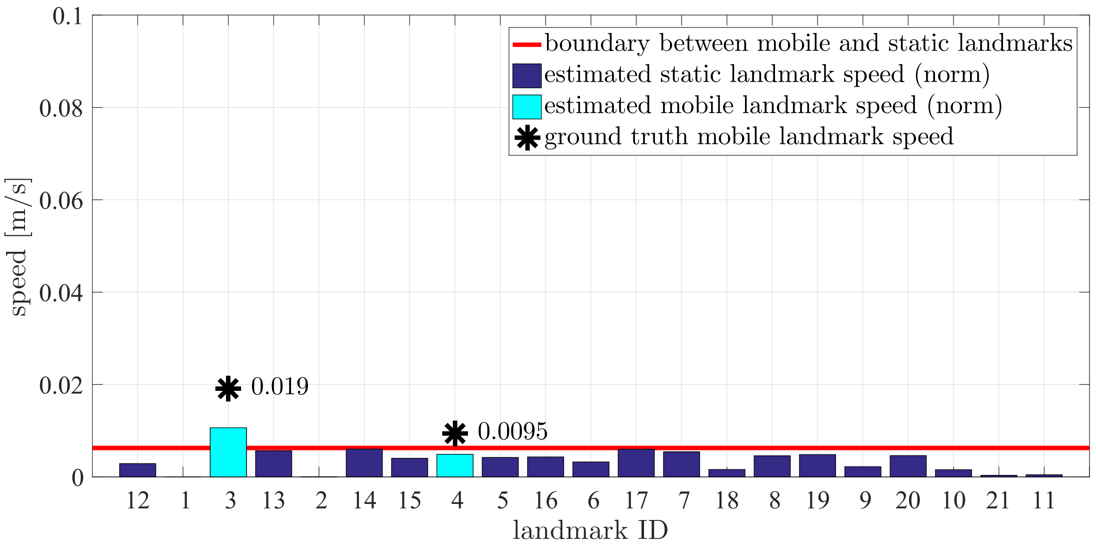
(b) speed estimates

Figure 6.9: Case 2 simulation of MBQS-SLAM in model-scale, quasi-static environment.

length with most of the error attributed to the altitude component of the estimate. The mean speed boundary between static and mobile landmarks is 8.5 km/day equivalent, meaning both the 10 km/day and 20 km/day landmark could be discerned if their speeds are well estimated. The error on the mobile landmark speeds is on average 0.0028 m/s (on 0.01 m/s and 0.02 m/s true speeds). As the mobile landmark speeds are typically underestimated using MBQS-SLAM, some trials do not succeed



(a) planar map



(b) speed estimates

Figure 6.10: Case 2 experiment of MBQS-SLAM in model-scale, quasi-static environment.

in distinguishing the 10 km/day landmarks from the static landmarks.

In the experiment, the Bebop quadrotor's motion deviated farther in cross-track from the designed one-dimensional trajectory, however, as seen in Table 6.3, the localization error in the cross-track direction is not statistically larger than in the simulation. In the experiment, the altitude components of the localization and mapping errors were largest by a small margin compared with the along-track and cross-track

Table 6.3: Case 2 (one-pass, model-scale, quasi-static) simulation (Figure 6.9) and experiment (Figure 6.10) performance summary for MBQS-SLAM.

metric	**unit**	**simulation**	**experiment**
localization error (norm)	[m]	0.48 ± 0.19	0.37 ± 0.09
x error	[m]	0.14 ± 0.06	0.20 ± 0.06
y error	[m]	0.10 ± 0.03	0.13 ± 0.06
z error	[m]	0.42 ± 0.19	0.21 ± 0.09
mapping error (norm)	[m]	0.50 ± 0.19	0.35 ± 0.10
x error	[m]	0.15 ± 0.07	0.17 ± 0.07
y error	[m]	0.13 ± 0.11	0.14 ± 0.08
z error	[m]	0.42 ± 0.20	0.22 ± 0.11
mean static landmark speed error	[m/s]	0.0036 ± 0.0011	0.0031 ± 0.0005
mean dynamic landmark speed error	[m/s]	0.0028 ± 0.0009	0.0062 ± 0.0018
mean speed boundary	[m/s]	0.0085 ± 0.0016	0.0066 ± 0.0005

errors. Speeds of mobile landmarks were again under-estimated in the experiments as indicated by the larger mean error for the mobile landmark speed estimates (0.0062 m/s) than for the static landmark speed estimates (0.0031 m/s). The boundary between the static and mobile landmark speeds was on average 6.6 km/day meaning landmarks at 20 km/day could be discerned, while landmarks at 10 km/day may struggle to be assigned as mobile given the error on mobile landmark speed estimation.

Case 2 Discussion

In general, the patterns in performance of simulation and experiment trials were similar with a few exceptions. While the along-track component of the the localization error was larger in the experiment than in the simulation, the altitude component was notably smaller, resulting in localization and mapping errors smaller in the experiment than in the simulation. This aligns with the expectation that the simulated quadrotor altimeter is modelled with the worst case noise model to be conservative. The mean speed boundary in both cases was below the 10 km/day quasi-static landmark speed, meaning quasi-static landmarks could be distinguished from static landmarks. On the other hand, speed estimates of mobile landmarks were notably better in simulation than in the experiment, possibly due to the smooth, consistent motion of simulated landmarks compared to the somewhat lurching motion of NXT landmarks at low speed in the experiment.

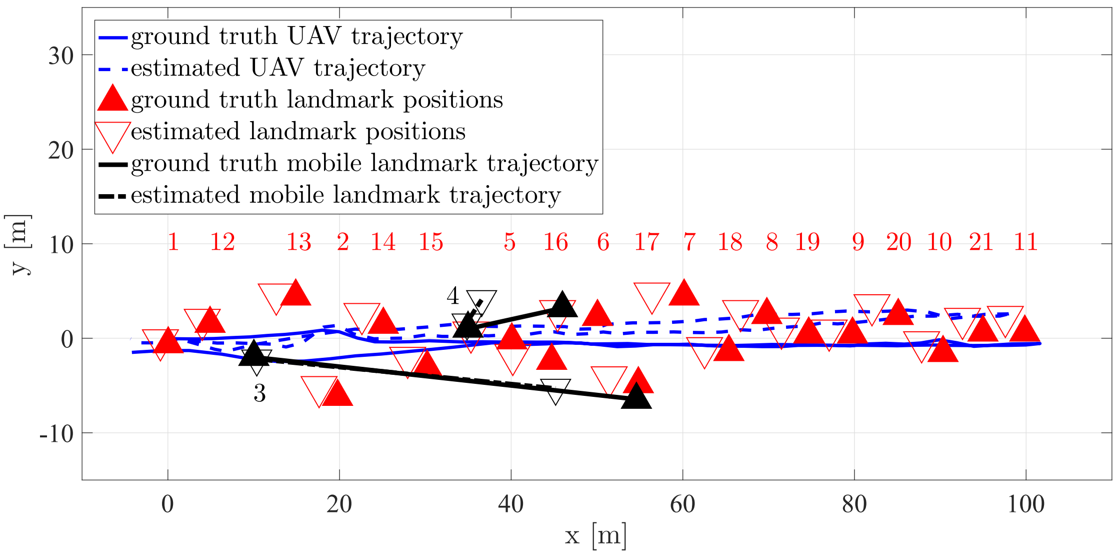
Compared to Case 1, Case 2 shows very similar performance with the full-scale simulation performance falling between the model-scale simulation and experiment performance in terms of localization and mapping errors. The model-scale trials show improvements to the velocity estimation accuracy of both the static and mobile landmarks, resulting in the boundary between static and dynamic landmarks being a slower speed. At model-scale, an iceberg with an equivalent speed of 7-9 km/day could be detected and the average static landmark is estimated with a speed of 3-4 km/day. This improvement compared to the full-scale case (17 km/day boundary between static and mobile landmarks) is attributed to the lower impact of quadrotor attitude error propagation over shorter distances.

6.5.3 Case 3: Full-Scale with Anomalously Fast Landmark

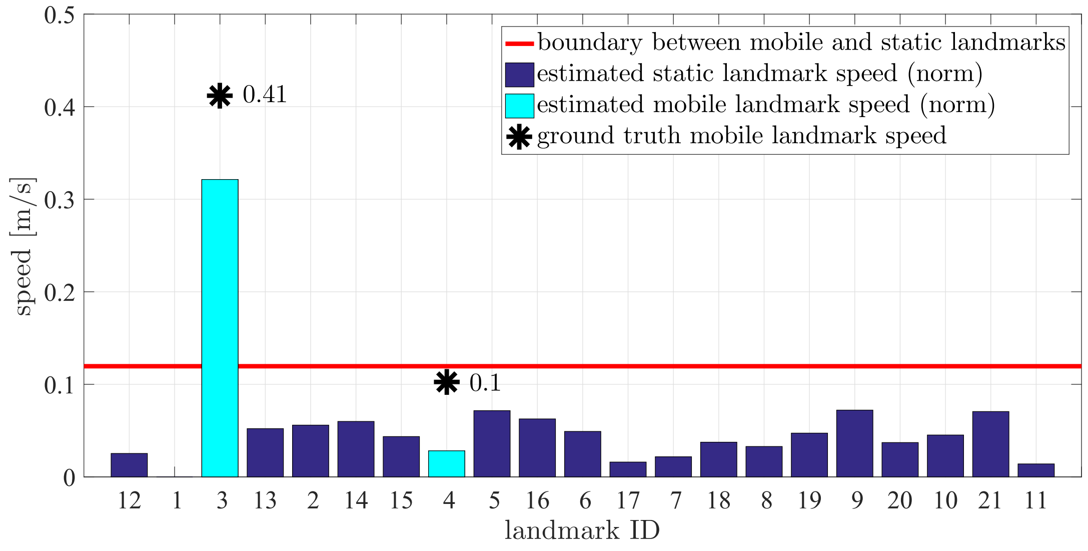
As it is impossible to be certain that all landmarks in an environment are quasi-static, the MBQS-SLAM must be robust to faster-than-expected landmarks. In this case, an anomalously fast landmark travelling at 80 km/day (fastest speed used in the slow environment tests in Chapter 5) and a quasi-static landmark at 20 km/day are simulated. Note that the value of σ_{QSA} is unchanged from the parent case (Case 1) as the faster landmark is unexpected. The performance is summarized in Table 6.4 and the median-performance trial is shown in Figure 6.11.

Case 3 Localization and Mapping Results

The localization and mapping error remains at 3-4% of the pass length (100 m) despite the presence of a landmark moving much faster than σ_{QSA} was designed for. However, the error on both static and mobile landmark speed estimation is much larger. For the anomalously fast landmark (labelled landmark 3), the transition from using the QSA to using the WLS estimate is delayed. The consequence is the underestimation of this landmark's speed, subsequently underestimating the UAV's change in X position when it observes that landmark. This makes the nearby static landmarks appear to be moving in the negative X direction and makes landmark



(a) planar map



(b) speed estimates

Figure 6.11: Case 3 simulation of MBQS-SLAM in full-scale, anomalously fast environment (one quasi-static landmark and one slow landmark).

4 (the quasi-static landmark) appear to be slower. This accounts for the underestimation of the positions of landmarks in the along-track direction of approximately 2.5 m, the high static speed error of 6.6 km/day and the shorter estimated trajectory than ground truth trajectory in the X direction. It also exacerbates the error on the mobile landmark speed estimates. In turn, this leads to an increase in the boundary used to discern mobile from static landmarks to 22 km/day.

Table 6.4: Case 3 (one-pass, full-scale, anomalously fast) simulation performance summary for MBQS-SLAM (Figure 6.11).

metric	**unit**	**simulation**
localization error (norm)	[m]	3.52 ± 1.03
x error	[m]	2.51 ± 1.31
y error	[m]	1.11 ± 0.32
z error	[m]	1.45 ± 0.69
mapping error (norm)	[m]	3.38 ± 1.01
x error	[m]	2.51 ± 1.33
y error	[m]	0.90 ± 0.30
z error	[m]	1.53 ± 0.76
mean static landmark speed error	[m/s]	0.0332 ± 0.0068
mean dynamic landmark speed error	[m/s]	0.0789 ± 0.0229
mean speed boundary	[m/s]	0.1111 ± 0.0080

Case 3 Discussion

The definition of the boundary between the static and mobile landmarks is one standard deviation above the mean speed of all landmarks. In this case, the boundary is skewed by the large speed for the anomalously fast landmark. If instead the boundary were calculated iteratively – first calculating the boundary with all landmarks, then removing any that are faster than the boundary and re-calculating the boundary – the boundary decreases. For instance, for the plotted example, the mean speed boundary calculated without landmark 3 is the equivalent of 12 km/day, which is similar to Cases 1 and 2. As the boundary is used only in post-processing data, its definition does not impact the functionality of MBQS-SLAM, but is important for an operator acting upon the results.

While the localization and mapping errors are comparable in the anomalously fast case to the parent case (Case 1) and to the baseline static SLAM applied in the static environment, the mobile landmark speed estimates and boundary both deteriorated in the presence of an anomalously fast landmark. This means that while the quadrotor would successfully localize itself to complete its mission, it would be difficult to make a safe navigation plan as the quasi-static landmark could not be detected. If this situation were to occur, the recommendation would be to perform an additional pass over the landmarks to take advantage of the improved estimate for landmark 3 that is available at the end of the mission.

The anomalously fast case is useful for comparing the velocity estimates from MBQS-SLAM to those from MBD-SLAM. In Chapter 5, MBD-SLAM was shown to over-estimate the velocities of mobile landmarks and typical estimates for static landmarks included motion in the positive X direction (aligned with the motion of the quadrotor). MBQS-SLAM on the other hand underestimates slow mobile landmark velocities and estimates of static landmark velocities are in the negative X direction. The QSA is responsible for this reduction in magnitude of the speed estimates.

Note that as the number of landmarks moving faster than the characteristic speed grows, so does the graph's effort to minimize the magnitude of the speed of those mobile landmarks. Assuming the mobile landmarks move in a consistent direction, as they do in this thesis, in compensation for these under-estimates, static landmarks are attributed motion in the opposite direction of the mobile landmarks. To avoid this, additional known-static landmarks such as landmark 1 could be deployed to strengthen the tie of the relative factors in the graph to the inertial reference frame. This approach is not available in an unstructured Arctic environment, making the proper selection of σ_{QSA} important.

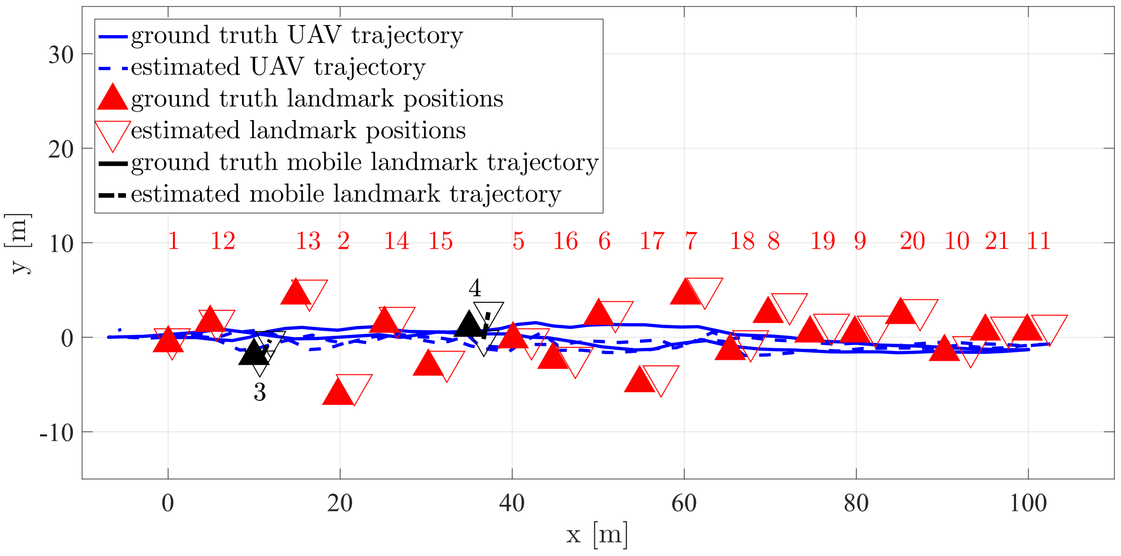
While MBQS-SLAM is shown to be robust to an anomalously fast landmark, it is preferable to select σ_{QSA} to suit the environment when possible, to improve the speed estimates of the landmarks.

