Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

Sigurd Vatn Totland

Graph-Based Multi-Modal SLAM for Resilience in Sensor-Degraded Environments

Masteroppgave i Kybernetikk og robotikk
Veileder: Kostas Alexis
Medveileder: Nikhil Khedekar

Juli 2021

NTNU
Kunnskap for en bedre verden

Sigurd Vatn Totland

Graph-Based Multi-Modal SLAM for Resilience in Sensor-Degraded Environments

Masteroppgave i Kybernetikk og robotikk
Veileder: Kostas Alexis
Medveileder: Nikhil Khedekar
Juli 2021

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

NTNU
Kunnskap for en bedre verden

Sigurd Vatn Totland

Graph-Based Multi-Modal SLAM for Resilience in Sensor-Degraded Environments

Master Thesis in Cybernetics and Robotics
Supervisor: Kostas Alexis
Co-supervisor: Nikhil Khedekar
July 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU

Norwegian University of
Science and Technology

Abstract

Reliable and accurate *simultaneous localization and mapping*, or *SLAM*, is a vital prerequisite for effective deployments of autonomous systems that enable robots to map and navigate their surroundings, even in a GPS-denied setting. While current state-of-the-art SLAM systems are highly accurate and precise, most still struggle in sensor-degraded environments, due to reliance on a single exteroceptive sensor modality such as a camera or LiDAR. Resilience to such sensor degradation could extend the reach of autonomous systems beyond what is possible today, with applications ranging from subterranean operations to space exploration. It is essential for the success of this to take advantage of multiple complementary sensors, thus providing resilience to single-sensor failure through the redundancy and resourcefulness offered by additional sensor modalities. Motivated by this, we present a semi-tight LiDAR-visual-inertial fusion system based on the mathematical framework of factor graphs and the iSAM2 incremental smoothing and mapping algorithm. Our contribution includes a) a monocular visual frontend that enriches visual features with LiDAR depth, and b) an iSAM2-based sliding-window backend that fuses the visual data with inertial measurements and odometry constraints from a LiDAR odometry system. We test our proposed system on the open *Newer College* dataset as well as in a degraded environment where LiDAR-only localization fails due to self-similar geometry. Our results suggest that a multi-modal approach can offer increased resilience to sensor degradation, motivating the need for further research on multi-modal perception. The work presented in this thesis stands as a preliminary effort to help develop a new tight-fusion paradigm for multi-modal SLAM.

Sammendrag

Pålitelig og nøyaktig *samtidig lokalisering og kartlegging* (SLAM) er en nødvendig forutsetning for vellykket bruk av autonome systemer, som gjør det mulig for roboter å kartlegge og navigere omgivelsene sine, uten bruk av GPS. Selv om dagens SLAM-systemer er svært nøyaktige og presise, sliter de fleste likevel i sensordegraderte miljøer, siden de er avhengige av én enkelt eksteroceptiv sensormodalitet som for eksempel et kamera eller en LiDAR. Motstandsdyktighet ovenfor slik sensordegradering kan utvide horisonten til autonome systemer utover det som er mulig i dag, og dermed muliggjøre alt fra underjordiske operasjoner til romforskning. For å lykkes med dette er det nødvendig å ta i bruk flere komplementære sensorer og dermed ta nytte av den økte motstandsdyktigheten og redundansen tilleggssensorer kan tilby. På bakgrunn av dette presenterer vi et semi-tett LiDAR-visuell-inertiell fusjonssystem baser på det matematiske rammeverket som *faktorgrafer* tilbyr og den inkrementelle glatting- og kartleggingsalgoritmen iSAM2. Vårt bidrag er todelt og inkluderer a) en monokulær visuell *front-tend* som beriker visuelle landemerker med dybde fra LIDAR, og b) en iSAM2-basert "bevegende-vindu" *backend* som fusjonerer den visuelle dataen med inertielle sensormålinger og odometrifaktorer fra et LiDAR-odometrisystem. Vi tester det foreslåtte systemet på det åpne datasettet *Newer College dataset* og i et degradert miljø der lokalisering basert på LiDAR alene feiler på grunn av geometrisk selvlikhet i omgivelsene. Resultatene våre tilsier at en multimodal fremgangsmåte kan gi økt motstandsdyktighet ovenfor sensordegradering og motiverer dermed videre forskning på multimodal persepsjon. Arbeidet presentert i denne oppgaven er hovedsakelig en innsats for å utvikle et nytt paradigme for tett koblet sensorfusjon.

Preface

This master's thesis is the result of five years of study in *cybernetics and robotics*, specifically within the direction of *autonomous systems*, at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology, NTNU. The thesis project is meant as a work of a larger character performed individually by the student under supervision from an academic supervisor and co-supervisor.

As with any academic work, it is made possible only by standing on the shoulders of giants. In my case, certain open source frameworks and software libraries are foundational and so can not go without a mention. Software wise, the libraries GTSAM (Georgia Tech Smoothing and Mapping) and OpenCV (Open Computer Vision) are foundational to everything developed for this thesis. Additionally, the excellent Robotic Operating System (ROS) is the workhorse that makes everything work together. In addition to open source software, several code snippets that my co-supervisor Nikhil has provided have been of great help. These include snippets for stationary IMU–gravity alignment and LiDAR pointcloud-to-image projection. His `ely` software repo has also been of great inspiration for setting up image undistortion using OpenCV. Finally, the datasets of LiDAR, visual and IMU data provided by the lab have been foundational to the testing of the proposed system and the results presented.

But the greatest help of all has been the excellent guidance offered by my supervisor, Kostas Alexis and my co-supervisor Nikhil Khedekar. Without all your guidance and counselling, none of this would have been possible. You have my deepest thanks. I would also like to thank the others at the Autonomous Robots Lab, especially Huan Nguyen and Shehryar Khattak for giving me access to datasets and configuration files. Last but not least, I thank Linn and my loving family for all their unconditional love and support and for helping me stay sane through all of this.

Contents

1 Introduction

One of the primary challenges faced by autonomous robotic systems, is the need for accurate and resilient perception. Of particular interest within the field of robot perception is the ability to perform SLAM, or *simultaneous localization and mapping*. The SLAM problem amounts to mapping the environment of the robot while simultaneously estimating its location within it. Solutions to this problem enable a wide range of applications such as control, planning, mapping and environmental understanding. It is therefore of paramount importance to solve the problem in a way that is both accurate, resilient to challenging surroundings and—crucially—capable of real time operation.

Within the field of simultaneous localization and mapping, a large body of research has been dedicated to solutions with a single *exteroceptive* sensor, such as a visual camera or a LiDAR sensor [4]. Many solutions have also made use of inertial measurement units (IMUs) in conjunction with these sensors [1, 31, 38]. Fusion of several *exteroceptive* sensors however still presents an open and active research field. Highly complementary sensor configurations such as LiDAR-visual or visual-thermal have shown promising results in enabling resilient perception, yet are still largely unexplored in the literature. It is now undestood however, that exploiting such complementary configurations is the key to achieving true resilience in SLAM and many such approaches have therefore been presented in recent years [22, 49, 42, 29].