6.5.4 Case 4: Full-Scale with All Static Landmarks

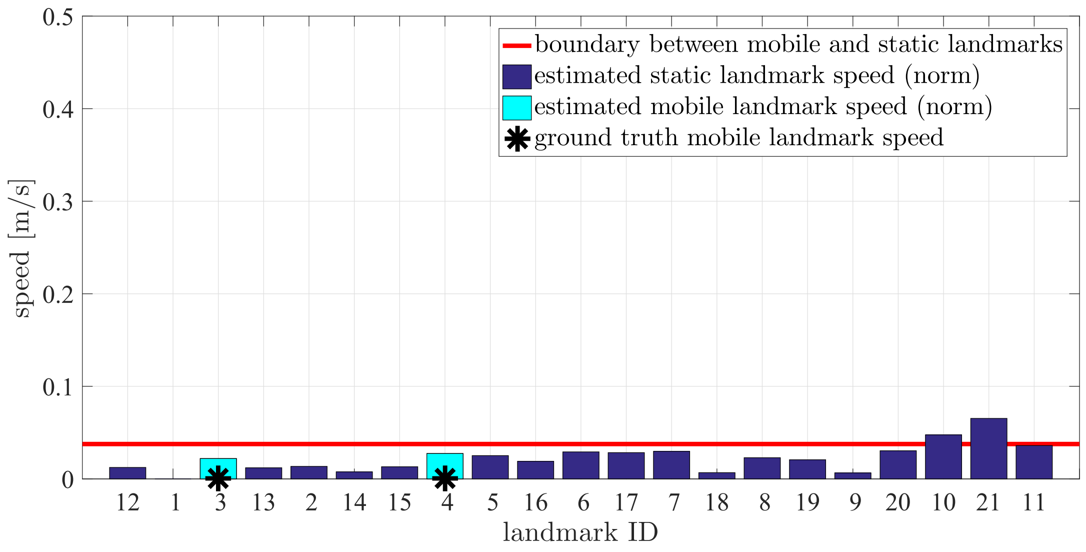
The opposite limiting case to the anomalously fast landmark in Case 3, is a fully static environment. This case gives insight into the lower-bound on the landmark speeds that can be detected by simulating all 21 landmarks as stationary while using σ_{QSA} designed for the environment of Case 1. The performance is summarized in Table 6.5 and the median-performance trial is shown in Figure 6.12.

Case 4 Localization and Mapping Results

The localization and mapping errors (3% and 3.3% respectively of the 100 m pass length) are similar to the previous three cases in quasi-static environments. The



(a) planar map



(b) speed estimates

Figure 6.12: Case 4 simulation of MBQS-SLAM in full-scale, static environment.

most notable difference to the parent case (Case 1) is the boundary between static and mobile landmarks is less than 8 km/day (17 km/day in Case 1). The average speed estimated for the static landmarks (all 21) is 5 km/day, also smaller than Case 1 (9 km/day).

Table 6.5: Case 4 (one-pass, full-scale, static) simulation performance summary for MBQS-SLAM (Figure 6.12).

metric	**unit**	**simulation**
localization error (norm)	[m]	3.10 ± 1.23
x error	[m]	1.82 ± 1.16
y error	[m]	0.81 ± 0.64
z error	[m]	1.84 ± 1.00
mapping error (norm)	[m]	3.29 ± 1.51
x error	[m]	2.16 ± 1.39
y error	[m]	0.78 ± 0.54
z error	[m]	1.91 ± 1.11
mean static landmark speed error	[m/s]	0.0241 ± 0.0059
mean speed boundary	[m/s]	0.0391 ± 0.0094

Case 4 Discussion

Although this Case is performed at full-scale, the localization and mapping accuracy is even slightly better (3-3.5% of the 100 m pass length) than the baseline Static SLAM in a static environment from Section 4.4.4 (4.5-5.5% in simulation relative to the 10 m pass length). Given this, unless on-board processing is limited, there is no downside to applying MBQS-SLAM in a static environment, if there is a possibility of mobile landmarks.

6.5.5 Case 5: Full-Scale with Multiple Episodes

Case 5 is the two-pass, full-scale, quasi-static environment simulation of MBQS-SLAM. It is identical to Case 1 in this chapter, except:

1. after the quadrotor returns to the start position it pauses for 5 seconds then performs a second pass from $x = 0$ m to $x = 100$ m, and,

2. the pose update rate is halved (to 0.5 Hz) to keep the total number of observations of each landmark the same.

This update rate change is reflected in the X and Y elements of the dead-reckoning factor covariance matrices as those elements are based on the optical flow speed measurement (Table 4.7). All other covariance matrices are the same as Case 1.

As discussed earlier in this chapter, while the second pass in a static environment does not improve or degrade the static SLAM performance, a second pass over mobile landmarks results in a longer time spanned by the set of observations used to calculate the WLS velocity estimate. The WLS convergence study in Section 6.2 showed that longer time spanned by observations resulted in more certainty in the velocity estimate. This means the performance is expected to improve in this case compared to Case 1.

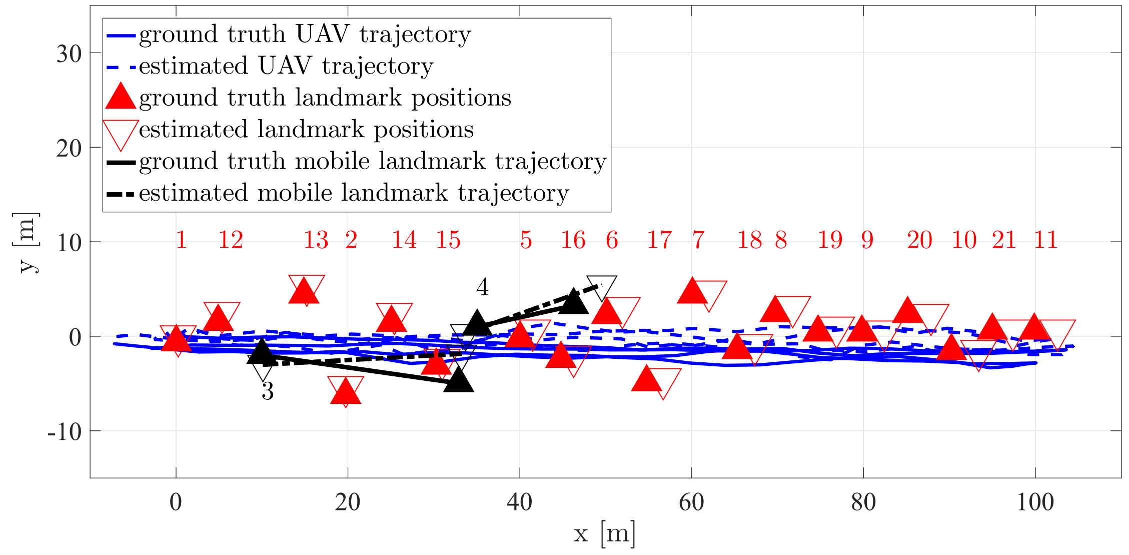
On the other hand, as more time passes, the likelihood of an event occurring that disrupts a landmark's motion increases (see Chapter 3). While an event is not simulated in this case, events will be studied in Chapters 7 and 8 in a mission and environment otherwise similar to this case.

Also, it was previously noted that since the velocity estimates used to construct the loop closure factors are only updated when the landmark is observed, there is more uncertainty in the velocity estimates of landmarks 10, 21 and 11 than in landmarks 12, 13 and 2 as they cannot take advantage of the loop closure factors constructed during the return to the start location. These new factors are only used to re-estimate landmarks 10, 21, and 11 when those landmarks are re-observed during the second pass, which is expected to improve their estimation accuracy.

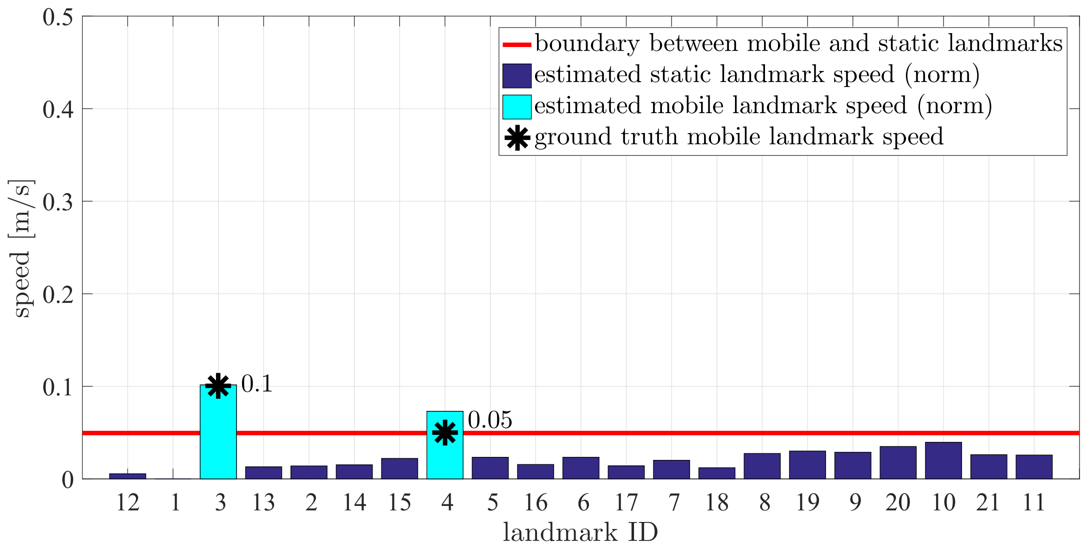
The performance is summarized in Table 6.6 and the median-performance trial is shown in Figure 6.13.

Case 5 Localization and Mapping Results

Table 6.6 shows an average localization error of 3.6 m and an average mapping error of 3.1 m over 10 repeat trials. The median case plotted as a map in Figure 6.13a does not show a growth in the cross-track error further from the start position, and shows good accuracy in the estimated and ground truth mobile landmark paths. This is also seen in the bar chart of landmark speeds in Figure 6.13b with well estimated mobile landmark speeds and all static landmarks correctly located below the boundary line. However, the table shows that statistically, the mean speed boundary is 12.4 km/day which would be too fast to discern that the 10 km/day landmark was mobile. The mean static landmark speed estimate (ie. error on the static landmarks) was 6 km/day, while the mobile landmarks were more accurately estimated with an average



(a) planar map



(b) speed estimates

Figure 6.13: Case 5 simulation of MBQS-SLAM in full-scale, quasi-static environment with UAV performing two passes.

error of 3.6 km/day.

Case 5 Discussion

The two pass mission not only showed an improvement over the one-pass mission (see Table 6.2) in velocity estimation accuracy and a lower boundary for distinguishing

Table 6.6: Case 5 (two-pass, full-scale, quasi-static) simulation performance summary for MBQS-SLAM (Figure 6.13).

metric	**unit**	**simulation**
localization error (norm)	[m]	3.60 ± 0.80
x error	[m]	1.88 ± 0.45
y error	[m]	1.76 ± 0.83
z error	[m]	1.86 ± 0.69
mapping error (norm)	[m]	3.05 ± 0.95
x error	[m]	1.22 ± 0.57
y error	[m]	1.36 ± 0.89
z error	[m]	2.01 ± 0.85
mean static landmark speed error	[m/s]	0.0304 ± 0.0117
mean dynamic landmark speed error	[m/s]	0.0183 ± 0.0090
mean speed boundary	[m/s]	0.0622 ± 0.0139

mobile landmarks from static landmarks, but also showed an improvement in localization and mapping accuracies. These improvements were consistent across the ten trials as indicated by the smaller standard deviation on nearly every metric (the exception being the altitude error for both localization and mapping which each had a smaller mean value, but a larger standard deviation in the two-pass trials). Much of the improvements in speed and mapping estimates are due to the improvements on landmarks that are farthest from the start position (landmarks 10, 21 and 11), as expected. Case 5 also shows that reducing the SLAM update rate to 0.5 Hz does not reduce the localization and mapping accuracy.

The improvement in mapping and localization accuracy is not sufficient to recommend performing multiple passes in every case due to the costs in terms of (1) energy, (2) processing power, and (3) time delay. However, if the accuracy of landmark speed estimation is critical to the mission, then multiple passes are effective towards this.

6.6 Summary of Results and Discussion

This chapter addressed the limiting case of quasi-static environments by extending MBD-SLAM to MBQS-SLAM using the proposed MBQSFs. The quasi-static limiting case is particularly interesting as direct measurements of landmark velocity are not possible, making visual-SLAM approaches to dynamic environments unsuitable. Optimizing the parameters of quasi-static landmark KMMs in the graph was shown to be effective, where otherwise this motion could cause the localization estimate to diverge.

Focused only on the deterministic motion during an epoch between events, the novel MBQS-SLAM method used the proposed QSA to treat the landmark velocity as zero until the WLS regression velocity estimate converged. After defining the MBQSFs and the QSA, as well as studying the convergence of the WLS estimate, MBQS-SLAM was defined and successfully demonstrated in simulations at full- and model-scale and in experiments at model-scale.

In Cases 1 and 2, MBQS-SLAM proved to have similar mapping and localization accuracies to MBD-SLAM in quasi-static environments, while the velocity estimation accuracy improved with MBQS-SLAM. MBQS-SLAM tended to under-estimate the speed of landmarks when only a few observations were used to calculate the speed. This is better suited to quasi-static environments than the over-estimation that is typical of MBD-SLAM. In exchange for this improvement of velocity estimation, a characteristic speed of landmarks in the environment is needed. As this prior knowledge is unlikely to be known absolutely, two boundary cases were chosen to assess the impact of imprecisely specifying the landmark characteristic speed: Case 3 in which motion is faster than expected, and Case 4 in which motion is slower than expected).

Performance in boundary cases

Simulation cases showed that for two boundary conditions, firstly an anomalously fast landmark and secondly a fully static case, the SLAM estimate was not degraded, instead, only the accuracy of the landmark velocity estimates was impacted. These two cases were ones that challenged the MBD-SLAM proposed in Chapter 5, and bounded the design space for the landmark KMM velocity. These boundary cases show that if prior knowledge of the expected landmark velocities is available, it should be incorporated into the variance measure of the QSA, but even if that prior measure is imprecise, localization and mapping accuracies comparable to the baseline static SLAM can be achieved with MBQS-SLAM.

The quasi-static pose graph SLAM approach proposed here was finally applied to a mission with two passes through the environment (a multi-episode mission) in Case 5. This case demonstrated that, after the second pass, the trajectory, map and velocity estimates were improved as more observations were made over the extended mission.

Since much of the improvement is attributed to the longer time elapsed between observations, it is expected that a single pass with longer length would also show a similar velocity estimation accuracy improvement. However, longer pass lengths mean a longer duration relying on dead-reckoning during the outbound portion of the pass before the loop closures during the return trip, and, like all SLAM, poses estimated with MBQS-SLAM would accrue more uncertainty during this period. Improving the UAV's orientation accuracy (ie. compass and gyroscope) or identifying known-static landmarks along the route would reduce the uncertainty growth rate.

6.6.1 Concluding Remarks

In conclusion, the proposed MBQS-SLAM algorithm enabled a UAV to remain on-station, persistently mapping a quasi-static environment. It was able to maintain an accurate map and trajectory estimate despite an anomalously fast landmark, and performed as well as the baseline static SLAM in static environments. **Given these results, MBQS-SLAM can be recommended for quasi-static environments, when a characteristic landmark velocity is available, even if the environment includes landmarks that are much faster or slower than the expectation.** This ability to accurately localize the UAV in a quasi-static environment cannot be achieved through traditional static SLAM or existing dynamic SLAM algorithms. The next chapter explores the event detection capability that will enable the UAV to apply MBQS-SLAM during epochs in a general piecewise-deterministic dynamic environment, without the KMM of a previous epoch corrupting the estimate in the current epoch.

Chapter 7 Detection of Disruptions to Environmental Model

The proposed MBDFs and MBQSFs were shown to be effective for SLAM in environments where the landmark KMMs have constant (though unknown a priori) parameters. However, general dynamic environments do not follow this constant constraint. UAV performance in general dynamic environments relies on correct modelling as the inclusion of a factor based on an incorrect KMM – whether a static assumption that is incorrectly applied to a mobile landmark, or a landmark's KMM parameters have evolved – can cause a graph to diverge.

Piecewise-deterministic KMMs are better descriptions of landmark motion in general dynamic environments, such as marine Arctic environments, and are characterized by a series epochs, each with a deterministic KMM, disturbed by events. Events are disruptions that underlie changes in the environment, which subsequently change a landmark's KMM. Typically, events cannot be directly observed, but instead are inferred from the unanticipated changes to the landmark position. Events will be treated as instantaneous discontinuities in a landmark's path since the duration of an event is typically much shorter than the duration of the mission. Since epochs are characterized by constant velocity KMMs, events change either (1) speed (magnitude), (2) direction of motion, or (3) both. The events studied in this chapter will be of all three types and will have different magnitudes.

To apply piecewise-deterministic KMMs, it is necessary to detect disruptions to the KMM that are estimated using MBQS-SLAM in quasi-static environments. This chapter will describe the strategy for event detection (Section 7.1), propose the method for detecting disruptions (events) in a piecewise-deterministic environment (Section 7.2) and test the proposed approach's ability to detect a variety of events in idealized conditions (Section 7.3). Finally, Section 7.4 discuses the performance of the proposed event detection algorithm as an enabler for SLAM in piecewise-deterministic quasi-static environments that will be discussed in Chapter 8.

7.1 Proposed Event Detection Strategies

As already noted, whether applied in static or dynamic environments, constructing the factor graph is a critical process for graph SLAM. Incorrect factors can cause a graph to diverge or result in solutions that differ from the ground truth.

It is common for incorrect factors to arise due to incorrect data association – when an observed landmark is mistakenly associated (matched) to a previously observed landmark during the SLAM loop closure step. This is often the case in static SLAM. Meanwhile, in piecewise-deterministic environments, it may arise when a landmark's KMM has changed due to a disruption and an observation after this disruption is incorrectly associated with an observation prior to the disruption. In the static SLAM case, the goal is to remove the incorrect factors. Meanwhile in dynamic SLAM postulated for a piecewise-deterministic environment as it is here, the goal is to detect the event, disconnect the newest observation and use it to begin learning the new KMM parameters. In this sense, event detection is the process of recognizing that a new landmark observation is inconsistent with the previous KMM based on the measurement and model variances. In this thesis, data association challenges from mis-identifying landmarks will not be addressed since landmarks are uniquely identified by AprilTags. Rather, this chapter will address data association challenges due to events.