Multi-modal sensor fusion systems can generally be categorized into either *loosely coupled* systems or *tightly coupled systems*. In a loosely coupled system, the localization problem is solved separately for each modality, and the results subsequently fused. Tightly coupled systems instead optimize over a joint problem comprising the measurements from all sensors. This tightly coupled paradigm currently constitutes the majority of current reseach, because, while it requires more computing power, it is generally considered to provide superior results [6]. In the context of resilient perception, tight coupling enables using information from a partially failing sensor modality, such as a LiDAR measuring a geometrically self-similar environment or a camera that is tracking a low number of features. A loosely coupled system would in such cases be forced to turn off one of the modalities to avoid deteriorating the fused result [29], but a tightly coupled system could instead benefit from the additional information from the degraded sensor.

Because the SLAM problem is fundamentally of nonlinear nature, solutions must be equipped with ways of handling the nonlinear, non-convex optimization problem that arises. The extended Kalman filter was for many years the dominating solution to this problem, however recent advances in computer hardware has enabled solutions based on the more computationally expensive, but also more accurate *maximum a posteriori* (MAP) optimiza-

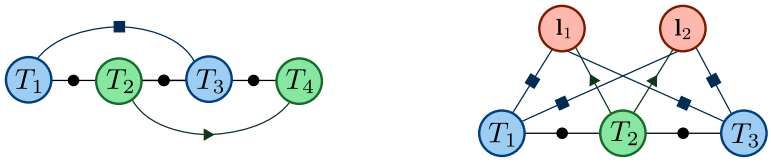


Figure 1.1: Example factor graphs for loosely coupled **(left)** and tightly coupled **(right)** fusion. In the left figure, we show an interleaved pose graph with green and blue poses corresponding to two different sensor modalities. The poses are connected both with factors from their individual single-modality subsystems indicated respectively with triangles and squares, and a motion model connecting the consecutive variables. In the right figure, the factors from the two modalities are instead measurements of the same underlying quantities: the landmarks $\mathbf{l}_1$ and $\mathbf{l}_2$.

tion [4]. This formulation is typically expressed graphically as a *factor graph*, a graphical model that seamlessly incorporates the optimization variables, measurements and the probabilistic relations between them. Factor graphs are well suited for multi-modal SLAM as they naturally allow expressing both loosely and tightly coupled sensor fusion formulations with ease, examples of which as are shown in Figure 1.1. Methods based on factor graphs and MAP optimization solve the inference problem as a nonlinear optimization problem and employ nonlinear solvers such as Levenberg Marquardt [32] and Gauss Newton [37]. Many state-of-the-art algorithms today, fall into this category [5, 22, 38, 40, 41, 48, 49], several of which make use of the factor graph framework of Georgia Tech Smoothing and Mapping (GTSAM) [11]. The latter also includes the incremental smoothing and mapping algorithm iSAM2 by Kaess et al, which exploits the sparsity of the SLAM problem factor graph to save computation for large problem instances [28].

The contribution presented in this work is a LiDAR-visual-inertial odometry system formulated as a factor graph in the GTSAM library. Our system fuses odometry estimates from a *LiDAR odometry and mapping* (LOAM) subsystem [53] together with LiDAR-depth-enriched visual landmarks and preintegrated IMU factors [19] in a semi-tight coupling. The implementation consists of a custom visual frontend and a sliding-window factor graph backend based on the iSAM2 framework [28]. We test our system on the open source Newer College dataset [39] with and without aiding from LOAM, and then present a simulated degeneracy study, where LOAM is artificially degraded for parts of the dataset. Finally, we test our system in a geometrically self-similar environment where the LiDAR odometry is naturally degenerate.

The rest of this text is organized as follows. In Chapter 2 we present related work to this thesis. Chapter 3 then presents the theory forming the background for our work. In Chapter 4, our proposed approach to the multi-modal fusion problem is presented, and in Chapter 5 we present an experimental evaluation of our method. We finally end with discussion and concluding remarks in Chapter 6.

2 Related Work

2.1 Graph-Based SLAM

Our work builds on the many existing SLAM and odometry methods based on factor graphs and nonlinear optimization. Within the field of computer vision, the SLAM problem has for a long time been solved using nonlinear optimization techniques similar to the graph-based approach we use. Here, the problem is known as *bundle adjustment* (BA) and amounts to simultaneously estimating a set of camera poses and *landmarks* in the map observed by the cameras. It is typically formulated as a *maximum likelihood problem*, which is a close cousin of MAP estimation as we will see in Chapter 3 [24]. Advances in computer hardware have recently made solving this problem feasible in a real-time setting. One of the first methods to do so was the work of Mouragnon et al. [35] which solved the bundle adjustment problem over a small set of local keyframes. Later, the highly successful *parallel tracking and mapping* (PTAM) algorithm of Klein and Murray [30] solved the frame-to-frame localization problem with nonlinear optimization and performed full BA on a larger map of keyframes and landmarks. OKVIS by Leutenegger et al. [31] later extended the bundle adjustment problem by including measurements from inertial measurement units (IMUs) in a tightly coupled keyframe-based sliding window factor graph. This paradigm of tightly coupling IMU and visual data in a joint factor graph has since become the de-facto standard for formulations of visual-inertial-odometry [6], and many of the recent works in this direction use similar approaches [5, 38, 40, 48, 49].

The work of Daellert and Kaess et al. on square root SAM [12] and the subsequent iSAM [26] and iSAM2 [28] algorithms present ways of efficiently solving the nonlinear MAP optimization problem by exploiting the sparsity of the graphical model associated with the SLAM problem. While square root SAM and iSAM based this on sparse matrix factorization methods, iSAM2 operates directly on the underlying graphical models. The Bayes tree datastructure underlying the method enables full MAP optimization on large problem instances through caching and partial re-linearization, by structuring the graphical model so that new updates only affect local variables. Recent methods such as Kimera [40] and VILENS [49] make use of the efficient marginalization capabilities of the Bayes tree to maintain their factor graph as a sliding window of states within iSAM2.

2.2 Loosely Coupled Lidar-Visual-Inertial Fusion

LiDAR and visual cameras are good candidates for fusion because of their complementary properties [29]. We categorize these systems as either loosely or tightly coupled depending on whether they fuse the results of standalone

localization systems (loose coupling), or whether they optimize over a single joint optimization problem that includes the sensor measurements directly (tight coupling). Many loosely coupled methods use the well known LiDAR odometry and mapping (LOAM) algorithm [53] based on point cloud scan matching with optional IMU-based point cloud undistortion. One example is V-LOAM, which adds a visual odometry system as a motion prior for the odometry [52]. This is mainly done to improve localization performance, and successfully does so as it currently ranks highest of the LiDAR solutions on the KITTI odometry benchmark [21]. This approach adds no extra resilience to LiDAR degradation however, as the visual odometry is only included as a prior. Khattak et al. presented in [29] a degeneracy-aware variant of this scheme which could heuristically detect degeneracy of the LOAM subsystem and use a visual-inertial or thermal-inertial odometry estimate as a pass-through in such cases, thereby adding resilience to either LiDAR or camera failure. The recent work of LVI-SAM [42] proposes two tightly coupled systems, one LiDAR-inertial and one LiDAR-visual-inertial with depth-enriched visual features. The final outputs of these systems are then fused in a loosely coupled pose graph which is resilient to degeneracy of either modality. A similar recent system is SuperOdometry [55], which also loosely couples the outputs of individually tightly coupled LiDAR-inertial and visual-inertial systems, to provide accurate and resilient perception without sacrificing extensibility.

2.3 Tightly Coupled Lidar-Visual-Inertial Fusion

Several tightly coupled Lidar-Visual-Inertial methods have recently been presented. Many methods make use of LiDAR point clouds to enrich visual landmarks with depth measurements for tightly coupled LiDAR-visual-inertial fusion, examples of which are LIMO [22], LVI-SAM [42] and VILENS [49]. This can improve localization performance and robustness and is not prone to failure in geometrically self-similar environments because they do not rely on point cloud scan matching. LIC-fusion and LIC-fusion 2.0 [57, 56] includes LiDAR data as line and plane features extracted from the point clouds in the estimation problem and minimize the point-to-line and point-to-plane distances between these. VILENS [49] combines all these ideas into a single unified factor graph including depth-enriched visual features, LiDAR line and plane features and IMU factors.

3 Theory

3.1 Notation

We define most notation on an as-needed basic, but summarize in this section some of our broad notation characteristics used throughout.

Typefaces

For vector-valued quantities, we use bold typeface, e.g. $\mathbf{x}$. Matrices are denoted in uppercase X and scalars are denoted in regular typeface x. Sets of variables are denoted in caligraphy style, $\mathcal{X}$ and index sets in sans-serif style, X.

Quantities

We typically use the following symbols to refer to specific quantities: Robot poses on the *special euclidian group* $SE(3)$ are denoted with T, rotations are denoted with R and translations with $\mathbf{t}$. Measurements are denoted with $\mathbf{z}$, landmarks with $\mathbf{l}$, factors with ϕ and timestamps with t.

Indices

Because there are many sets of different sizes to keep track of, we try to stay consistent when indexing sets of the same quantities. For robot pose timesteps we use k, so e.g. T_k denotes the k-th pose. For measurements and corresponding factors, we use i so $\mathbf{z}_i$ and ϕ_i denotes the i-th measurement and factor, respectively. Landmarks are indexed with m and when in a multi-modal context, modalities are indexed with s.

Coordinate Frames

Coordinate frames for rotations and $SE(3)$ transformations are denoted with subscripts such as R_{AB} to describe the rotation of frame B expressed in frame A. For vectors, we write the subscript on the left side to indicate what frame it is in, e.g. ${}_{\mathrm{A}}\mathbf{v}$ is a vector in frame A. Furthermore, we denote with right subscripts what frames the vector is pointing from and to, e.g. ${}_{\mathrm{A}}\mathbf{v}_{\mathrm{AB}}$ is a vector from frame A to B, as seen in frame A. To disambiguate between other subscript quantities, we consistently use capital letters to denote frames.

3.2 SLAM From a Maximum a Posteriori Perspective

The problem of simultaneous localization and mapping consists of estimating the trajectory traversed by a robot as well as mapping its surroundings. The problem is complex due to the need for *exteroceptive* sensors that provide only information about the outside world and not the motion of the robot directly.

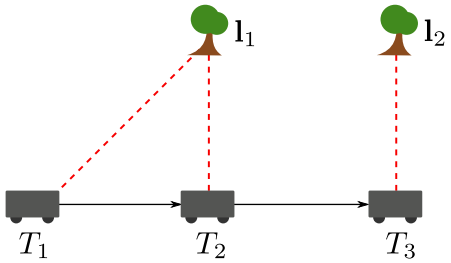


Figure 3.1: A toy SLAM problem showing a ground robot traversing an environment and receiving bearing-range measurements of nearby trees. Red dotted lines indicate bearing-range measurements from a laser range-finder whereas black arrows indicate integrated proprioceptive measurements from e.g. an IMU. The figure is adapted from [13].

Such exteroceptive sensors, of which LiDARs and cameras are examples, are necessary because *proprioceptive* sensors such as wheel encoders or inertial measurement units (IMUs) must be integrated over time and hence introduce considerable drift in a short amount of time. This results in a situation where the sensors measure–directly or indirectly–the structure of the environment, and the robot trajectory must be inferred from the estimated environment.

As an example, consider the toy SLAM problem presented in Figure 3.1. This is the same example as the one used throughout [47], originally adapted from [13]. In this example, a robot is traversing an environment and receiving bearing-range measurements of trees around it as well as some kind of integrated proprioceptive motion estimates. The SLAM problem here amounts to finding the location of the trees and the complete trajectory of the robot. In the SLAM literature, map features like the trees in this example are referred to as *landmarks* [4]. The fact that we use trees in this example is purely for illustratory purposes, and landmarks can in principle be any measured features of interest, as long as they satisfy the assumption of being stationary in the environment [46]. The static nature of the landmarks means we only need to represent them as a single variable in the estimation problem. The robot pose on the other hand, which includes both the location and rotation of the robot must be represented with one variable for each timestep, so that the entire trajectory can be estimated. The complete system state here hence consists of the two landmark locations $\mathbf{l}_1$ and $\mathbf{l}_2$ as well as the three poses T_1, T_2 and T_3. This state grows at every timestep, as a new pose is added and possibly new landmarks if new trees are detected.

The SLAM problem as defined above can be formulated as a probabilistic estimation problem where the past and current states of the robot, as well as landmarks and other support quantities, are all treated as random variables with a joint probability distribution [46]. The properties of this joint distribution is determined by the collection of sensor measurements received during operation. The *posterior* distribution, denotes the probability

distribution over the states, given the measurements. In the remainder of this section, we will see how finding the maximum of this posterior, in what is known as *maximum a posteriori* or MAP estimation, leads to an accurate estimate of the robot states. We will see how a Bayesian formulation allows us to find the MAP estimate by optimizing over an objective based on *likelihood functions* and how this can be decomposed through factorization to a graphical model known as a *factor graph*. Finally, we will discuss how nonlinear optimization must be employed to solve the factor graph due to the nonlinearities inherent in robot localization.

The section is heavily based on section 2.3 in [47] and both formulations and notation follows this quite closely. Additionally, the excellent tutorial on factor graphs for robot perception by Daellert and Kaess [13] provides the foundation for the theory in this section.