7.1.1 Postulated Loop Closure Factors

When a previously observed landmark is observed again, new loop-closures are postulated as shown in Figure 7.1. This figure shows a simple case where loop closures are only drawn for a single landmark in a piecewise-deterministic environment. The landmark is observed at consecutive UAV poses 3, 4, 5 and 6, then not observed again until pose 12. The observations at poses 3, 4, 5 and 6 are associated with the same landmark and KMM (i.e. all are within one epoch). At pose 12, the landmark is observed again, and new loop closure factors are postulated that connect to each pose that previously observed that landmark.

Figure 7.2 shows the resulting graph two poses later. If the observation made at pose 12 were determined to be from the same epoch as the observations made at poses 3, 4, 5 and 6, those loop closure factors should become part of the graph (as shown in (a)), however, if an event occurred during the time the UAV travelled from pose 6 to pose 12, the postulated loop closure factors should not be added (as shown in (b)).

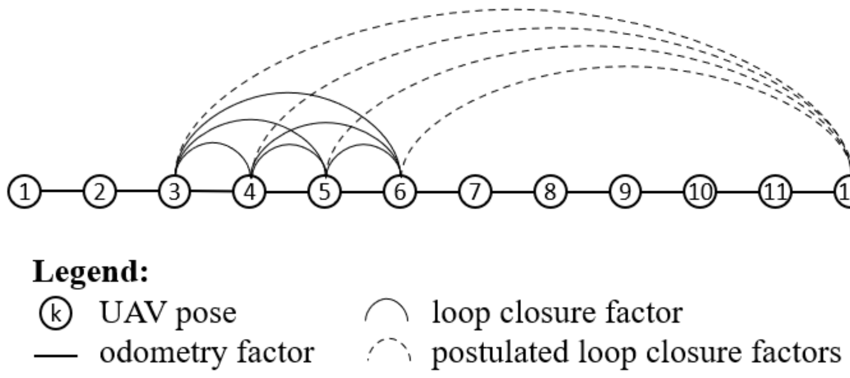


Figure 7.1: Factor graph constructed for a piecewise-deterministic environment, before postulated factors have been evaluated to detect an event. Figure shows only dead-reckoning factors and loop closure factors for a single landmark for clarity.

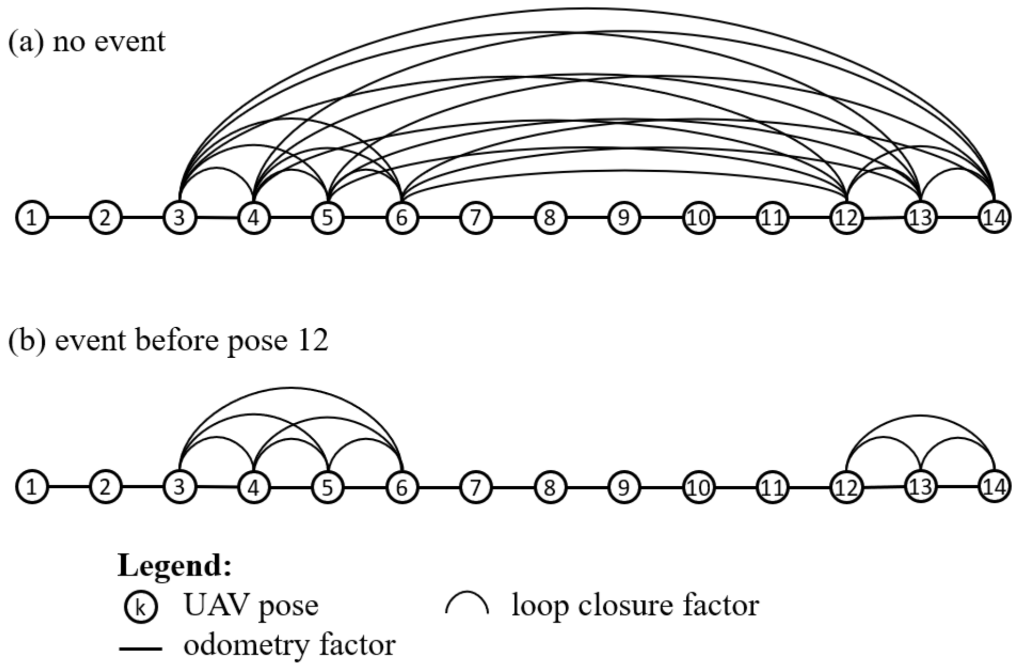


Figure 7.2: Factor graph constructed for a piecewise-deterministic environment in two cases: (a) no event detected and (b) event detected. Figure shows only dead-reckoning factors and loop closure factors for a single landmark for clarity.

Impact of False Positive Detections or False Negative Misses

Figure 7.2 shows how important the event detection decision is to a successful SLAM estimate. Even with only seven observations of the landmark, there are a total of 21 loop closure factors constructed, twelve of which would be incorrect if an event had occurred. Removing those factors unnecessarily means the dead-reckoning drift that occurs between poses 6 and 12 cannot be corrected through the loop closures, while keeping those factors when they are not connected to the same epoch for the landmark means more than half of that landmark's factors are driving the velocity estimate away from the correct value. Assuming persistent operations where there will be many observations of many landmarks over a long period of time, it is preferable to have a false positive detection (missing the gains of a loop closure) than to have a false negative (incorrect loop closure factors).

There are ways to assess whether a new measurement is consistent with the established model, as reviewed in Section 2.3. Of particular interest are the approaches of structure learning, which find erroneous factors by examining the graph structure itself. This method is computationally expensive so a pre-filter is proposed. This section will describe the application of *score-based structure learning* (SBSL), before the complete event detection approach is described in Section 7.2.

7.1.2 Score-Based Structure Learning (SBSL)

SBSL uses the information in the graph structure, to detect anomalous measurements. The error of a graph is defined as the average error of a graph's edges, where the error of an edge is defined as the difference between the measurement and the graph-optimized value estimated for that edge. This graph error is minimized during the optimization step (in this thesis, the iSAM back-end is used), and typically as more loop closure factors are incrementally added to the graph, the graph error will decrease or stabilize. In SBSL, the graph error can be used to score the graph. A small score is best, as it means the optimized poses have a better fit to the measurements.

Evolution of graph error in MBD- and MBQS-SLAM

As MBD-SLAM and MBQS-SLAM operate within a single epoch, each new landmark observation may be noisy, but the new observations are always of the same KMM. More observations reduce the uncertainty in the velocity measurement (as seen in the

WLS convergence study in Section 6.2), and improves the velocity measurement (as seen when comparing the two-pass mission in Section 6.5.5 to the one-pass mission in Section 6.5.1). As loop closure factors are added, and drift in the dead-reckoning factors is corrected, the values optimized through iSAM become a better fit for the set of all of measurements, and the graph error subsequently decreases.

However, in the case of adding a new observation after an event, this new observation will be inconsistent with the previously constructed factors moving the optimized poses and velocities away from the previous best fit and increasing the error on all of those factors. This means the graph error before adding the new observation will be measurably lower (better) than the graph error after adding the new observation.

SBSL uses this notable increase in graph error to indicate the newly added observation does not belong to the model that the rest of the graph represents (in this case the landmark KMM) and, thus, the factor(s) constructed using that observation should not be included in the graph. SBSL does not rely on the graph error to smoothly decrease when no event occurs, but only looks at how the error changes from adding one or more edges related to an observation. This means SBSL is well suited for complex dynamic environments where the graph error may fluctuate as a result of noisy measurements. Calculating the graph error is computationally expensive, but is a definitive way to detect whether an event occurred between observations. A pre-filter that identified potential events to then be scored would be more computationally efficient.

7.1.3 KMM Convergence Pre-Filter Design

In Chapter 6, the convergence of the WLS velocity estimate was demonstrated. Once this estimate converged, new observations of the landmark position did not cause large (compared to σ_{WLS}) changes in the velocity estimate for the landmark. As such, a heuristic threshold ε on the variation of the estimate in response to a new observation is a useful pre-filter for events. This pre-filter would identify potential events when either:

$$\Delta\hat{v} \geq \varepsilon_v, \tag{7.1}$$

or,

$$\Delta\hat{\omega} \geq \varepsilon_\omega, \tag{7.2}$$

where $\Delta\hat{\upsilon}$ and $\Delta\hat{\omega}$ are the changes in translational and angular velocity estimates respectively that result from incorporating the new observation in the WLS regression, and ε_υ and ε_ω are the pre-filter thresholds for translational and angular velocities, respectively.

This threshold provides an opportunity to balance the competing needs to (a) minimize the computational time (fewer potential events identified) and (b) catch events as they occur. A larger threshold means more potential events are detected for the SBSL event detector to assess, while a smaller threshold means events of small magnitude may not be detected, or may be detected late. As it is preferable to have a false-positive event detection than to miss an event, future work should be undertaken to optimize the computation time of the error calculation, allowing the increase of the threshold (or the removal of the pre-filter altogether) to bias the pre-filter towards false-positive detections.

Threshold selection: In the case of a quasi-static environment, there already exists a measure of the characteristic velocity (υ_k) for a landmark that was used to construct σ_{QSA} in the QSA. Using this velocity to design the threshold is appropriate. Simulation and experimental experience suggests that a threshold of $\varepsilon = \upsilon_k/3$ creates a filter that is sensitive to the types and magnitudes of events studied in this thesis. When the threshold is larger (such as $\varepsilon_\upsilon = \upsilon_k$), several false events were postulated and the cycle time became too long to maintain waypoint-tracking. When the threshold was smaller (such as $\varepsilon = \sigma_{WLS}$) many events were missed by the pre-filter.

7.2 Proposed Event Detection Algorithm

The proposed event detection algorithm is described in this section. It does not address SLAM or the method of estimating the landmark velocity, as those elements remain the same as in the MBQS-SLAM proposed in Chapter 6. The proposed approach uses a series of checks presented in Figure 7.3 to determine if an event has occurred.

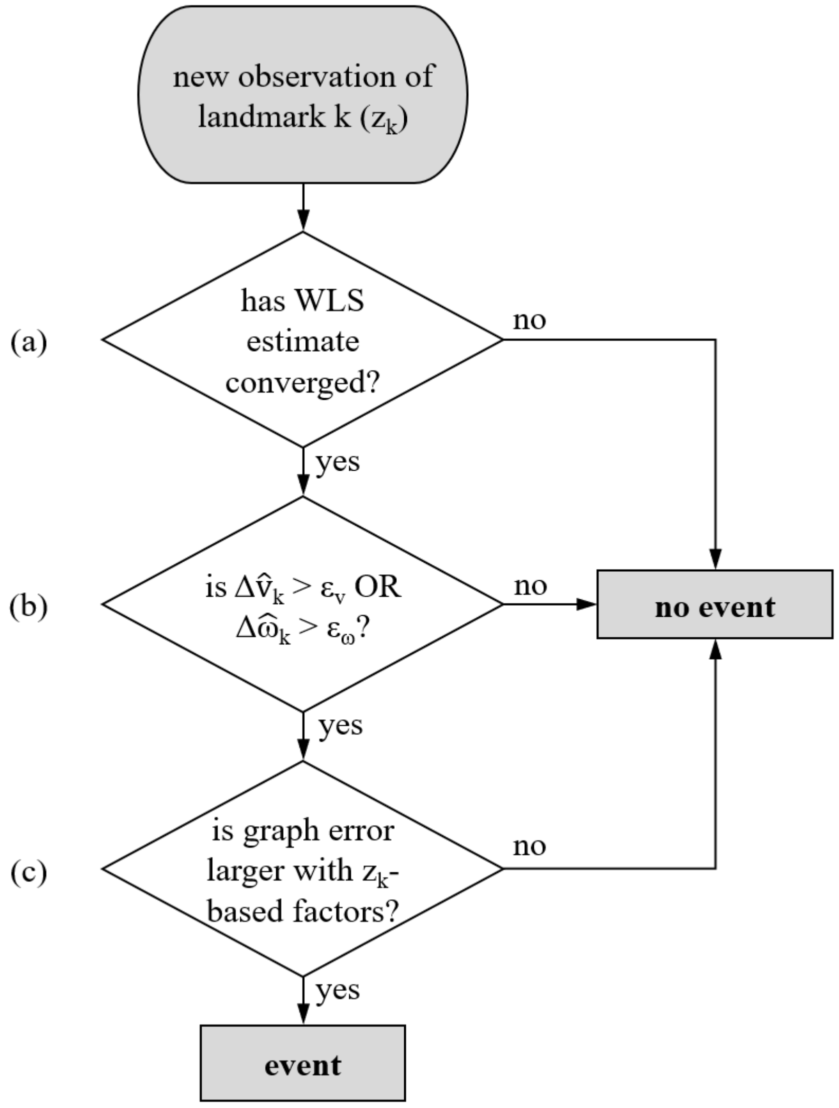

new observation of landmark k (z_k)
(a)
has WLS estimate converged?
no
yes
(b)
is $\Delta\hat{v}_k > \varepsilon_v$ OR $\Delta\hat{\omega}_k > \varepsilon_\omega$?
no
no event
yes
(c)
is graph error larger with z_k-based factors?
no
yes
event

Figure 7.3: Proposed event detection algorithm, consisting of three checks.

When a new observation (z_k) is made of landmark k, it is added to the set of all observations of landmark k. The first check that is performed (labelled (a) in the Figure 7.3) determines if the WLS estimate has converged. Recall that MBQS-SLAM assumes the landmark is static until the WLS velocity estimate converges (called the QSA) and as such, it is not possible to detect events before the WLS velocity estimate converges.

The next check (b) examines how the translational and angular velocity estimates ($\hat{v}_k$ and $\hat{\omega}_k$), respectively, change when z_k is added. This check functions as a pre-filter for event detection, costing little computation-wise, but potentially providing false positives. If the change in the estimates is small, the new observation is considered consistent with the current KMM, meaning no event has occurred, otherwise, the SBSL approach to event detection is applied.

The final check (c) consists of several steps: (1) the graph score without the postulated loop closures formed based on z_k is calculated; (2) the z_k-based factors are added to the graph; (3) the graph is optimized, and (4) the graph error is calculated. If the graph score without the loop closures (1) is larger than the graph score with the loop closures (4), the new loop closures are considered consistent with the current KMM, no event has occurred, and the standard MBQS-SLAM algorithm can continue. Otherwise, an event is detected.

Once the event has been detected, the postulated factors based on z_k must be removed, returning the graph to the state it was in at step (1). It is also necessary to appropriately track that the landmark has experienced an event causing it to transition to a new epoch. This means that all observations made in the previous epoch should be archived, and a new KMM should be started with z_k as its first observation. Starting a new KMM means returning to using the QSA until the WLS velocity estimate converges, treating future observations as part of the new KMM, and not constructing factors across the event.

7.3 Results

Unlike previous chapters where localization and mapping accuracy as well as velocity estimation accuracy were performance metrics, this chapter is interested in whether events are correctly detected, if they are detected at the correct time (before incorporating the first observation after the event) and if any false-positive events are detected by the proposed algorithm. As such, the simulation environment and cases defined in Chapter 4 are not used. Instead, a *sandbox* is developed with idealized conditions to focus on the event detection rather than the SLAM.

This section begins by defining the sandbox environment then uses that sandbox to test the detection success for different types and magnitudes of events.

7.3.1 Sandbox Environment

To assess the performance of the proposed event detection approach, many different events are tested in a sandbox environment. The sandbox environment uses the same basic structure as in Chapter 6 with 19 static and two mobile landmarks in the configuration of Figure 4.10 over a 100 m length, with only the first landmark known a priori to be static. Unlike the tests in the previous chapters, in the sandbox there is no added corrupting noise to either the tag detection position estimates or the odometry and altitude measurements. In this way, the SLAM inputs are idealized compared to Chapter 6. This means event detection limitations cannot be attributed to noisy observations, and repeat trials of the same set of parameters yield the same results (completely deterministic).

Task: The selected task is a two-pass mission as in Section 6.5.5, where in each pass the quadrotor nominally flies along the X-axis and returns to the start position while mapping the landmarks and localizing itself with MBQS-SLAM in the process. The update rate for the MBQS-SLAM algorithm is 0.5 Hz, and landmarks 3 and 4 are both mobile at 20 and 10 km/day, respectively (quasi-static).

Event characteristics: Events have different types and magnitudes. For this study, the type of an event will be limited to (1) changes in speed, (2) changes in direction, or (3) both. Due to the layout of the map and the quadrotor's FOV, to ensure the landmark is visible after the event, only one magnitude of direction change will be studied: a moderate turn to port. Speed changes will be either increasing or decreasing between three possible settings: static, quasi-static (0.1 m/s (20 km/day)), and slow (0.2 m/s (40 km/day)).

Event insertion: Events are scheduled to occur between passes while the quadrotor pauses (5 seconds duration) at the start position. First, a set of tests will insert an event for one landmark (landmark 3) only, while the other mobile landmark (landmark 4) remains in a single epoch. Next, an additional test will be performed where both landmark 3 and 4 undergo different events simultaneously.