3.2.1 Bayesian Formulation

In general, the system states in the SLAM problem can be defined as a set of state variables $\mathcal{X}$ containing poses and support variables such as landmark locations. As an example, for the toy SLAM problem presented in figure 3.1, $\mathcal{X}$ would be the set

$$\mathcal{X} = \{\mathbf{l}_1, \mathbf{l}_2, T_1, T_2, T_3\}. \tag{3.2.1}$$

Likewise, sensor measurements are all grouped together into a set $\mathcal{Z}$. The posterior distribution is then defined as $p(\mathcal{X}|\mathcal{Z})$ and the MAP estimate as the value of $\mathcal{X}$ that maximizes this posterior,

$$\mathcal{X}^{MAP} = \arg\max_{\mathcal{X}} p(\mathcal{X}|\mathcal{Z}). \tag{3.2.2}$$

By application of Bayes' rule, the posterior can be written as

$$p(\mathcal{X}|\mathcal{Z}) = \frac{p(\mathcal{Z}|\mathcal{X})p(\mathcal{X})}{p(\mathcal{Z})}. \tag{3.2.3}$$

Because the measurements $\mathcal{Z}$ are given, this can be further simplified into the proportionality expression

$$p(\mathcal{X}|\mathcal{Z}) \propto p(\mathcal{Z}|\mathcal{X})p(\mathcal{X}). \tag{3.2.4}$$

Note that forgoing the $p(\mathcal{Z})$ term does not change the maximum, meaning that

$$\mathcal{X}^{MAP} = \arg\max_{\mathcal{X}} p(\mathcal{Z}|\mathcal{X})p(\mathcal{X}). \tag{3.2.5}$$

As is standard in the estimation literature [13], we too will define the notion of a *likelihood function* as any function $l(\mathcal{X}; \mathcal{Z})$[1] that is proportional

[1]The $\mathcal{X}$ is placed before the $\mathcal{Z}$ in this notation to indicate that it is a function of the states, while the measurements are fixed. The measurements define the function. They are not an input to it.

to $p(\mathcal{Z}|\mathcal{X})$. With the likelihood function we can rewrite the proportionality expression for the posterior as

$$p(\mathcal{X}|\mathcal{Z}) \propto l(\mathcal{X};\mathcal{Z})p(\mathcal{X}), \tag{3.2.6}$$

and the maximum, still unchanged, as

$$\mathcal{X}^{MAP} = \arg\max_{\mathcal{X}} l(\mathcal{X};\mathcal{Z})p(\mathcal{X}). \tag{3.2.7}$$

A key property of likelihood functions which will become important later is that, unlike probability distributions, they do not need to be normalized and sum to 1.

We will shortly see how this formulation leads to the graphical framework of factor graphs, but will first consider a different graphical model known as a Bayesian network, or Bayes net. We consider this model both because it serves as a stepping stone towards factor graphs but also because it is central to the iSAM2 [28] algorithm which we will discuss in Section 3.4.1.

Formally, a Bayes net encodes a joint probability distribution with nodes representing random variables and directed edges representing conditional dependence relationships. We can find a Bayes net for the toy SLAM problem by considering the joint distribution $p(\mathcal{X}, \mathcal{Z})$. Because the measurements are assumed independent, the joint distribution factorizes into

$$\begin{aligned} p(\mathcal{X}, \mathcal{Z}) = p(T_1) \\ &\times p(\mathbf{z}_1|T_1, T_2)p(\mathbf{z}_2|T_2, T_3) \\ &\times p(\mathbf{z}_3|T_1, \mathbf{l}_1)p(\mathbf{z}_4|T_2, \mathbf{l}_1)p(\mathbf{z}_5|T_3, \mathbf{l}_2). \end{aligned} \tag{3.2.8}$$

Here, the $p(T_1)$ term corresponds to the prior on the first pose and the remaining conditional terms all correspond to measurements. The terms on the second line represent integrated pose-to-pose motion measurements which depend on two consecutive poses, forming a Markov chain. Likewise, those on the third line represent landmark measurements and hence depend on the landmark location and the pose it was observed in. This results in the Bayes net shown in figure 3.2.

3.2.2 Factor Graphs

The Bayes net model provides an intuitive way to model the conditional dependencies between variables, but as we will now see, *factor graphs* are a better language specifically for SLAM. In the Bayes net, measurements and system states are all treated as random variables in the network, even though measurements are given and should be expressed as parameters, rather than variables in the optimization problem. Additionally, as we have already seen, we commonly need to consider likelihood functions that are not properly normalized probability distributions, and often are far from linear and Gaussian.

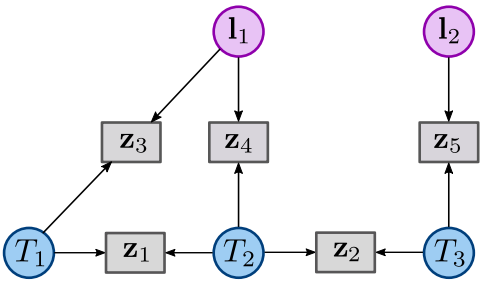


Figure 3.2: Toy SLAM problem represented as a Bayes net. Circle nodes represent variables and rectangles represent measurements. The figure is adapted from [13].

To illustrate how non-Gaussian likelihood functions arise even under Gaussian measurement noise, consider a bearing-only measurement $z \in (-\pi, \pi]$ made by a sensor stationed at the origin measuring a landmark $\mathbf{l} = [l_x, l_y] \in \mathbb{R}^2$. Because the sensor is at the origin, we do not need to consider its translational or rotational component in relation to the landmark. Instead the measurement model becomes simply,

$$z = h(\mathbf{l}) + w = \operatorname{atan2}(l_x, l_y) + w. \tag{3.2.9}$$

Here, $h(\,\cdot\,)$ denotes the measurement model while atan2 is the two-argument arctangent function. The measurement noise $w \sim \mathcal{N}(0, \sigma^2)$ is zero-mean Gaussian distributed with covariance σ^2. The likelihood in this case is any function proportional to $p(z|\mathbf{l})$. Conditioning on the landmark like this results in the following Gaussian distribution

$$p(z|\mathbf{l}) = \mathcal{N}(h(\mathbf{l}), \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big\{ -\frac{1}{2} \| \operatorname{atan2}(l_x, l_y) - z \|^2_{\sigma^2} \Big\}. \tag{3.2.10}$$

This distribution is Gaussian in z, but recall that z is a fixed measurement, and we are really interested in looking at the distribution as a likelihood function $l(\mathbf{l}; z)$ of the variable $\mathbf{l}$, with z treated only as a parameter. In this regard, (3.2.10) is clearly nonlinear and non-Gaussian, and could not be included into a Bayes net. There is no problem however including this in a factor graph, as we will now see.

The reason factor graphs are a better fit in this case is because they are more general than Bayes nets and can model any factorizable function. This means that we can define the factor graph as a function $\phi(\mathcal{X})$ proportional to the posterior $p(\mathcal{X}|\mathcal{Z})$ and maximise it to obtain the MAP estimate $\mathcal{X}^{MAP}$.