7.3.2 Detection of One Event for One Landmark

The set of all events studied for the case of one landmark experiencing one event are presented in Table 7.1. It shows the pre-event KMM (heading θ^0 and speed v^0)

of landmark 3 on the left hand side and the post-event KMM (θ^1 and v^1) on the top. Cells of the table that are shaded in gray with the label 'N/A' are arrangements that either do not result in an event (ie. the KMM parameters do not change), or they change in a way that is redundant. For example, if the quadrotor began at 0.1 m/s, then stopped moving, a rotation just before the stop would not change the detectability of the event. Colours of the other cells indicate the success of the event detection. Green cells are events that are detected at the correct time, yellow cells are events that are detected but not at the correct time and red cells are events that are not even detected. The number in the cell is the number of false positive events detected after the entire algorithm in Figure 7.3 is applied.

Table 7.1: Performance summary of event detection for landmark 3. At left is the pre-event KMM, on top is the post-event KMM. Cell colours indicate level of success (green: detected at the correct time, yellow: detected late, red: not detected) and numbers indicate any false-positive detections.

		θ^1 [rad]		-0.13			0.5	
		v^1 [m/s]	0	0.1	0.2	0	0.1	0.2
θ^0 [rad]	v^0 [m/s]							
	0		N/A	🟨 0	🟨 0	N/A	N/A	N/A
-0.13	0.1		🟨 0	N/A	🟨 0	N/A	🟩 0	🟩 0
	0.2		🟨 0	🟥 0	N/A	N/A	🟩 0	🟩 0

Table 7.1 shows that the proposed event detection algorithm in Figure 7.3 performed perfectly for events that altered the landmark direction. It was also able to detect events that changed only landmark speed but detected these events late (yellow). These events were detected on the first observation of the landmark as the UAV returned to its start point in the second pass. The one event that was not detected (red), was a reduction in speed from 40 km/day to 20 km/day. No false detections were reported in these tests. The proposed event detection algorithm was able to detect nine out of the ten events tested. The late detections for speed-change only events are acceptable given the small magnitudes in speed changes and the short duration between passes.

Detection of One Event Discussion

Direction changes are easier types to detect because the new landmark observation has both an unexpected position as well as an unexpected orientation. This combination means that the loop closure factor error is larger and has a strong impact on the graph error after a single observation.

When only the speed changes, the landmark needs time to transit far enough from the its expected position for the error to be significant relative to the estimated covariance. The observation that is captured when the UAV is returning to its start position the second time allows the landmark a longer interval to transit than the observation that is captured when the UAV travels away from its start position for the second time. A longer pause between passes would mean the event would have a larger impact on the magnitude of the position deviation, making the event more likely to be detected on-time.

A delayed event detection means the observations that are captured between the true event and the detected event are associated with an incorrect KMM, influencing the estimated velocity for the first epoch towards the true velocity of the second epoch (which is undesirable). If the event magnitude is small, this can cause an event to be missed entirely and the velocity to be estimated as between the values for the two epochs. The missed case in this test was a change in speed from 0.2 m/s to 0.1 m/s. This case was particularly challenging for the event detection as the MBQS-SLAM algorithm notably under-estimates slow speeds to begin with. As the first epoch's KMM would be under-estimated, when the second epoch began, the smaller difference in speed combined with the tendency to detect speed-only events late, resulted in a missed event.

In a persistent mapping scenario, the time between passes and the number of passes would both be much larger, increasing the likelihood of timely event detection. Persistent mapping scenarios also typically result in more observations within an epoch. More observations reduces the uncertainty on the WLS velocity estimate (σ_{WLS}). If an event occurs for a landmark with a smaller σ_{WLS}, the amount by which the landmark's position must deviate from the expected position before its error is noticed is smaller, thus making timely event detection more likely.

In this analysis a core assumption is that an appropriate KMM is selected for the environment, such that landmarks spend the majority of their time following a KMM with infrequent disruptions. In other words, these simulations only test the event detection algorithm for cases when the landmarks are truly moving with a constant-velocity for nearly the entire mission, with the except of one or only a small number of disruptions (events). This means the characteristic duration of an epoch is assumed to be longer than the time required for the WLS estimate of KMM parameters to converge, and much longer than the time between consecutive observations and pose updates. If that is not the case, then the KMM should be changed to acknowledge that the best description of landmark motion is not constant-velocity. As an example in the Canadian marine Arctic environment, if the region of interest was a narrow straight, where the ice masses used as landmarks were frequently colliding, a constant-velocity KMM would be inappropriate, as there would be events happening at high frequency. These high frequency events defeat the purpose of attempting to estimate the KMM parameters of any particular epoch.

7.3.3 Detection of Simultaneous Events

In addition to testing if different magnitudes and types of events could be detected, it is important that the algorithm can detect events for multiple (ex. two) landmarks at similar times. This is representative of a case where a weather system moves through the marine Arctic environment, simultaneously influencing the motion of all ice masses in the active season.

This test used the same sandbox with the pre- and post-event KMM parameters given in Table 7.2. Landmark 3 experiences an event that changes only its heading, while landmark 4 experiences an event that changes only its speed.

Table 7.2: Parameters of KMMs before (θ^0 , v^0) and after (θ^1 , v^1) the events for landmarks 3 and 4.

Landmark ID	θ^0 [rad]	v^0 [m/s]	θ^1 [rad]	v^1 [m/s]
3	-0.13	0.1	0.5	0.1
4	0.2	0.1	0.2	0.2

The proposed event detection algorithm successfully detected both events on the first observation after the event occurred (i.e. on-time). The detection of the event for landmark 3 is not surprising given the results in Table 7.1. However the detection of the event for landmark 4 at the correct time is notable, as a similar event was detected late (Table 7.1). Landmark 4 is located further from the start position than landmark 3, meaning that the UAV requires more transit time before the first observation of landmark 4 is made after the event. This means the landmark's position has deviated further from the expected position, increasing both σ_{WLS} and the graph error by more than the WLS velocity estimate confidence permits, triggering the event's detection.

The algorithm did not struggle to identify the simultaneous events as each landmark is treated independently for event detection. However, the execution time for the cycles when the SBSL score calculation is performed does take longer, and if landmarks 3 and 4 had been observed in the same frame, that pose update cycle would have exceeded the nominal 2 seconds allotted (the pose update rate is 0.5 Hz). Scenarios where there are many mobile landmarks all of whose KMM's are affected by the same event could mean that the score calculation needs to be performed many times within a cycle. This would drastically reduce the frequency of pose update to the UAV waypoint tracking controller, possibly to the point where it becomes unstable. Managing the processing power required for this algorithm is a key task for future persistent applications.

7.4 Discussion

This chapter proposed an algorithm for detecting events in a piecewise-deterministic, quasi-static environment as a step towards enabling a UAV to operate persistently in marine Arctic environments. While the previous chapters addressed estimation of a landmark's KMM in slow and quasi-static environments, this chapter addressed the problem of the events that inevitably disrupt those KMMs as the time on station increases. Timely detection of these events is essential to avoiding incorrect loop closure factors that can cause the SLAM estimate to diverge.

The event detection algorithm proposed in this chapter is based on the concept that a graph's structure itself contains information that can be used to identify factors that are inconsistent with the underlying model that the graph represents. Following methods of SBSL, the graph error is calculated before and after a new observation is incorporated to determine if that observation is consistent with its landmark's KMM, or if it indicates an event has occurred. To compensate for the high computational expense of calculating the graph error, a pre-filter is used to identify potential events by comparing the change in the velocity estimate resulting from the new observation to a heuristic threshold. While the SBSL detector requires no prior knowledge or pre-selected thresholds, the pre-filter is designed based on the characteristic velocity of landmarks that is already required for MBQS-SLAM. This threshold puts a limitation on the flexibility of the event detector to uncertain environments, but also presents an opportunity to manage the computational load. Improving the efficiency of the SBSL implementation would allow the pre-filter threshold to be loosened or event removed, decreasing the reliance on prior knowledge of the characteristic velocity.

The event detector was first tested on 10 events spanning different types and magnitudes, and then tested with two simultaneous events (one event for each of two landmarks). While the event detector did not always detect the event at the correct time, these tests revealed important considerations to event detection: the time between the event and the observation of the landmark, and the confidence in the WLS velocity estimate before the event occurs. The time element determines the amount of deviation between the expected landmark position (under the KMM of the previous epoch) and the observed landmark position (due to the KMM of the new epoch). The longer the time the landmark transits under the new KMM, the more likely the event detector is to be successful, assuming a subsequent event does not disrupt that KMM. In a similar way, higher confidence in the WLS velocity estimate means a smaller deviation between the expected and observed landmark positions will trigger detection of an event, as the graph error is calculated with consideration for the uncertainty in the estimated velocity.

In Chapter 6, SLAM in an epoch characterized by quasi-static motion was developed. **This chapter added the capability to detect events that bound the epochs to quasi-static motion.** To fully address operations in a piecewise-deterministic quasi-static environment, these two capabilities need to consider the behaviour of a landmark as it evolves through several epochs and events. This is the focus of the next chapter.

Chapter 8 Proposed Piecewise-Deterministic Quasi-Static SLAM

All of the necessary components for SLAM in a piecewise-deterministic, quasi-static environment have been presented in the previous chapters. This chapter combines those components into a complete algorithm to estimate the deterministic KMM of mobile landmarks in the environment during an epoch using the MBQS-SLAM, and to detect and recover from disruptions (events) to the KMM using the event detection algorithm in Chapter 7. The new approach proposed in this chapter is *piecewise-deterministic quasi-static pose graph SLAM* (PDQS-SLAM).

The application of PDQS-SLAM is captured in Figure 8.1, where MBQS-SLAM is applicable between events (during epochs), while PDQS-SLAM is applicable both between and across events.

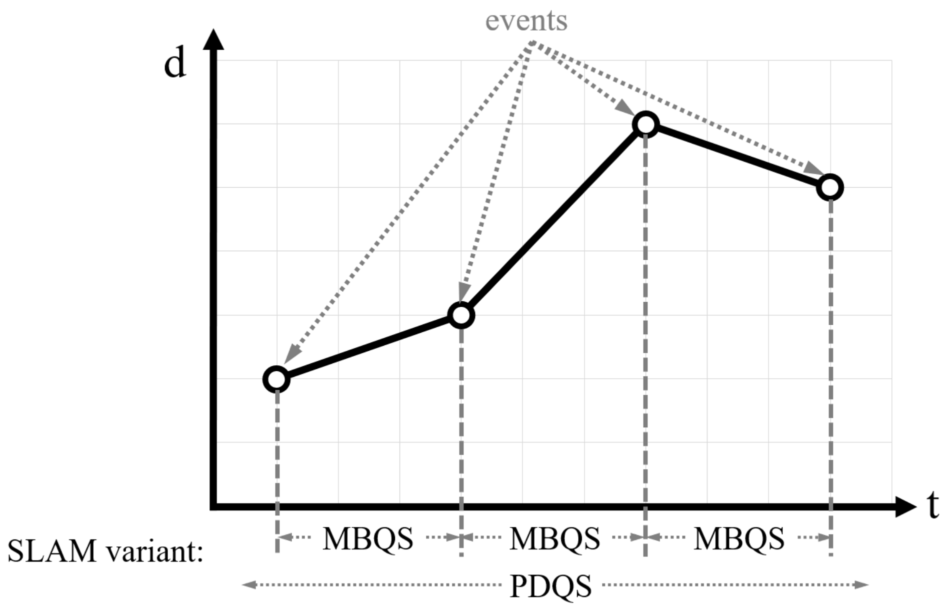


Figure 8.1: PDQS-SLAM is proposed to bring together the methods of MBQS-SLAM that is applicable to quasi-static environments within epochs, and event detection, to enable SLAM in a piecewise-deterministic environment.

PDQS-SLAM was designed in response to the concept that a landmark in a piecewise-deterministic environment can be described as existing in a stable, active state within an epoch, then undergoes a transition to a new stable state as a result of an event. Since the graph is an attempt to model the landmark's state, it can also be described by a series of states and transitions as it models the landmark's KMM. This inspired the development of an FSM to capture the evolution of the graph's confidence that an event has occurred. In designing the FSM, it became clear that the FSM was a property of the set of edges that are constructed from multiple observations of a common landmark, inspiring the addition of the state, as an attribute, to each loop closure edge in the graph.

This chapter first defines the proposed FSM and the augmented loop closure factors (Section 8.1), then presents the PDQS-SLAM algorithm. Next, PDQS-SLAM is tested in three cases:

- **Case 1:** two-pass, full-scale, quasi-static environment with one mobile landmark experiencing one event – simulation (Section 8.3.1)
  This case is similar to Case 5 for MBQS-SLAM in Section 6.5.5, with the exception that there was no event in Case 5 for MBQS-SLAM. This is the parent case for the PDQS-SLAM algorithm and shows the opportunities of applying PDQS-SLAM in complex piecewise-deterministic quasi-static environments.

- **Case 2:** two-pass, model-scale, quasi-static environment with one mobile landmark experiencing one event – simulation and experiment (Section 8.3.2)
  This is the same as Case 1 but at model-scale. It is performed both in simulation and experiment to assess the impact of actual sensors and mobile landmarks on the performance.

- **Case 3:** two-pass, full-scale, quasi-static environment with two mobile landmarks each experiencing one event – simulation (Section 8.3.3)
  The additional event in this case is selected to challenge the algorithm with two events as in the tests of the event detection algorithm in Chapter 7.

Finally, Section 8.4 summarizes the results and discusses the strengths and limitations of the PDQS-SLAM algorithm.

8.1 Novel Edge Finite State Machine

In a piecewise-deterministic environment, a landmark transitions between two deterministic KMM's as a result of an event. The event detection algorithm proposed in Chapter 7 attempts to model that transition, while the MBQS-SLAM algorithm proposed in Chapter 6 attempts to model the KMM's in the epochs before and after the event. Together these strategies indicate how the set of loop-closure edges associated with a particular landmark should behave. They provide transition points from initialization, to a state where the edge set is actively contributing to SLAM, through the state of the set potentially spanning an event, to a decision point where the event is detected and the KMM is divided.

Finite State Machines: An FSM models a system as a finite set of discrete states and transitions [85]. The behaviour of the system depends only on its state. In an FSM, transitions can consist of a triggering event, a condition that must be true for the transition to occur, or an action that occurs while in transition between states. FSMs are useful to capture a complex system's behaviours succinctly, for example Araujo et al. modelled a search and rescue UAV using an FSM [86].

Based on the event detection algorithm described in 7.3 and the MBQS-SLAM described in 6.6, the proposed FSM for each set of edges related to a landmark k, E_k, is shown in Figure 8.2.

The FSM defines six states for the set of edges (loop closure factors) related to landmark k in the current epoch e. The entry point to the FSM is the initialization of the empty set with the first observation of the landmark in the epoch.

1. initialized: In the initialized state, E_k^e is an empty set as there has only been a single observation of landmark k. In this state, SLAM variables are initialized but the velocity cannot be estimated. Transition from the initialized state is triggered when a second observation of landmark k is made, resulting in E_k^e containing one edge. From this state, E_k^e either transitions to being under the QSA, if the characteristic velocity of landmarks in the environment is below the threshold for quasi-static motion ($\upsilon_k \leq \upsilon_{QS}$), or directly to the active state if the environment is slow.

2. under QSA: While under the QSA, the MBQS-SLAM algorithm uses the QSA to assume the landmark is static. When the WLS regression converges, the transition is triggered to the active state.