By conditional probability, the posterior can be obtained from the joint distribution as

$$p(\mathcal{X}|\mathcal{Z}) = \frac{p(\mathcal{X}, \mathcal{Z})}{p(\mathcal{Z})}. \tag{3.2.11}$$

Again, since the measurements are given, we can like in (3.2.6) rewrite the

posterior as a proportionality expression with likelihood functions, i.e.

$$\begin{aligned} p(\mathcal{X}|\mathcal{Z}) \propto\ & p(T_1) \\ & \times l(T_1, T_2; \mathbf{z}_1) l(T_2, T_3; \mathbf{z}_2) \\ & \times l(T_1, \mathbf{l}_1; \mathbf{z}_3) l(T_2, \mathbf{l}_1; \mathbf{z}_4) l(T_3, \mathbf{l}_2; \mathbf{z}_5). \end{aligned} \tag{3.2.12}$$

The factorized function of likelihoods and priors on the right hand side of this expression, is the function we wish to express as a factor graph. We define this function as

$$\begin{aligned} \phi(\mathcal{X}) = & \phi_1(T_1) \\ & \times \phi_2(T_1, T_2) \phi_3(T_2, T_3) \\ & \times \phi_4(T_1, \mathbf{l}_1) \phi_5(T_2, \mathbf{l}_1) \phi_6(T_3, \mathbf{l}_2) \end{aligned} \tag{3.2.13}$$

and note that again, by proportionality, the maxima of the two objective functions coincide, so that the MAP estimate is

$$\mathcal{X}^{MAP} = \arg\max_{\mathcal{X}} p(\mathcal{X}|\mathcal{Z}) = \arg\max_{\mathcal{X}} \phi(\mathcal{X}). \tag{3.2.14}$$

As with likelihood functions, the factors ϕ_i of this function do not take measurements as arguments, and the notation reflects this. The measurements instead *shape* the factor itself. This is why factors like ϕ_2 and ϕ_3 in (3.2.13) are non-equal, $\phi_2 \neq \phi_3$, even though they both take two poses as arguments. Figure 3.3 shows the factor graph for the toy SLAM problem, as given in (3.2.13).

When referring to factor graphs in the remainder of this text we implicitly mean factor graphs in the sense described above: a factorization of a function $\phi(\mathcal{X})$ proportional to the posterior $p(\mathcal{X}|\mathcal{Z})$. For completeness however, we will state the more general definition of a factor graph, as defined in [13]. A factor graph is an undirected bipartite graph $(\Phi, \mathcal{X}, \mathcal{E})$ that encodes a particular factorization of an arbitrary function $\phi(\mathcal{X})$. It consists of factor nodes $\phi_i \in \Phi$, variable nodes $\mathbf{x}_j \in \mathcal{X}$ and edges $e_{ij} \in \mathcal{E}$. Edges are always

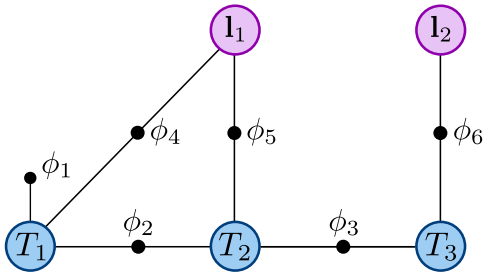


Figure 3.3: Factor graph representation of the toy slam problem. Large colored nodes denote variables and small black nodes denote factors. The factor nodes are labeled for clarity here, but this is often omitted. The figure is adapted from [13].

between a factor and a variable node and never between two factor nodes or two variable nodes. A factor ϕ_i is always a function of the variables adjacent to it, $\mathcal{X}_i \subset \mathcal{X}$. The factor graph therefore defines the factorization [13]

$$\phi(\mathcal{X}) = \prod_{\phi_i \in \Phi} \phi_i(\mathcal{X}_i). \tag{3.2.15}$$

As we have now seen, factor graphs allow us to pose the MAP optimization problem as an maximization problem of a factorized function $\phi(\mathcal{X})$. This graphical model provides an intuitive understanding of the variables involved and the dependencies between them. We have seen how defining factors allows us to conceptualize measurements as constraints rather than variables. This allows us to pose the MAP optimization problem as that of smoothing a factor graph, an approach that has come to be known as the smoothing and mapping solution to SLAM, that is now prevalent in many state-of-the-art SLAM pipelines [11] [12] [26] [28]. But optimizing the factor graph is often non-trivial due to the highly nonlinear factors that arise in the SLAM problem. In the following section, we will therefore describe how nonlinear optimization techniques are employed to obtain the MAP estimate.

3.2.3 Nonlinear Least-Squares and Nonlinear Optimization

Finding the MAP estimate by maximizing the nonlinear factor graph $\phi(\mathcal{X})$ is easier said than done, because it is a function of often hundreds or thousands of variables, often highly nonlinear and contains many local maxima. In this section, we will see how, under the assumption of Gaussian measurement noise, the problem is equivalent to a nonlinear least-squares minimization problem. We will then see how to solve this problem with nonlinear optimization methods that repeatedly linearize the objective function, solve the resulting linearized local problem, and update the global solution with the result.

Nonlinear Least-Squares

It is not difficult to show that under Gaussian measurement noise, the MAP optimization problem is equivalent to a least-squares optimization problem. With Gaussian measurement noise, all factors can be written on the form

$$\phi_i(\mathcal{X}_i) \propto \exp\Big\{ -\frac{1}{2}\|h_i(\mathcal{X}_i) - \mathbf{z}_i\|^2_{\Sigma_i} \Big\}, \tag{3.2.16}$$

where h_i is an arbitrary, possibly nonlinear measurement model and $\|\cdot\|^2_{\Sigma_i}$ is the squared Mahalanobis distance for a measurement with covariance Σ_i. Taking the logarithm of this turns the product of exponentials into a sum of exponents, i.e.

$$\log \prod_{\mathbf{z_i} \in Z} \exp\Big\{ -\frac{1}{2}\|h_i(\mathcal{X}_i) - \mathbf{z}_i\|^2_{\Sigma_i} \Big\} = -\frac{1}{2} \sum_{\mathbf{z}_i \in Z} \|h_i(\mathcal{X}_i) - \mathbf{z}_i\|^2_{\Sigma_i}. \tag{3.2.17}$$

This operation does not change the maximum because the logarithm is a monotonically increasing function. Finally, scaling the result by $-\frac{1}{2}$ turns this into a nonlinear least-squares minimization problem

$$\mathcal{X}^{MAP} = \arg\min_{\mathcal{X}} \sum_{\mathbf{z_i} \in Z} \|h_i(\mathcal{X}_i) - \mathbf{z}_i\|^2_{\Sigma_i} \tag{3.2.18}$$

which can be solved by nonlinear optimization methods such as Gauss-Newton or Levenberg-Marquardt.

Gauss-Newton

Gauss-Newton (GN) is an optimization method for solving nonlinear least-squares problems that takes curvature into account by approximating the *Hessian*. The basic algorithm iterates a sequence of three steps in order to minimize the sum of squared residuals in (3.2.18). First, the objective function is linearized at the current estimate to obtain a standard, *linear* least-squares problem. Then, this linearized problem is solved using the *normal equations* to obtain an update vector. Finally, the current estimate is updated with the update vector, and the process repeats from the beginning.

For the linearization, we consider a single n-dimensional i-th residual

$$h_i(\mathcal{X}_i) - \mathbf{z}_i. \tag{3.2.19}$$

We denote the current estimate as $\mathcal{X}^0$, and the n-dimensional component of this corresponding to the i-th residual as $\mathcal{X}_i^0$. With this, a linearized version of the residual at $\mathcal{X}_i^0$ is obtained as the first-order Taylor approximation

$$h_i(\mathcal{X}_i) = h_i(\mathcal{X}_i^0 + \Delta_i) \approx h_i(\mathcal{X}_i^0) + H_i\Delta_i. \tag{3.2.20}$$

Here, H_i is the *measurement Jacobian* calculated at $\mathcal{X}_i^0$ as

$$H_i \triangleq \left.\frac{\partial h_i(\mathcal{X}_i)}{\partial \mathcal{X}_i}\right|_{\mathcal{X}_i^0}. \tag{3.2.21}$$

Note that this linearization is possible only when $\mathcal{X}_i^0$ lies on an Euclidean vector space. This is not always the case for the types of quantities we need to optimize over in SLAM. In particular, the rotational component of a 3D robot pose lies on the nonlinear manifold of the *special orthogonal group* $SO(3)$. One way to handle this is instead to define and linearize a measurement model in the *tangent space* of the manifold. Because $SO(3)$ is a Lie group, this can be done using the Lie-group *exponential map* $\mathrm{Exp}(\cdot)$ and *logarithmic map* $\mathrm{Log}(\cdot)$. We refer the reader to [44] for the Lie-group machinery needed for this, and [13] for how it is incorporated in Gauss Newton or other nonlinear optimization methods.

Inserting the linearized measurement model (3.2.20) into the nonlinear least-squares objective (3.2.18), we obtain a linear least-squares problem

$$\begin{aligned}\Delta^* &= \arg\min_{\Delta} \sum_{\mathbf{z}_i \in Z} \|h_i(\mathcal{X}_i^0) + H_i\Delta_i - \mathbf{z}_i\|^2_{\Sigma_i} \\ &= \arg\min_{\Delta} \sum_{\mathbf{z}_i \in Z} \|H_i\Delta_i - (\mathbf{z}_i - h_i(\mathcal{X}_i^0))\|^2_{\Sigma_i}.\end{aligned} \tag{3.2.22}$$

Here, we define Δ^* as the optimal update step obtained from solving the complete linear least-squares problem. Note that Δ^* and Δ here are *not* concatinated vectors of the individual update vectors Δ_i, as these can have overlapping variables. This is because any variable $\mathbf{x}_j \in \mathcal{X}$ may be involved in multiple factors $\phi_i \in \Phi$.

Before we can solve the linear least-squares problem in (3.2.22) however, we need to formulate it as a *standard* least-squares problem. To do this, we use the same trick as in [13] which amounts to whitening the measurement noise: The Mahalanobis norm for an error term $\mathbf{e}$ can be written as

$$\|\mathbf{e}\|^2_{\Sigma_i} = \mathbf{e}^\top \Sigma_i^{-1}\mathbf{e} = \left(\Sigma_i^{-1/2}\mathbf{e}\right)^\top \left(\Sigma_i^{-1/2}\mathbf{e}\right) = \|\Sigma_i^{-1/2}\mathbf{e}\|_2^2. \tag{3.2.23}$$

Thus, by defining substitutions

$$\begin{aligned}A_i &= \Sigma_i^{-1/2} H_i, \\ \mathbf{b}_i &= \Sigma_i^{-1/2}\Big(\mathbf{z}_i - h_i(\mathcal{X}_i^0)\Big),\end{aligned} \tag{3.2.24}$$

we obtain an equivalent least-squares problem in standard form

$$\Delta^* = \arg\min_{\Delta} \Sigma_i \|A_i\Delta_i - \mathbf{b}_i\|_2^2. \tag{3.2.25}$$

The $n \times n$ matrix A_i and the n-length vector $\mathbf{b_i}$ here are respectively referred to as the *whitened* Jacobian matrix and residual (or prediction error). By collecting the A_i matrices into one large matrix and the $\mathbf{b}_i$ vectors into a large concatinated vector,

$$A = \begin{bmatrix} A_0 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_N \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_0 \\ \vdots \\ \mathbf{b}_N \end{bmatrix}, \tag{3.2.26}$$

(assuming for the moment that we have N total residuals), we obtain a standard least squared problem

$$\Delta^* = \arg\min_{\Delta} \|A\Delta - \mathbf{b}\|_2^2 \tag{3.2.27}$$

that can be readily solved by solving the normal equations. The normal equations are defined by the expression

$$(A^\top A)\Delta^* = A^\top \mathbf{b}. \tag{3.2.28}$$

The matrix on the left-hand-side here is referred to as the *information matrix* [13] and is denoted

$$\Lambda \triangleq A^\top A. \tag{3.2.29}$$

It can be seen as an approximation of the *Hessian* of the nonlinear problem. Computing the update step using this matrix hence includes information about the curvature of the function, which allows taking more accurate steps without having to compute costly second derivatives.

The solution to (3.2.28) can be obtained either by inverting Λ, or as described in [13] by factorizing it. With the solution Δ^* obtained from the linearized problem, the current estimate $\mathcal{X}^0$ is updated as

$$\mathcal{X}_i^0 \leftarrow \mathcal{X}_i^0 + \Delta^*. \tag{3.2.30}$$

This linearize-solve-update process can be continued until a convergence criteria is met, or a maximum number of iterations is reached. In applying (3.2.30), we again assume $\mathcal{X}_i^0$ lives on a vector space. For quantities on a Lie-group manifold such as $SO(3)$, the estimate can instead be updated using the exponential map

$$\mathcal{X}_i^0 \leftarrow \mathcal{X}_i^0 \cdot \mathrm{Exp}(\Delta^*), \tag{3.2.31}$$

and for arbitrary differentiable manifolds, it can be applied using a *retraction*. See [13] for more details on this.