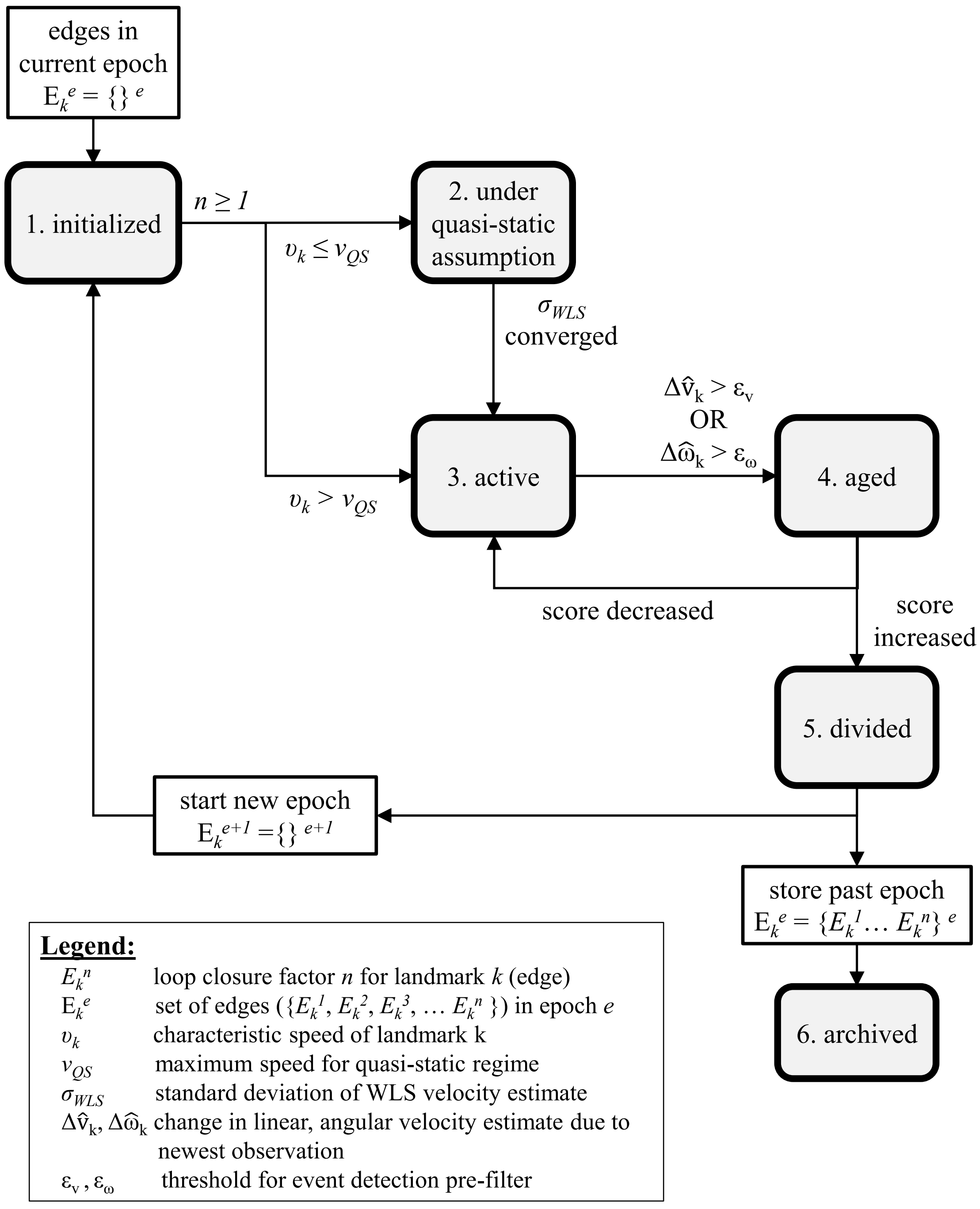

edges in current epoch $E_k^e = \{\}^e$
1. initialized
$n \geq 1$
$v_k \leq v_{QS}$
2. under quasi-static assumption
σ_{WLS} converged
$v_k > v_{QS}$
3. active
$\Delta\hat{v}_k > \varepsilon_v$ OR $\Delta\hat{\omega}_k > \varepsilon_\omega$
4. aged
score decreased
score increased
5. divided
start new epoch $E_k^{e+1} = \{\}^{e+1}$
store past epoch $E_k^e = \{E_k^1 \dots E_k^n\}^e$
6. archived
Legend:
E_k^n loop closure factor n for landmark k (edge)
E_k^e set of edges ($\{E_k^1, E_k^2, E_k^3, \dots E_k^n\}$) in epoch e
v_k characteristic speed of landmark k
v_{QS} maximum speed for quasi-static regime
σ_{WLS} standard deviation of WLS velocity estimate
$\Delta\hat{v}_k$, $\Delta\hat{\omega}_k$ change in linear, angular velocity estimate due to newest observation
ε_v, ε_ω threshold for event detection pre-filter

Figure 8.2: Proposed FSM governing state of the set of edges formed through loop closures of landmark k during epoch e (E_k^e).

3. active: The active state is the primary state for E_k^e. This is the state the set is in when MBQS-SLAM or MBD-SLAM (as driven by v_k) is applied to a single epoch as in Chapters 5 and 6. The event detection pre-filter (with thresholds ϵ_v and

ϵ_ω) triggers transition out of this state.

4. aged: The event detection approach in Chapter 7 describes a state for E_k^e where there is the potential that an event has occurred, but it has not been confirmed. This is the aged state. Using the event detection algorithm proposed in Chapter 7 keeps E_k^e in this state for less than one SLAM update cycle, with E_k^e transitioning out of this state when the graph error (score based on SBSL) confirms the potential event. The calculation of the score triggers the event. If the score increases (event detected), E_k^e moves to the divided state. If the score decreases (no event detected), E_k^e returns to the active state.

5. divided: The divided state performs the actions to recover from the detected event. In this state, the set of observations Z_k^e used to form the loop closures E_k^e spans across an event – the newest observation captured after the event and all other observations captured before the event. This means: (1) any loop closure edges formed relating to the newest observation must be removed from the graph; (2) the newest observation is assigned to the next epoch of the landmark; (3) E_k^e needs to be archived; and (4) a new set of edges for the next epoch, $\mathrm{E}_k^{(e+1)}$, is to be initialized as an empty set.

6. archived: The exit state for an edge is archived. Archiving this set of edges keeps them in the graph, but prevents future observations of the landmark from being associated with the observations from previous epochs.

FSM state as an attribute of edges

The current state of E_k defined through the FSM can be considered an attribute of each of the edges in the set E_k^e in a similar way to the attributes of the nodes proposed by DPG-SLAM [29]. The DPG-SLAM node attributes track the state of observations made at the pose to determine if that pose adds observations to the graph that are part of the current map state or from a previous state of the map. When the observations of a node are from a previous state, the node is removed, which simultaneously removes the incorrect observations and reduces the number of nodes in the graph. Assigning the state to the edges is a better representation of the landmark time evolution compared to assigning the state to the node which would represent the evolution of the map state as a single entity.

An augmented loop closure factor edge is recommended in the form of:

$$E_k^n =< T, \Sigma_T, p_k, \Sigma_k, s_k >, \tag{8.1}$$

where T and Σ_T are the standard geometrical constraint and covariance on that constraint. MBDFs and MBQSFs added the landmark's KMM parameters, p_k, and their covariance Σ_k to the edge definition. PDQS-SLAM adds the current state of the set of edges that this edge belongs to, s_k, which is governed by the FSM.

This addition of state attribute to the graph edges is a novel development in response to dynamic environments. In static SLAM, edges are constant once formed since the measurements used to construct the edge do not change (assuming perfect data association). The graph is constructed to estimate the map which is a static, unchanging structure. In dynamic SLAM however, the underlying map changes with time, meaning the graph edges should also reflect this and change with that map. Assigning state attributes do just that: they allow the edges to change as more knowledge of the motion in the environment is acquired.

8.2 Proposed Piecewise-Deterministic Quasi-Static SLAM Algorithm

The integration of MBQS-SLAM and event detection results in the proposed PDQS-SLAM algorithm shown in Figure 8.3. The quadrotor (including downward camera, altimeter and odometry), [1]AprilTag detection [28], velocity controller, dead-reckoning factors construction, [2] iSAM graph optimizer [9] and mapping blocks are all consistent with the baseline static SLAM (Figure 4.11). In lighter yellow, the landmark velocity estimation and MBQSF construction blocks are both from MBQS-SLAM (Figure 6.6). In darker yellow, the event detection pre-filter and the SBSL arbiter (compares graph errors before and after a new observation to detect an event) are both from the event detection algorithm (Figure 7.3). The dashed arrows indicate steps that are only performed if a potential event is detected by the pre-filter.

In PDQS-SLAM, after the WLS regression algorithm is applied to estimate the landmark velocity, the event detection pre-filter checks if the change in the velocity estimate, in response to the newest observation, is larger than the threshold ε. If so, the FSM sets the edge set to the aged state and the graph error is calculated by optimizing the graph (A) having added the new dead-reckoning factor, but no loop

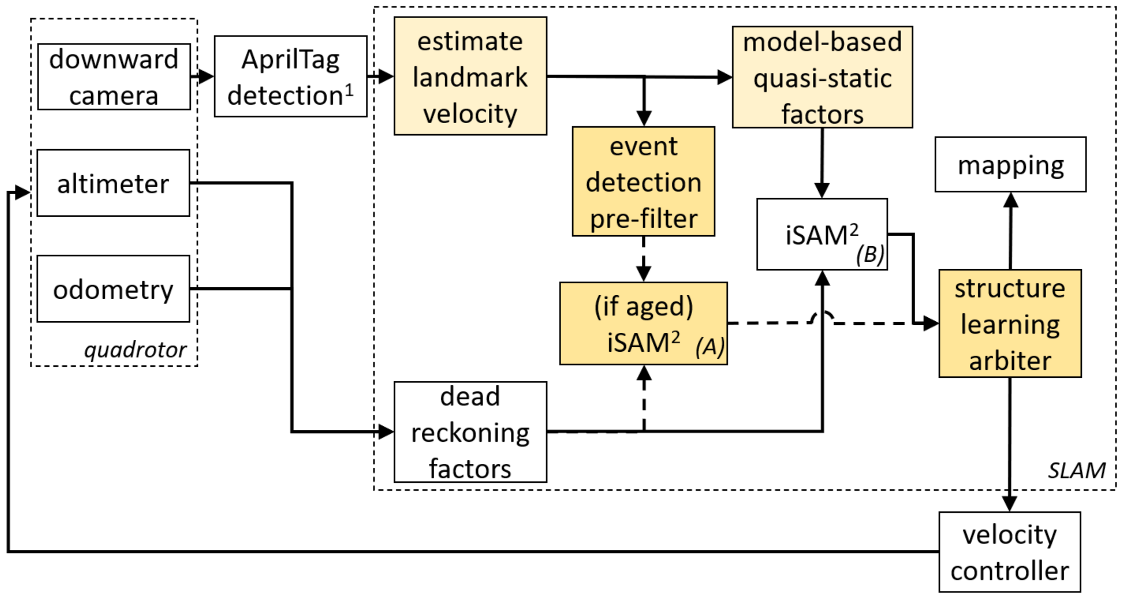


Figure 8.3: Proposed PDQS-SLAM algorithm. The yellow blocks indicate the new elements relative to static SLAM.

closure factors related to this newest observation. This graph error at (A) is compared with the graph error calculated by optimizing the graph (B) with the new loop closure factors added, by the SBSL arbiter. The pose estimate from the graph with the lowest error of (A) or (B), is used to map and generate the quadrotor's velocity input. If (A) is used, then an event has been detected, sending E_k^e to the divided state. On the other hand, if (B) was invoked, no event is detected returning E_k^e to the active state. If the pre-filter does not detect a potential event (dashed lines removed), then iSAM is not applied at (A), meaning the arbiter selects the iSAM estimate generated at (B). This is the same outcome as if the MBQS-SLAM algorithm had been applied.

8.3 Results

Three cases, identified in Table 4.6 are selected to demonstrate PDQS-SLAM performance. The cases in this chapter are two-pass missions with one or two events occurring between passes. This event timing ensures that the UAV does not witness the event happening and that there is enough time for the first WLS estimate to converge before the event. The three illustrative cases are:

- **Case 1:** two-pass, full-scale, piecewise-deterministic environment with one

event – simulation (8.3.1)
This represents the parent case for which PDQS-SLAM has been designed and shows successful localization and mapping despite events disrupting the KMM used in MBQS-SLAM. Landmark 3 experiences an event that changes its speed and its heading (20 km/day at -0.13 rad to 40 km/day at 0.5 rad), while landmark 4 does not experience an event (10 km/day at 0.2 rad).

- **Case 2:** two-pass, model-scale, piecewise-deterministic environment with one event – simulation and experiment (8.3.2)
  Geometrically similar to Case 1, this case determines if or how PDQS-SLAM differs in performance in simulations compared to experiments.

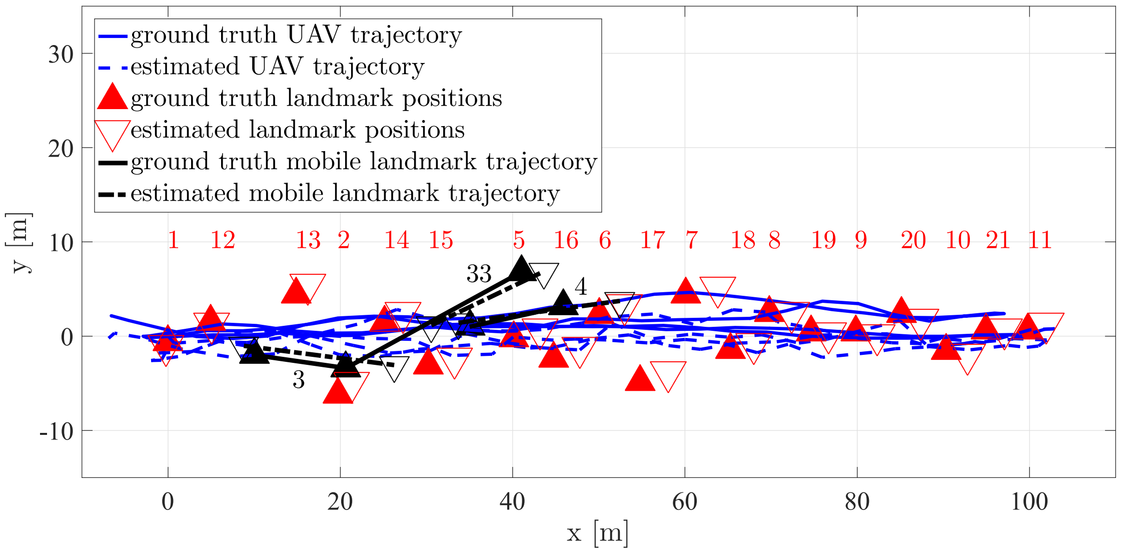
- **Case 3:** two-pass, full-scale, piecewise-deterministic environment with two events – simulation (8.3.3)
  In Case 3, both landmarks experience an event. Landmark 3 experiences a change in heading only (20 km/day at -0.13 rad to 20 km/day at 0.5 rad), while landmark 4 experiences a change in speed only (20 km/day at 0.2 rad to 40 km/day at 0.2 rad). This is a challenging case identified in Section 7.3.3 as the event applied to landmark 4 was detected late in the one event sandbox experiment. This could result in the speed of landmark 4 being poorly estimated and the KMM after the event not being observed over a long enough time span for the WLS estimate to converge in the second epoch.

While Chapter 7 provided insight on event detection success in an idealized (noise-free) world, these three cases will provide insight into the impact of anticipated corrupting noise in the odometry and tag position estimates on the event detection success, and on the ability to successfully estimate the KMM parameters for a landmark after an event. As in Chapter 6, success is measured by the localization and mapping accuracy relative to the baseline static SLAM applied to a static environment, and as well by the accuracy of the velocity estimate for each landmark. In addition, successful event detection means all events are detected, at the right time, without false-positive event detections (events that are detected but did not actually happen).

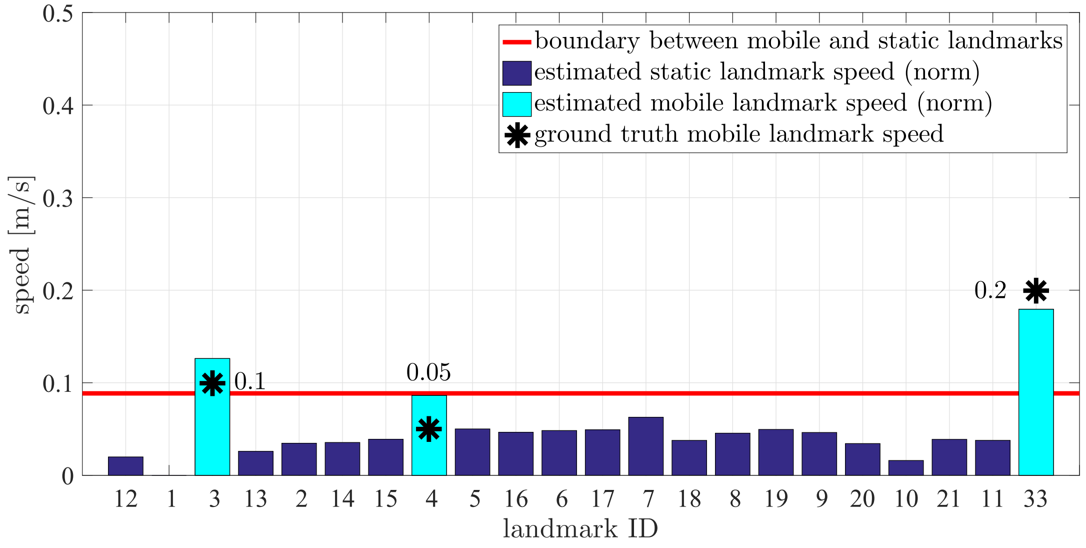
8.3.1 Case 1: Full-Scale in Simulation

The proposed PDQS-SLAM algorithm is tested at full-scale in the quasi-static environment for a two-pass mission with one event impacting landmark 3. This case is summarized in Table 8.1 and the median-performance trial is shown in Figure 8.4.

Case 1 Localization and Mapping Results



(a) planar map



(b) speed estimates

Figure 8.4: Case 1 simulation of PDQS-SLAM in full-scale, piecewise-deterministic environment with one event.