Levenberg-Marquardt

The Levenberg-Marquardt (LM) algorithm extends upon the simple Gauss-Newton scheme by adding a damping factor $\lambda > 0$ to the normal equations. The augmented version of the normal equations solved in LM is

$$(A^\top A + \lambda \mathrm{diag}(A^\top A))\Delta^* = A^\top \mathbf{b}. \tag{3.2.32}$$

This simple modification provides two benefits over the regular GN algorithm. Firstly, the diagonal term $\lambda \mathrm{diag}(A^\top A)$ means the linear system can always be solved for the update step Δ^*, even in the case of a singular or near singular $A^\top A$. Secondly, the λ factor makes the method act like a trust-region method. Large values of λ result in smaller steps, whereas smaller values result in larger steps. This means the algorithm can act cautiously: If an update steps too far and *increases* the error rather than decreasing it, the update is rejected, and λ is increased, resulting in a more cautius step the next time around. At nominal updates where the error decreases as it should, λ is decreased. This has the effect of making steps larger in flat directions, and shorter in steeper directions [13].

3.3 Multi-Modal Localization

In many robot applications, especially those in challenging environments, the accuracy and resilience of the localization system can be increased through the use of multiple sensors. This holds especially true if the sensors provide complementary information, such as is the case for visual cameras and inertial measurement units, because situations triggering failure in one sensor might not do so in the other [31]. Such different sensor types are referred to as sensor *modalities*, and when multiple different sensor modalities are included in an estimation task, it is referred to as *multi-modal*[2].

This section describes the theory underlying multi-modal localization in a factor graph setting. We will discuss the general mechanics of tightly and loosely coupled approaches, before first specifically considering the fusion of exteroceptive sensors, and second inertial measurement units. Finally, we will put this all together and describe the workings of a visual-LiDAR-inertial localization and mapping system.

3.3.1 General Mechanics: Tight vs Loose Coupling

Multi-modal localization systems can be broadly categorized into two groups: those that are tightly coupled, and those that are loosely coupled. There is some consensus within the robotics community that tight coupling theoretically yields the most accurate and robust results, while loose coupling sacrifices some accuracy for increased flexibility and more economical computational requirements [8, 49]. These terms are sadly fuzzy terms, and what some authors refer to as a tightly coupled system, others will refer to as loosely coupled. For instance, the authors of LVI-SAM [42] refer to their system as tightly coupled, even though their factor graph optimizes over odometry estimates generated by separate LiDAR-odometry and visual-odometry systems. Comparing this to a system like VILENS [49], where LiDAR landmarks and visual landmarks are added as measurements to a combined factor graph, one may argue that this latter approach is *more* tightly coupled than the former.

Where the line goes exactly between a tightly coupled and a loosely coupled system may be hard to define, but we can still define the characteristics of a *fully* tightly coupled system and a *fully* loosely coupled system. In a system with fully tight coupling, sensor measurements are directly fed into the system and optimized over in a joint manner. Some preprocessing of measurements may still be required before they can be included in the factor graph, such as detecting image corners or performing data association, but apart from that, the sensor data is included as-is. In a system that is en-

[2]The usage of the term multi-modal here should not be confused with multi-modal probability distributions, i.e. distributions with more than one *mode* (informally, having multiple peaks).

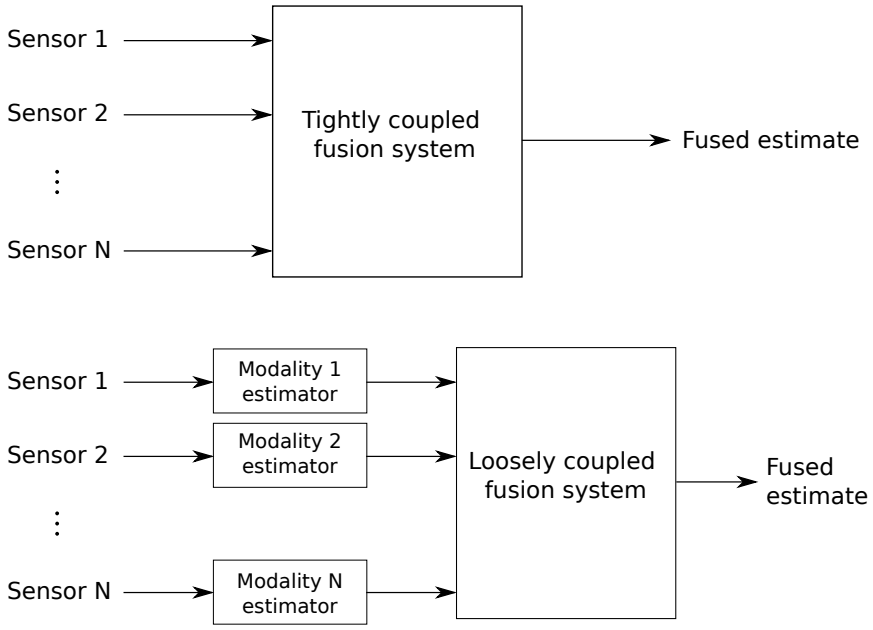


Figure 3.4: Tight (top) vs loose (bottom) coupling in sensor fusion systems. The tightly coupled system fuses measurements directly from the sensors, whereas the loosely coupled system fuses estimates from individual sensor-spesific estimators.

tirely loosely coupled however, sensor data from each individual sensor is fed into a separate sensor-specific estimator, the output of which is included as factors in the joint factor graph. Figure 3.4 illustrates this in block-diagram form.

3.3.2 Loosely Coupled Exteroceptive Modality Fusion with Pose-Graphs

For robot localization there are typically two types of sensors: *exteroceptive* sensors, sensors that measure the external environment of the robot, and *proprioceptive* sensors, which are sensors that measure things internal to the robot. Among the exteroceptive type, we find sensors like cameras, LiDARs, sonars and radar. In the proprioceptive group on the other hand, we have accelerometers, gyroscopes and wheel encoders. A successful localization system can not only rely on proprioceptive sensors, as when integrating the measurements over time they quickly accumulate significant drift. Exteroceptive sensors however, anchor the robot in its surrounding environment.

Accuracy and robustness of the localization task can be improved by fusing the measurements of multiple complementary exteroceptive sensors. We will in this and the subsequent section therefore describe the major paradigms for how multiple exteroceptive modalities can be fused. We will first look at the loosely coupled approach, where each sensor stream is pro-

cessed by an individual localization system, before the outputs of these systems are incorporated into a pose graph and solved for the fused result. In the next section, we will contrast this with the more tightly coupled approach where each exteroceptive modality produces landmarks that are included into a bundle-adjustment optimization problem formulated as a factor graph.

Loosely Coupled Odometry Systems and Pose-Graph Optimization

The simplest, most plug-and-play approach to multi-modal exteroceptive fusion is to loosely couple two or more odometry systems and combine their results as factors in a pose-graph. An odometry system such as LOAM, a LiDAR odometry system, or ROVIO, a visual odometry system, outputs at each timestep an estimate of the robot pose. Since such systems will inevitably drift, the two odometry estimates will increasingly disagree with each other. It therefore makes little sense to include the absolute pose estimates directly into the factor graph. A better approach is to instead compute the relative transformations between consecutive poses and include these as *between-factors*. We will now see how to create such a between-factor, how to include it in the factor graph, as well as how to handle when the different odometry estimates are unsynchronized and arrive at different rates.