Table 8.1: Case 1 (two-pass, full-scale, one event) simulation performance summary for PDQS-SLAM (Figure 8.4).

metric	**unit**	**simulation**
localization error (norm)	[m]	5.38 ± 1.69
x error	[m]	2.56 ± 0.56
y error	[m]	2.30 ± 0.83
z error	[m]	3.20 ± 2.02
mapping error (norm)	[m]	4.61 ± 1.91
x error	[m]	1.52 ± 0.91
y error	[m]	1.27 ± 1.09
z error	[m]	3.54 ± 2.32
mean static landmark speed error	[m/s]	0.0375 ± 0.0110
mean dynamic landmark speed error	[m/s]	0.0988 ± 0.1279
mean speed boundary	[m/s]	0.1536 ± 0.0998

Figure 8.4a shows the planar map of the trajectory and landmark positions for the median trial of Case 1. Note that landmark 3 in epoch 2 is labelled as landmark 33 for clarity. This figure shows that the along-track landmark positions are over-estimated, with the along-track error larger for landmarks further from the start position. The bar chart of the speeds shows that all three KMMs (landmark 3 before the event, landmark 3 after the event, and landmark 4) converged by the end of the second pass. It also indicates that the choice of boundary is less meaningful when there are a mix of quasi static (10-20 km/day) KMMs and slow (40 km/day) KMMs as the boundary is skewed high by the slow landmark. This chapter will continue to report that metric for consistency, but the focus will be on the localization and mapping accuracies and event detection success.

The localization error over the 10 trials was on average 5.4% of the 100 m pass length, while the mapping error was 4.6% of the pass length. The average speed estimated for a static landmark was 7.5 km/day, which is slightly higher than the two-pass, no event mission in Section 6.5.5 (6 km/day).

Case 1 Discussion

Unlike SLAM within an epoch, the map shows that landmarks in the middle of the pass – close to the landmark that experiences an event – have the largest error. This is because the KMM of landmark 3 after the event does not converge until the quadrotor travels back over landmark 3 during the return to the start position of the second pass. This motion uncertainty subsequently increases the pose uncertainty near landmark 3 which translates to uncertainty in the map of nearby landmarks. Just as the landmark positions furthest from the start position become more accurate in a second pass (when they are observed after their motion model converges), so should the poses near landmark 3 in a third pass (assuming another event did not disrupt the 'new' KMM).

In all 10 repeat trials, PDQS-SLAM was able to detect the event to landmark 3 at the correct time (i.e. the first observation after the event occurred). However, the KMM for landmark 3 after the event did not always converge. This is apparent in the large mean error on the mobile landmark speed (0.0988 m/s). In the cases that the WLS velocity estimate did not converge, it is because fewer observations were made of the landmark during the return to the start position on the second pass. This happens because either: (1) the UAV trajectory deviates from the x-axis to a position where the AprilTag is not visible; (2) the UAV travels slower or faster than nominal so the AprilTag is outside the FOV when the UAV passes over it, or (3) the computation time is longer than the nominal 2 s due to potential event detection(s), causing the SLAM to skip an observation.

Table 8.1 shows that, taking into account the 10× geometrical scaling between full and model-scale, the localization error is larger (5.38 m ± 1.69 m) using PDQS-SLAM than the baseline static SLAM at model-scale (0.46 m ± 0.17 m), while the mapping error is smaller (4.61 m± 1.91 m) than baseline (0.54 m ± 0.24 m). Recalling that in the application of static SLAM to the slow dynamic environment, the localization error was 0.84 m ± 0.33 m and the mapping error was 0.82 m ± 0.34 m, the performance of PDQS-SLAM is more similar to the baseline static SLAM performance in a static environment. This shows that PDQS-SLAM is successful in a complex environment (quasi-static with an event) that was otherwise unsuccessfully mapped by a UAV.

8.3.2 Case 2: Model-Scale in Experiment and Simulation

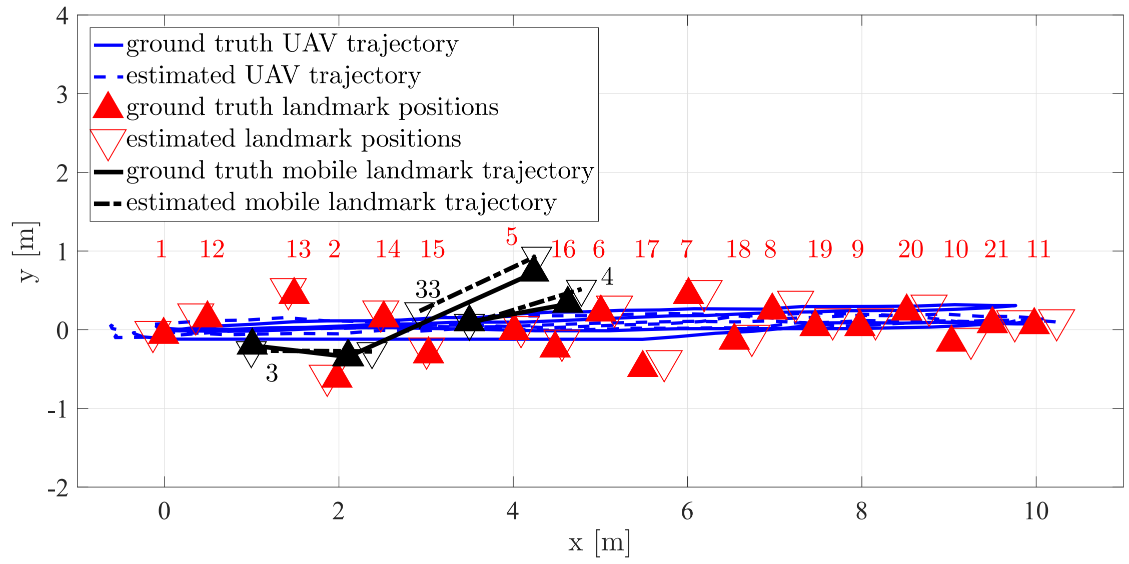
The second case tests the PDQS-SLAM in simulations and experiments at model-scale. Both tests are summarized in Table 8.2. The median-performance simulation trial is shown in Figure 8.5, while the median-performance experiment trial is shown in Figure 8.6.

Case 2 Localization and Mapping Results

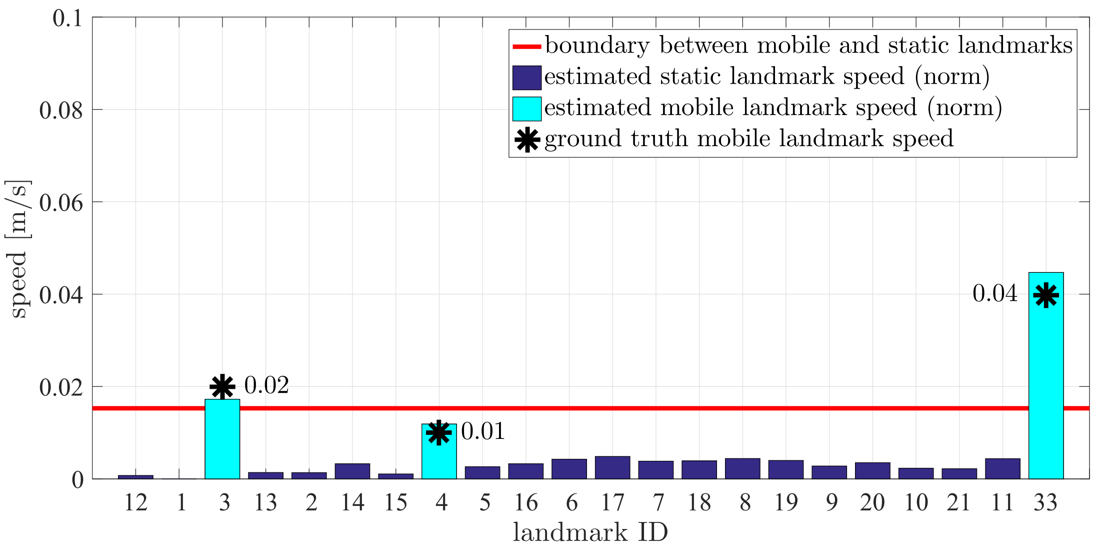
Table 8.2: Case 2 (two-pass, model-scale, one event) simulation (Figure 8.5) and experiment (Figure 8.6) performance summary for PDQS-SLAM.

metric	**unit**	**simulation**	**experiment**
localization error (norm)	[m]	0.33 ± 0.13	0.46 ± 0.16
x error	[m]	0.18 ± 0.05	0.29 ± 0.11
y error	[m]	0.11 ± 0.03	0.15 ± 0.05
z error	[m]	0.20 ± 0.16	0.24 ± 0.17
mapping error (norm)	[m]	0.34 ± 0.12	0.47 ± 0.19
x error	[m]	0.16 ± 0.06	0.32 ± 0.16
y error	[m]	0.11 ± 0.07	0.11 ± 0.03
z error	[m]	0.22 ± 0.18	0.25 ± 0.20
mean static landmark speed error	[m/s]	0.0028 ± 0.0007	0.0038 ± 0.0019
mean dynamic landmark speed error	[m/s]	0.0031 ± 0.0025	0.0104 ± 0.0052
mean speed boundary	[m/s]	0.0142 ± 0.0008	0.0114 ± 0.0035

The median simulated trial plotted in Figure 8.5a shows little cross-track error in the UAV trajectory and static landmark positions, and good matches between the simulated and ground truth mobile landmark trajectories. There is an along-track error on the trajectory and static landmark positions of approximately 20-30 cm, smaller than the baseline static SLAM case. The speed estimate for static landmarks is on average 0.0028 m/s (2.8 km/day equivalent) and the average error on mobile landmarks is 0.0034 m/s (3.4 km/day). Figure 8.5b shows that in the median-performance trial, the three KMMs are well estimated and the static landmarks have low speed estimates. Of the 10 trials, the event detection was always successful (detected at the correct time) for landmark 3's events and one trial had a false-positive event detected for landmark 5 during the return-to-start position of the second pass.



(a) planar map

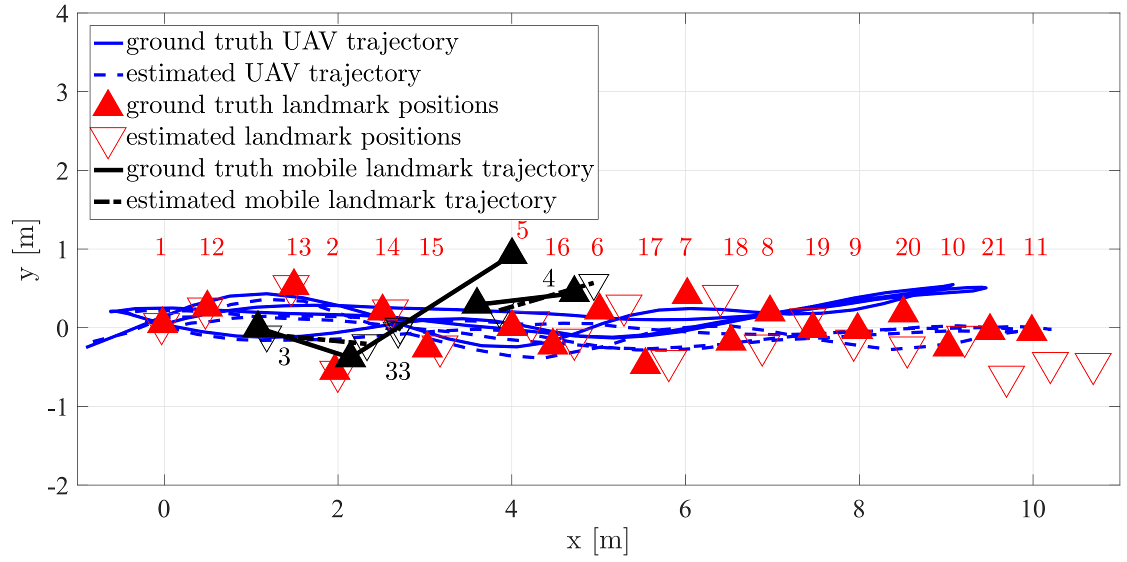


(b) speed estimates

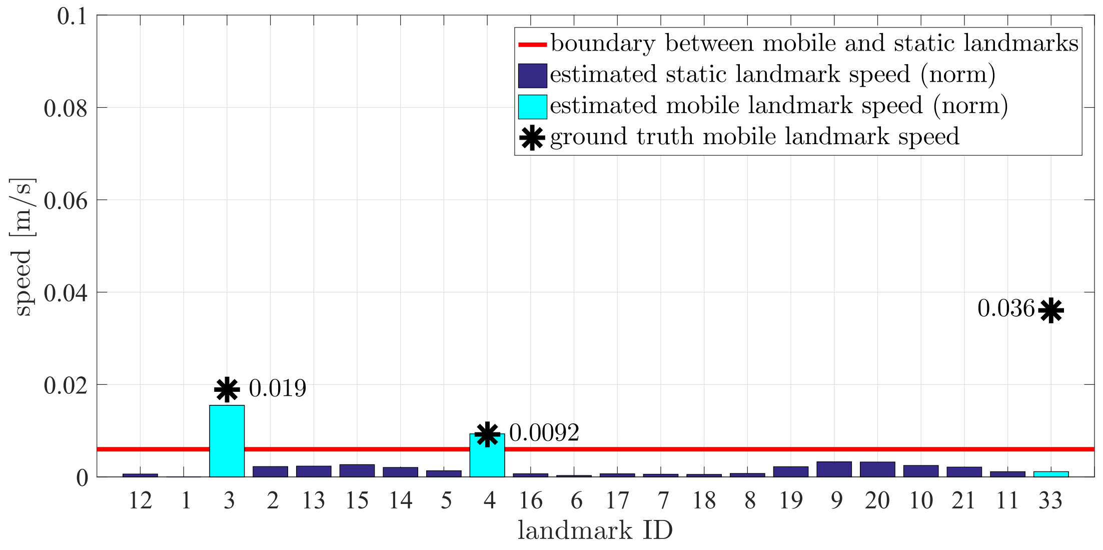
Figure 8.5: Case 2 simulation of PDQS-SLAM in piecewise-deterministic environment with one event.

This false-positive event did not diminish the accuracy of the PDQS-SLAM trajectory and is attributed to a particularly noisy AprilTag position estimate.

The experiment showed considerable cross-track drift of the quadrotor that the PDQS-SLAM was only partially able to compensate for. This drift resulted in mixed success in event detection. Of the ten trials performed in the lab the event on landmark 3 had the following event performance:



(a) planar map



(b) speed estimates

Figure 8.6: Case 2 experiment of MBQS-SLAM at model-scale, piecewise-deterministic environment with one event.

- one trial missed an event on landmark 3,
- six trials detected the event on landmark 3, but were late (event detected on return of second pass), and
- three trials successfully detected the event for landmark 3 and were able to correctly estimate the KMM after the event.

With regards to false-positive events, the ten trials included:

- three trials that had one false-positive event each (landmark 9; landmark 18; and landmark 5), and
- two trials that had three false-positive events each (landmarks 16, 4 and 5; and landmarks 12, 5, and 6).

In all of the trials the quadrotor was able to successfully localize itself despite the mixed event detection performance. This mixed event detection performance is not only due to the cross-track drift of the quadrotor, but also the variable motion of the NXT propelling the landmarks. Depending on their condition (such as the on-board battery charge, the cleanliness of the floor and wheels, initial orientation etc.) their path and speed through the environment varied across trials. Together with the drift in the quadrotor's route and altimeter noise indoors, these variations in NXT motion were large enough to result in observations being captured at different times or from different relative orientations leading to the variation in the ten trials. This variation was at times large enough that unintended events occurred, and other times intended events were less intense than planned. While future tests could endeavour to reduce this variance in the mobile iceberg analogues by more strictly controlling the set-up for consistent NXT charge and floor cleaning, or by considering changing from the NXTs to a platform with more precise control inputs, in all trials PDQS-SLAM successfully localized the quadrotor and mapped the environment despite the variability.

Case 2 Discussion

The reason that experimental trials are performed is especially clear in Case 2. While the accuracy of the simulation and experiment were similar (ex. localization accuracy in simulation 3.3% and in experiment 4.6% of the 10 m pass length), the variability of the trial success in the experiment is notably different from the consistency achieved in the simulation. This type of variation arises from the interaction of multiple factors that are not captured in a simulation environment. In general, the experiment trials showed that the event detection algorithm is prone to false-positive event detections. This could be reduced by tuning the pre-filter threshold, ε, or requiring a larger increase in graph error after adding the loop closure factors. However, as discussed during the design of the pre-filter in Section 7.1.3, it is preferable to have a false-positive error than to miss an event. The sensitivity of the event detection algorithm of PDQS-SLAM to the variability of trials is tied to the selection of the strict constant-velocity landmark KMM. If the variability of landmark motion in an environment is as seen with the NXTs, a constant acceleration or constant jerk model might be a more effective choice. Alternatively, the covariance used on the landmark velocity measurements (σ_{WLS} and σ_{QSA}) could be artificially increased, to accommodate small deviations from a strict constant-velocity KMM. However, this alternative would reduce the confidence, and potentially the accuracy, of the velocity estimates, when the constant-velocity model was well-followed by the landmarks.

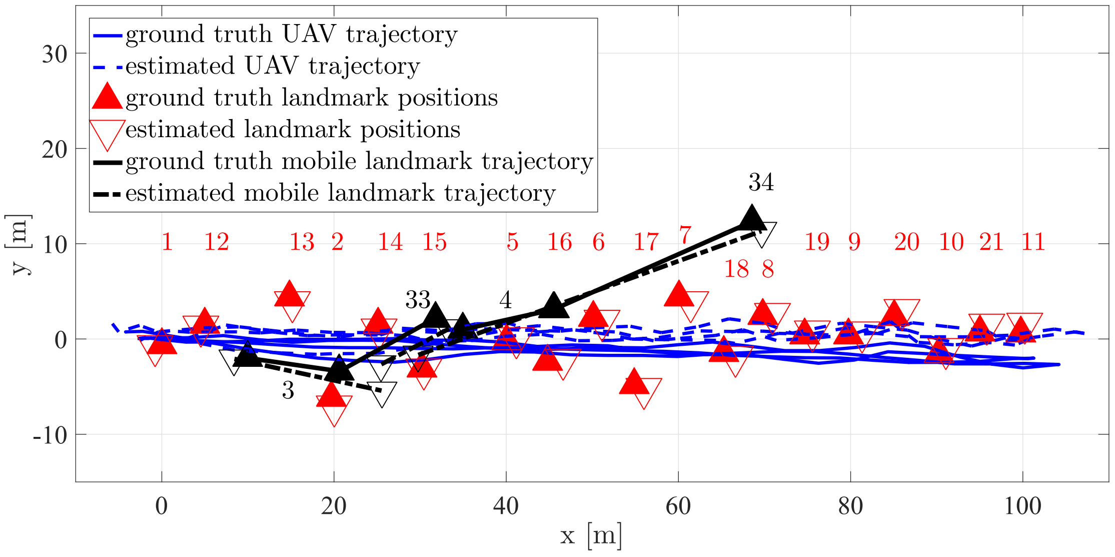
8.3.3 Case 3: Full-Scale in Simulation with Multiple Landmarks' KMM Disrupted

This third case presents the PDQS-SLAM algorithm with two events, one per mobile landmark. Landmark 3 experiences a change in heading only while landmark 4 experiences a change in speed only. This case was also performed in the sandbox in in Section 7.3.3, where it was able to successfully detect both events.

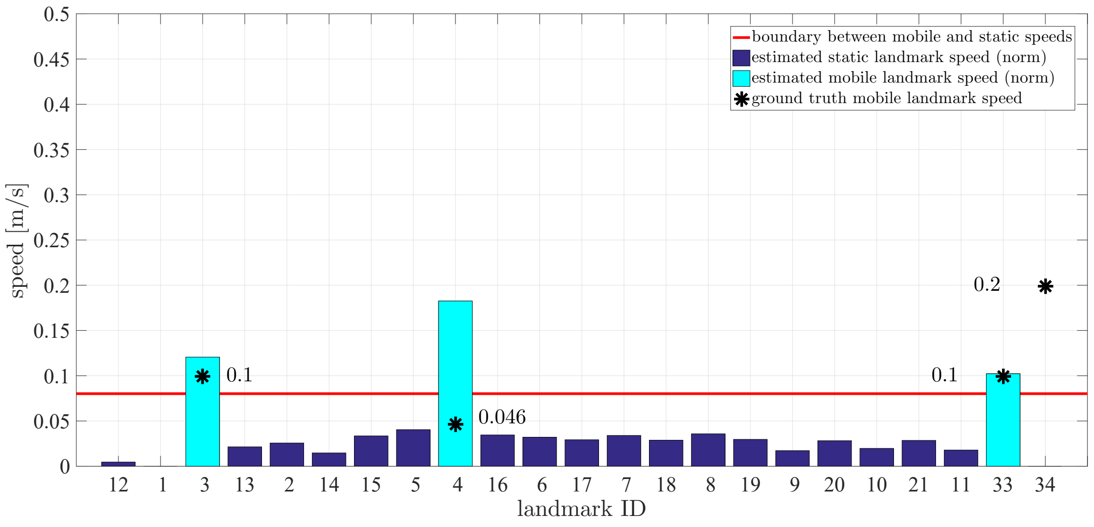
The same case is tested here, but in the simulated environment with odometry and landmark position estimation noise representative of the real environment. This not only tests the event detection algorithm, but also assess the ability to correctly estimate the KMMs after the events. This case is summarized in Table 8.3 and the median-performance trial is shown in Figure 8.7.

Case 3 Localization and Mapping Results

In this case with two events, the PDQS-SLAM is able to localize the robot with an average error of 3.53 m. The map has an average error on the static landmark positions of 2.98 m while the static landmarks are estimated with a small speed (6.2 km/day equivalent). The median trial that was plotted in Figure 8.7a shows what happens when the (speed-only) event on landmark 4 is not detected. The estimated trajectory of landmark 4 stretches the entire distance that landmark 4 travelled in the two epochs without recognizing that an event occurred. This is also visible in Figure 8.7b where the speed estimated for landmark 4 (0.18 m/s) is between the speed in the



(a) planar map



(b) speed estimates

Figure 8.7: Case 3 simulation of PDQS-SLAM at full-scale, piecewise-deterministic environment with two events.

first epoch (0.046 m/s) and the second epoch (0.20 m/s). In this case, not detecting the event for landmark 4 does not change the outcome for a ship navigating the environment as the magnitude of the event (speed change) was small so the impact on the map used to select a safe path was similarly small when looking ahead to project the future position of icebergs.

Table 8.3: Case 3 (two-pass, full-scale, two events) simulation performance summary for PDQS-SLAM (Figure 8.7).

metric	**unit**	**simulation**
localization error (norm)	[m]	3.53 ± 0.68
x error	[m]	2.04 ± 0.72
y error	[m]	1.67 ± 0.67
z error	[m]	1.60 ± 0.66
mapping error (norm)	[m]	2.98 ± 0.68
x error	[m]	1.44 ± 0.74
y error	[m]	1.34 ± 0.77
z error	[m]	1.72 ± 0.85
mean static landmark speed error	[m/s]	0.0310 ± 0.0067
mean dynamic landmark speed error	[m/s]	0.0785 ± 0.0206
mean speed boundary	[m/s]	0.0827 ± 0.0112

Case 3 Discussion

In the ten trials conducted, all ten successfully detected the event to landmark 3 at the correct time. The success for landmark 4 was mixed:

- one trial detected the event at the correct time;
- two trials detected the event but late, and
- seven trials did not detect the event at all.

Also, for one of the trials where the event to landmark 4 was not detected, one false-event was detected for landmark 6. This difference in success rate for landmark 3 (change in orientation only) and landmark 4 (change in speed only) indicates that the current pre-filter threshold is more sensitive to orientation than speed changes. This is actually a desirable characteristic in the context of marine Arctic environments where an iceberg's change in direction can result in a more difficult navigation problem than an iceberg increasing or decreasing speed on a trajectory that the navigator knows.

In terms of localization, the PDQS-SLAM algorithm performed better in the case of two events (localization: 3.53 m ± 0.68 m) than with only a single event (localization error: 5.38 m ± 1.69 m), with most of the difference between the two cases in the UAV altitude estimation (1.60 m ± 0.66 m in Case 3, compared to 3.20 m ± 2.02 m in Case 1) as also noted in previous cases. The along-track, cross-track and velocity estimation accuracies agree much more closely between Cases 1 and 3. Even with the high variance in the altitude estimate, both cases show the UAV is able to successfully complete the two-pass mission to return to the start position, and is able to consistently detect the events of landmark 3's KMM.

8.4 Summary of Results and Discussion

This chapter proposed PDQS-SLAM, an approach that enables a UAV to operate persistently in general quasi-static environments, where UAVs could not otherwise operate. This method combines the MBQS-SLAM developed in Chapter 6 for use within an epoch, with the event detection algorithm developed in Chapter 7, by proposing a novel state attribute, governed by an FSM, for edges in the pose graph. Insight into the evolution of a mobile landmark in a dynamic environment is at the core of the proposed PDQS-SLAM developments.

Despite being applied to a complex, piecewise-deterministic environment, PDQS-SLAM generally maintained the localization and mapping accuracy of the baseline static SLAM applied in a static environment. It showed similar success in estimating the velocity of mobile landmarks as MBQS-SLAM, and similar success in detecting events as in the idealized sandbox used in Chapter 7. While only three cases were studied here (with only two different event types/magnitudes), the more complete cases study the separate components in Chapters 6 and 7 provide confidence in the broader applicability. The patterns in the performance indicate that the FSM is a good description of the evolution of a quasi-static landmark in a piecewise-deterministic environment.

Parallel cases and variants

The proposed FSM for sets of edges related to a landmark in the piecewise-deterministic environment provides several opportunities for parallel cases to be addressed by varying the specific transition triggers and rules to, for example, different KMMs than constant velocity models. The FSM could also be adjusted for additional or substituted states if a different front-end were used that provided additional insight to the mobile landmarks besides their characteristic velocity. This might include a visual

SLAM system that provided insight into the expected direction or speed of motion based on a visual estimate of the age of the ice or environmental factors such as currents and waterway dimensions.

The detection of events in PDQS-SLAM could also function as a health monitor in other situations. For example, if the number of events were much higher than expected (information embedded in the selection of ε), the operator could be alerted to a potential sensor fault in the UAV, or an unanticipated environmental condition that warrants attention.

The states and transitions of a FSM could also be analysed probabilistically to, potentially, select the appropriate state for a set of edges by optimizing for the state in the same pose graph used for SLAM. Optimizing for the state of edges has been proposed by Sunderhauf et al. with the Switching Constraints Robust SLAM method [44], but an FSM provides more granularity than the simple on/off switch available with the switching constraints method.

8.4.1 Concluding Remarks

The PDQS-SLAM is a step towards persistence in quasi-static environments as it responsive to the inevitable disruptions to a landmark's motion in a complex environment. PDQS-SLAM improves with prior knowledge of the environment – particularly landmark characteristic velocity – however, it does not require precise knowledge and it is flexible to environments with a mix of static, quasi-static and slow landmarks present. This quasi-static regime is one that has not been previous addressed by SLAM problems, but is an important enabler for applications such as UAVs in marine Arctic environments.

Chapter 9 Conclusions

The research conducted through this thesis contributed to the field of persistent mapping by relaxing the static assumption in traditional SLAM algorithms that prevented a UAV from operating in a quasi-static dynamic environment, exemplified by the Canadian marine Arctic.

9.1 Summary

While SLAM in slow and fast dynamic environments has been previously addressed in literature, there existed a gap in capabilities for quasi-static environments that this research into relaxing the static assumption in graph SLAM addresses. This work was supported through several stages of development:

1. **landmark kinematic motion models:** a clear definition of a quasi-static dynamic environment was developed in the form of attributing a piecewise-deterministic KMM (a series of epochs disturbed by events) to landmarks that are traditionally considered static.

2. **quasi-static motion definition:** the interesting quasi-static motion regime was quantified relative to the mission and sensor parameters to distinguish its properties from semi-static and slow environments. Clarity on what factors result in a quasi-static environment and how landmarks in a quasi-static environment behave, allowed this motion to be incorporated into the graph structure using novel factors.

3. **model-based dynamic and quasi-static factors:** factors were constructed that allowed the KMM parameters of landmarks to be estimated directly in the graph, or through the expectation-maximization approximation. These factors allow the graph to optimize simultaneously for the UAV trajectory, as well as the landmark KMM parameters (ie. velocity). This means the graph no longer requires the landmarks to be static, only that the landmarks behave according to a constant velocity motion model within an epoch.

4. **score-based structure learning event detection:** the capabilities of SBSL were applied to the dynamic SLAM problem to take advantage of the concept that since the graph is a representation of the environment, the graph structure itself (specifically, whether a proposed loop-closure edge should be included, or be removed due to an event that disrupts the epoch) can be determined by monitoring the changes in the graph's score or error. This means that SLAM is no longer limited to epochs where the mobile landmarks have a constant motion model, as disruptions can be identified and appropriate recovery steps can be implemented.

5. **edge finite state machine attribute:** the definition of an edge in a factor graph was refined to include a state that specifies the behaviour of the edge. This state is governed by a finite state machine that describes the transitions between states based on the epoch or event the landmark used to construct the edge is experiencing or has experienced. This creates the opportunity for edges to evolve over time as the geometrical constraint they represent evolves in response to landmark motion, and allows the graph to detect and recover from events that disrupt that motion.

6. **end-to-end PDQS-SLAM validation:** these elements were combined into the PDQS-SLAM algorithm, which was validated in its elements (MBQS-SLAM in quasi-static epochs, and SBSL event detection), and as a single end-to-end algorithm. The PDQS-SLAM algorithm was evaluated in simulation and in experiments in a controlled laboratory setting. It was shown to be capable of accommodating multiple events for an environment with a mix of static, quasi-static and slow landmarks.

PDQS-SLAM was shown to have a similar performance in a piecewise-deterministic, quasi-static environment as the baseline static SLAM in a static environment (3-5% error on localization and mapping, relative to the pass length). This means that without loss of SLAM performance, the proposed approach provides additional capabilities of landmark motion estimation and UAV localization in quasi-static environments – key enablers to persistent mapping.

9.2 Future Work

This thesis developed the PDQS-SLAM algorithm to enable persistent mapping in quasi-static dynamic environments using a UAV, by relaxing the static assumption of traditional SLAM algorithms. While the simulation tests and experiments performed in this thesis provided important insight to the performance of PDQS-SLAM in dynamic environments, they could not capture the full range of phenomena that would occur in a deployment in the Arctic environment. The most impactful of these are recommended for further validation studies. In addition, this research opens several opportunities to expand the study of the PDQS-SLAM algorithm, and to apply elements of this work in other scenarios, including underwater navigation, under-ice mapping or agricultural mapping. These opportunities are listed as follows.

1. Further validate PDQS-SLAM towards Arctic trials:

   - Full-scale experiment and in-situ testing: the next stage of testing for the PDQS-SLAM would be to move to more complex trajectories and larger environments. An appropriate setting for such tests could be a nearby lake, with the operation of multiple unmanned surface vehicles (e.g. small remote controlled or autonomous boats) carrying tags to act as landmarks. Such an experiment would provide insight into the uncertainties and errors that arise in natural environments and impact SLAM performance.

   - Validation on long-distance and long-duration, persistent missions: this thesis limited tests to one- and two-pass missions as these passes are the most challenging for SLAM in unknown environments. In the Arctic environment, it would be useful to perform longer passes and longer-duration missions, if similar performance could be achieved. These tests would improve the understanding of the effects of graph size and density on the performance of PDQS-SLAM, as well as more complex sensor models and vehicle dynamics.

   - Landmark detection and identification without fiducial tags: this thesis relied on AprilTags to ensure any data association challenges were due to dynamics, not mis-identification of landmarks. A logical next step would be to integrate PDQS-SLAM with an appropriate front-end system to study these data association challenges. In particular, satellite observations of icebergs or observations gathered by aircraft (manned or unmanned) in previous trials would be ideal to incorporate into the Gazebo simulation environment. This test would help to identify what features on an iceberg are useful for recognition. It would also expose the impact of incorrect data associations on PDQS-SLAM performance, and allow an assessment of the relative magnitude of harm arising from incorrect data association in the dynamic sense (i.e. a loop closure across an event) compared to incorrect data association in the traditional sense (i.e. mis-recognition of landmarks).

2. Expand study of PDQS-SLAM:

    - Expand understanding of solution robustness: the work presented in this thesis showed that PDQS-SLAM provides advantages in many scenarios, however it also showed limitations for example, when there are many moving targets moving with a common velocity vector. As more trials are conducted, especially trials with larger numbers of observations of a common landmark, the robustness of the solution can be assessed more completely.

    - Application of Active SLAM to improve the SLAM performance: this thesis showed that there were many situations where the SLAM performance was limited by the number of observations, or the position of the UAV when making the observation. Active SLAM is a method by which the UAV's path is selected during the mission to meet the mission objectives (i.e. survey a region for icebergs) while simultaneously optimizing the UAV's localization accuracy. As an example, when the landmark velocity vector is very similar to the UAV direction of motion, it is difficult to accurately measure the relative motion. Active SLAM might choose a lawnmower (paired-track) path over the same region so that the directions of motion would be perpendicular and more easily measured by the moving UAV. Similarly, changing altitudes to increase the resolution of the sensor or decrease the number of landmarks visible in the UAV's FOV would be another example where Active SLAM would be useful.

    - Validation on a wider variety of motion models and events: this thesis explored key KMMs and events that are appropriate to the Canadian marine Arctic environment. Each application will have specific characteristic motion models and a broader study would be useful to understanding the breadth of PDQS-SLAM's applicability.

- Recommendations for characteristic velocity selection in complex environments: the PDQS-SLAM relied on prior knowledge of the characteristic speed of landmarks in the environment. It was shown to be flexible to landmarks outside this speed for the range from static through to slow, however more specific guidance on how to determine this value would be helpful especially in environments where landmarks move with very different speeds.

- Improve computational efficiency of implemented code: the software implemented in this thesis would benefit from re-factoring to improve its computational efficiency, especially before moving to larger environments with more landmarks and more potential events. This would also potentially allow the removal of the pre-filter on the event detector.

- Improvements to SBSL event detector: the field of structure learning provides many different approaches to scoring a graph. While the event detector was successfully demonstrated here, further study into other approaches may reveal techniques that improve the performance.

- Improvement to selection of threshold for segmenting static and mobile landmarks in post-processing: to aid understanding of the results, a threshold was selected to distinguish static and mobile landmarks. It was shown to be less useful in environments with a mix of quasi-static and slow landmarks, leaving room for improvement. This boundary does not change the performance of PDQS-SLAM, but could help an operator to understand the output more quickly.

- Generalize to a hierarchy of discretizations: the idea of dynamic landmarks as a continuum (static, quasi-static, slow, fast) was developed in this thesis in such a way that it lends itself well to a broader idea of a hierarchy of discretizations not only in terms of velocity thresholds at very slow speeds, but expanding to faster speed thresholds and labelling of landmarks in meaningful ways. The relationship between how a landmark is observed (e.g. the vehicle's speed, altitude, position uncertainty and the sensor's resolution, frequency and field of view) and how the observation impacts the SLAM problem provides insight for designing missions to better meet objectives. Descriptions of environments such as *fast*, *many moving landmarks*, *large* are intuitively only relevant within a defined context. Exploring non-dimensional parameters and or thresholds are interesting lines in inquiry to better support missions in complex environments.

3. Elements to consider applying in other scenarios:

    - Variants of the FSM: the FSM designed in this thesis is tailored to a specific KMM and and event detection algorithm. The flexibility of FSMs mean a deeper study of the impact of adding more granular states or adjusting the transition rules could provide further insight to dynamic SLAM applications beyond the quasi-static Arctic environment addressed in this work.

    - Variants of landmark evolution beyond KMMs: this thesis focused on SLAM in marine Arctic environments, making KMMs the most interesting type of landmark evolution. It would be interesting to explore the ability to apply these methods to model changes in other properties (ie. size, colour, reflectivity, etc).

    - Event detection or FSM state as health indicator: The rate of transition across states in the FSM, or the state of persisting in a particular state of the FSM could provide an indication of an anomaly or fault in the autonomous system. In the scenario described in this these, frequent transitions between the Aged and Active states might indicate observations that are very noisy or a poorly selected landmark KMM. Insights into these types of anomalies can allow interventions, either autonomously in-situ or by an operator between passes, that can improve the vehicle's performance.

    - Explore other useful edge attributes: The refinement of edges to include attributes besides the geometric constraint, in this case a state governed by an FSM, is a novel approach that could inspire a review of other attributes that would be useful to track along with the edges. In a dynamic environment the state provides a time-varying relevance for the edge, while other problems may reveal other context-aware attributes.

- Explore a unified graph optimization solution: having expressed the problem of estimating the KMM using a FSM in the thesis, it may be possible to consider a method to solve the SLAM problem while simultaneously estimating the appropriate state for each set of edges associated with a landmark. This could even result in a structure that uses a multiple-hypothesis model to assess whether the characteristic velocity for a landmark should be static, quasi-static, slow or fast without a prior (i.e. expert) expectation. This is likely to be computationally expensive, and may be best performed in an off-line capacity where the UAV performs the mission with PDQS-SLAM, then the map is improved by re-estimating the full trajectory and map after the mission is complete with the additional multiple-hypothesis model.

Bibliography

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics.* Cambrdige, Massachusetts: The MIT Press, 2006.

[2] M. L. Seto, *Marine Robot Autonomy.* New York: Springer Science+Business Media, 2013.

[3] S. Saeedi, M. Trentini, M. L. Seto, and H. Li, "Multiple-Robot Simultaneous Localization and Mapping: A Review," *Journal of Field Robotics*, vol. 33, no. 1, pp. 3–46, 2016.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, pp. 1309–1332, Dec 2016.

[5] H. F. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping: Part I," *IEEE Robotics and Automation Magazine*, pp. 99–108, June 2006.

[6] M. Kaess, *Incremental smoothing and mapping.* Ph.D. Thesis, Georgia Institute of Technology, 2008.

[7] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert, "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors," in *IEEE International Conference on Robotics and Automation*, pp. 4290–4297, 2014.

[8] F. Dellaert, "Factor Graphs and GTSAM : A Hands-on Introduction," Tech. Rep. GT-RIM-CP&R-2012-002, 2012.

[9] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental Smoothing and Mapping," *IEEE Transactions on Robotics*, vol. 24, pp. 1365–1378, Dec 2008.

[10] M. Pilté, S. Bonnabel, and F. Livernet, "A novel nonlinear least-squares approach to highly maneuvering target tracking," *Comptes Rendus Physique*, vol. 20, no. 3, pp. 228–239, 2019.

[11] R. Smith, M. Self, and P. Cheeseman, "Estimating Uncertain Spatial Relationships in Robotics," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 167–193, 1990.

[12] I. J. Cox, "Blanche-an experiment in guidance and navigation of an autonomous robot vehicle," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 193–204, 1991.

[13] J. J. Leonard and H. F. Durrant-Whyte, “Simultaneous map building and localization for an autonomous mobile robot,” in *Proceedings IROS ’91:IEEE/RSJ International Workshop on Intelligent Robots and Systems ’91*, pp. 1442–1447 vol.3, 1991.

[14] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, “Map validation and self-location in a graph-like world,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence*, (Chambery, France), pp. 1648–1653, August 1993.

[15] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, “Map validation and robot self-location in a graph-like world,” *Robotics and Autonomous Systems*, vol. 22, no. 2, pp. 159–178, 1997.

[16] B. Kuipers and P. Beeson, “Bootstrap learning for place recognition,” in *Eighteenth National Conference on Artificial Intelligence*, (USA), p. 174–180, American Association for Artificial Intelligence, 2002.

[17] T. Bailey and H. F. Durrant-Whyte, “Simultaneous Localization and Mapping (SLAM): Part II,” *IEEE Robotics and Automation Magazine*, pp. 108–117, September 2006.

[18] C. Stachniss, J. J. Leonard, and S. Thrun, “Simultaneous Localization and Mapping,” in *Springer Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), pp. 1153–1176, Springer, Cham, 2016.

[19] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, “A Tutorial on Graph-Based SLAM,” *IEEE Intelligent Transportation Systems Magazine*, pp. 31–43, 2010.

[20] A. N. Walcott, *Long-term robot mapping in dynamic environments*. Ph.D. Thesis, Massachusetts Institute of Technology, 2011.

[21] H. Kretzschmar, *Life-long Map Learning for Graph-based Simultaneous Localization and Mapping*. Msc thesis, Albert-Ludwigs-Universitat Freiburg, 2009.

[22] F. Lu and E. Milios, “Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans,” *Journal of Intelligent and Robotic Systems*, vol. 18, pp. 249–275, 1997.

[23] F. Dellaert and M. Kaess, “Square Root SAM: Simultaneous location and mapping via square root information smoothing,” *International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[24] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, and C. Hertzberg, “Hierarchical Optimization on Manifolds for Online 2D and 3D Mapping,” in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 273–278, 2010.

[25] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “G2o: A general framework for graph optimization,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3607–3613, 2011.

[26] M. Kaess, V. Ila, R. Roberts, and F. Dellaert, “The Bayes Tree: An Algorithmic Foundation for Probabilistic Robot Mapping,” in *Algorithmic Foundations of Robotics*, vol. 68, pp. 157–173, 2010.

[27] M. Hsiao and M. Kaess, “MH-iSAM2: Multi-hypothesis iSAM using bayes tree and hypo-tree,” *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2019-May, pp. 1274–1280, 2019.

[28] E. Olson, “AprilTag: A robust and flexible visual fiducial system,” in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3400–3407, May 2011.

[29] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, “Dynamic pose graph SLAM: Long-term mapping in low dynamic environments,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1871–1878, IEEE, oct 2012.

[30] J. McDonald, M. Kaess, C. Cadena, J. Neira, and J. J. Leonard, “Real-time 6-DOF multi-session visual SLAM over large-scale environments,” *Robotics and Autonomous Systems*, vol. 61, pp. 1144–1158, 2013.

[31] J. Aulinas, X. Llado, J. Salvi, and Y. R. Petillot, “Selective Submap Joining for Underwater Large Scale 6-DOF SLAM,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2552–2557, 2010.

[32] W. Maddern, M. J. Milford, and G. F. Wyeth, “Towards Persistent Localization and Mapping with a Continuous Appearance-based Topology,” in *Robotics: Science and Systems (RSS)*, 2012.

[33] C.-C. Wang, C. Thorpe, and S. Thrun, “Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas,” in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, pp. 842–849, IEEE, 2003.

[34] F. Pomerleau, P. Krusi, F. Colas, P. Furgale, and R. Siegwart, “Long-term 3D map maintenance in dynamic environments,” in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3712–3719, May 2014.

[35] Z. Wang, R. Ambrus, P. Jensfelt, and J. Folkesson, “Modeling motion patterns of dynamic objects by IOHMM,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1832–1838, 2014.

[36] E. Galceran, E. Olson, and R. M. Eustice, “Augmented vehicle tracking under occlusions for decision-making in autonomous driving,” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3559–3565, 2015.

[37] A. K. Ushani, N. Carlevaris-Bianco, A. G. Cunningham, E. Galceran, and R. M. Eustice, “Continuous-time estimation for dynamic obstacle tracking,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1137–1143, Sept 2015.

[38] K. P. Murphy, “Bayesian map learning in dynamic environments,” *Advances in Neural Information Processing Systems*, vol. 12, pp. 1015–1021, 2000.

[39] D. Moratuwage, B.-N. Vo, and D. Wang, “Collaborative Multi-Vehicle SLAM with Moving Object Tracking,” in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5702–5708, 2013.

[40] D. Moratuwage, D. Wang, A. Rao, N. Senarathne, and H. Wang, “RFS Collaborative Multivehicle SLAM: SLAM in Dynamic High-Clutter Environments,” *IEEE Robotics and Automation Magazine*, vol. 21, pp. 53–59, June 2014.

[41] J. P. Fentanes, B. Lacerda, T. Krajnik, N. Hawes, and M. Hanheide, “Now or later? Predicting and maximising success of navigation actions from long-term experience,” in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1112–1117, May 2015.

[42] P. Kimball and S. Rock, “Sonar-based iceberg-relative AUV navigation,” *2008 IEEE/OES Autonomous Underwater Vehicles, AUV 2008*, pp. 1–6, 2008.

[43] P. Kimball and S. Rock, “Estimation of iceberg motion for mapping by AUVs,” *2010 IEEE/OES Autonomous Underwater Vehicles, AUV 2010*, pp. 1–9, 2010.

[44] N. Sunderhauf and P. Protzel, “Towards a robust back-end for pose graph SLAM,” in *IEEE International Conference on Robotics and Automation*, pp. 1254–1261, 2012.

[45] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, “Robot Map Optimization using Dynamic Covariance Scaling,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[46] P. Agarwal, *Robust Graph-Based Localization and Mapping.* Ph.D. Thesis, Albert-Ludwigs-Universitat Freiburg, 2015.

[47] P. Y. Lajoie, S. Hu, G. Beltrame, and L. Carlone, “Modeling Perceptual Aliasing in SLAM via Discrete-Continuous Graphical Models,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1232–1239, 2019.

[48] V. J. Hodge and J. Austin, “A survey of outlier detection methodologies,” *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[49] K. Murphy, "Logistic Regression: Residual analysis (outlier detection)," in *Machine Learning: A Probabilistic Perspective*, ch. 8.4.5, pp. 260–261, MIT Press, 2012.

[50] D. Brauckhoff, K. Salamatian, and M. May, "A signal processing view on packet sampling and anomaly detection," *Proceedings - IEEE INFOCOM*, 2010.

[51] J. Mai, C. N. Chuah, A. Sridharan, T. Ye, and H. Zang, "Is sampled data sufficient for anomaly detection?," *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, pp. 165–176, 2006.

[52] T. Krajnik, J. P. Fentanes, G. Cielniak, C. Dondrup, and T. Duckett, "Spectral Analysis for Long-Term Robotic Mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3706–3711, 2014.

[53] K. P. Murphy, "Graphical model structure learning," in *Machine Learning: A Probabilistic Perspective*, ch. 26, pp. 907–944, MIT Press, 2012.

[54] D. Koller and N. Friedman, *Probabilistic Graphical Models- Principles and Techniques.* Cambridge, Massachusetts: The MIT Press, 2009.

[55] A. Carvalho, "Scoring functions for learning bayesian networks," *Inesc-id Tec. Rep*, 2009.

[56] A. Hinduja, B. J. Ho, and M. Kaess, "Degeneracy-Aware Factors with Applications to Underwater SLAM," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1293–1299, 2019.

[57] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 1557–1563, 2003.

[58] D. F. Wolf and G. S. Sukhatme, "Mobile robot simultaneous localization and mapping in dynamic environments," *Autonomous Robots*, vol. 19, no. 1, pp. 53–65, 2005.

[59] C. Sheng, S. Pan, W. Gao, Y. Tan, and T. Zhao, "Dynamic-DSO: Direct sparse odometry using objects semantic information for dynamic environments," *Applied Sciences (Switzerland)*, vol. 10, no. 4, 2020.

[60] S. Han and Z. Xi, "Dynamic Scene Semantics SLAM Based on Semantic Segmentation," *IEEE Access*, vol. 8, pp. 43563–43570, 2020.

[61] D. Li, W. Yang, X. Shi, D. Guo, Q. Long, F. Qiao, and Q. Wei, "A visual-inertial localization method for unmanned aerial vehicle in underground tunnel dynamic environments," *IEEE Access*, vol. 8, pp. 76809–76822, 2020.

[62] E. Einhorn and H.-M. Gross, “Generic NDT mapping in dynamic environments and its application for lifelong SLAM,” *Robotics and Autonomous Systems*, vol. 69, pp. 28–39, 2015.

[63] M. J. Milford and G. F. Wyeth, “Persistent Navigation and Mapping using a Biologically Inspired SLAM System,” *International Journal of Robotics Research*, vol. 29, no. 9, pp. 1131–1153, 2010.

[64] M. Labbe and F. Michaud, “Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation,” *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 734–745, 2013.

[65] Y. Li, S. Li, and Y. Ge, “A biologically inspired solution to simultaneous localization and consistent mapping in dynamic environments,” *Neurocomputing*, vol. 104, pp. 170–179, 2013.

[66] B. Bacca, J. Salvi, and X. Cufi, “Long-term mapping and localization using feature stability histograms,” *Robotics and Autonomous Systems*, vol. 61, pp. 1539–1558, 2013.

[67] F. Dayoub, G. Cielniak, and T. Duckett, “Eight Weeks of Episodic Visual Navigation Inside a Non-stationary Environment Using Adaptive Spherical Views,” *Field and Service Robotics*, pp. 379–392, 2015.

[68] M. S. Bahraini, M. Bozorg, and A. B. Rad, “SLAM in dynamic environments via ML-RANSAC,” *Mechatronics*, vol. 49, pp. 105–118, 2018.

[69] M. Henein, J. Zhang, R. Mahony, and V. Ila, “Dynamic SLAM: The need for speed,” *arXiv*, pp. 2123–2129, 2020.

[70] P. W. Kimball and S. M. Rock, “Mapping of translating, rotating icebergs with an autonomous underwater vehicle,” *IEEE Journal of Oceanic Engineering*, vol. 40, no. 1, pp. 196–208, 2015.

[71] V. Estivill-Castro, “NXTDriver.” Available: `https://github.com/mipalgu/NXTdriver` (accessed: Sept. 1, 2016), 2014.

[72] V. Estivill-Castro, “NXTController.” Available: `https://github.com/mipalgu/NXTcontroller` (accessed: Sept. 1, 2016), 2014.

[73] A. Chovancová, T. Fico, E. Chovanec, and P. Hubinský, “Mathematical modelling and parameter identification of quadrotor (a survey),” *Procedia Engineering*, vol. 96, pp. 172–181, 2014.

[74] Z. Huang, *Consensus Control of Multiple-Quadrotor Systems under Communication Delays*. M.A.Sc. Thesis, Dalhousie University, 2017.

[75] A. Jaegle, S. Phillips, and K. Daniilidis, "Fast, robust, continuous monocular egomotion computation," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 773–780, June 2016.

[76] P. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch trajectory estimation using temporal basis functions," in *International Conference on Robotics and Automation (ICRA)*, 2012.

[77] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor UAVs using ROS and Gazebo," *Lecture Notes in Computer Science*, vol. 7628, pp. 400–411, 2012.

[78] M. Monajjemi, "Bebop Autonomy." Available: `https://github.com/AutonomyLab/bebop_autonomy` (accessed: July 1, 2019), 2018.

[79] G. Silano, P. Oppido, and L. Iannelli, "Software-in-the-loop simulation for improving flight control system design: A quadrotor case study," *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pp. 466–471, October 2019.

[80] D. Kortenkamp, R. Simmons, and D. Brugali, "Robotic Systems Architectures and Programming," in *Springer Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), pp. 283–306, Springer, Cham, 2016.

[81] A. Deeb, M. L. Seto, and Y. Pan, "Model-based Dynamic Pose Graph SLAM in Unstructured Dynamic Environments," in *International Conference on Advanced Robotics (ICAR)*, pp. 123–128, 2019.

[82] F. Sittel, J. Müller, and W. Burgard, "Computing Velocities and Accelerations from a Pose Time Sequence in Three-dimensional Space," Tech. Rep. TR 272 Informatik Uni-Freiburg De, April 2013.

[83] R. M. Neal and G. E. Hinton, "A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants," in *Learning in Graphical Models*, pp. 355–368, 1998.

[84] M. C. Graham and J. P. How, "Robust simultaneous localization and mapping via information matrix estimation," in *IEEE/ION Position, Location and Navigation Symposium - PLANS*, pp. 937–944, 2014.

[85] D. R. Wright, "Finite State Machines," 2005.

[86] V. de Araujo, A. Paula G. S. Almeida, C. T. Miranda, and F. de Barros Vidal, "A Parallel Hierarchical Finite State Machine Approach to UAV Control for Search and Rescue Tasks," in *Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics*, pp. 410–415, SCITEPRESS - Science and and Technology Publications, 2014.