In a pose-graph setting, a pose between-factor is a factor $\phi_i(T_k, T_{k+1})$ between two pose variables $T_k \in SE(3)$ and $T_{k+1} \in SE(3)$[3]. The measurement for this factor, $\mathbf{z}_i \in SE(3)$ provides a transformation between the aforementioned poses. The error vector for the factor is then computed as

$$\mathbf{e}_i = \log(\mathbf{z}_i^{-1} T_k^{-1} T_{k+1}) \in \mathbb{R}^6. \tag{3.3.1}$$

This error is defined on the 6-dimensional tangent space of the $SE(3)$ manifold. The mapping from the manifold is done using the Lie-group log map [11].

Adding multiple exteroceptive modalities to a pose-graph optimization pipeline requires additional considerations due to the different frequencies of the measurements. In fact, this is one of the benefits of multi-modal localization in addition to increased resilience. It allows combining high-accuracy, low-rate estimates with lower-accuracy, high-rate estimates, to produce a high accuracy, high-rate result. We will now look at two ways of structuring the pose graph for handling this, one which interleaves the measurements, and one which uses interpolation to synchronize the measurements into a single chain.

[3] In general, a between-factor can be defined for two variables of any type so long as a concept of a between transformation can be defined on the type.

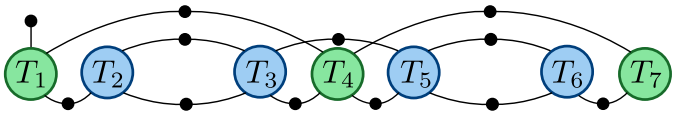


Figure 3.5: Multi-modal pose-graph with interleaved measurements for a configuration with two exteroceptive modalities. Pose variables are colored either blue or green depending on which modality they correspond to. The graph contains both between-factors between variables of the same modality, in addition to a prior factor on the first pose, and IMU factors between every consecutive pose.

Interleaved Measurement Pose-Graph

One way of fusing different-rate measurements is to form one chain of poses in the factor graph for each modality. Graphically, this may look like the graph shown in Figure 3.5. Whenever a new odometry measurement arrives, a new pose variable is inserted into the graph, and connected with a between factor to the previous pose variable from the same modality. Because estimates from the other modalities may have arrived in the meantime, this previous variable may not be the most recent pose. To handle this, the method needs extra bookkeeping variables for holding the previous pose variable for each given modality. If we define $\mathsf{S} \subset \mathbb{Z}^+$ as an index set, with an index for every modality, and let $s \in \mathsf{S}$ denote the index of a particular modality, we can let $T_{s,prev}$ be a bookkeeping variable for the most recent pose added for modality s. Adding a new odometry measurement then amounts to the following three operations

$$\Phi \leftarrow \Phi \cup \{\phi_i(T_{s,prev}, T_{k+1}), \phi_{i+1}(T_k, T_{k+1})\} \tag{3.3.2}$$
$$X \leftarrow X \cup \{T_{k+1}\} \tag{3.3.3}$$
$$T_{s,prev} \leftarrow T_{k+1}. \tag{3.3.4}$$

Here, (3.3.2) updates the factor graph with the between factor $\phi_i(T_{s,prev}, T_{k+1})$ and the IMU factor $\phi_{i+1}(T_k, T_{k+1})$, (3.3.3) adds the new pose T_{k+1} to the set of variables X, and (3.3.4) updates the bookkeeping variable $T_{s,prev}$. The IMU factors added are necessary in order to connect the two "chains" of pose variables. Without them, there would be nothing connecting the two chains, and the drift present in each individual modality would manifest itself in each chain. We will define this IMU factor properly in Section 3.3.4. Alternatively, if IMU or other proprioceptive measurements are not available, a markov motion model could be used instead.

As discussed in [47], there are some edge-cases that must be considered when applying this approach. One such case occurs if odometry estimates arrive at practically the same time and IMU measurements are not available for integration in between them. Another case is when one of the modalities becomes unavailable for a while, due to e.g. sensor failure, before later becoming available again. In that case, computing an $SE(3)$ transform from

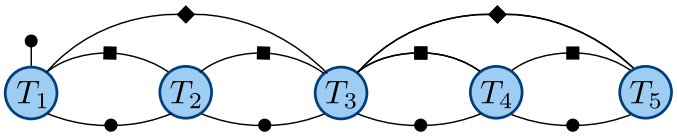


Figure 3.6: Multi-modal pose-graph with synchronied measurements for a configuration with two exteroceptive modalities. Between-factors from the main modality are shown as square nodes, whereas interpolated between-factors from the secondary modality are shown as diamonds. The graph additionally contains a prior factor on the first pose and IMU factors between every consecutive pose.

before the failure period to after, will yield wrong results, and a between factor $\phi_i(T_{s,prev}, T_{k+1})$ should not be added to the graph. For the sake of brevity however, we will not repeat the discussion of these problems and their solutions here, but instead refer the reader to [47] for the details. A completely different approach, which does not suffer from the aforementioned edge-cases, is to use interpolation to synchronize the pose-graph and create a single chain. We describe this approach in the following.

Synchronized Pose-Graph with Interpolated Poses

The other way to handle the asynchronous odometry streams, is to define one modality as the main modality and initialize new pose variables whenever its estimates arrive, while all secondary exteroceptive modalities are included as between-factors between these pose variables. An example of such a structure is shown in Figure 3.6. Because of the asynchronous nature of the estimates, the secondary odometry must be synchronized in some way so that the between-factors correspond to the same timestamps as the pose variables. One way to do this is by means of interpolation, which we will detail here.

For $SE(3)$ poses, simple linear interpolation is not possible, as the poses live on a nonlinear manifold. Instead, let us define an interpolation function $f : SE(3) \times SE(3) \times \mathbb{R}$ which can interpolate between two $SE(3)$ poses T_1, T_2 given an interpolation parameter $t \in [0, 1]$ such that

$$\begin{aligned} f(T_1, T_2, 0) &= T_1, \\ f(T_1, T_2, 1) &= T_2. \end{aligned} \tag{3.3.5}$$

Recall from the discussion of between-factors, that the between transform T_{12} can be obtained as $T_{12} = T_1^{-1} T_2$, such that

$$T_1 T_{12} = T_2. \tag{3.3.6}$$

There are several ways to define an interpolation that yields a pose which is informally "a distance t along the way to T_{12}". One way is to map T_{12} into

the tangent space and apply a linear interpolation, before mapping it back onto the manifold, i.e.

$$\exp(t \log(T_{12})). \tag{3.3.7}$$

This yields an interpolation function

$$f_{Lie}(T_1, T_2, t) = T_1 \exp(t \log(T_{12})). \tag{3.3.8}$$

Formally, this interpolates the pose along a *geodesic* in the Lie group, and in fact always yields the shortest transform [14]. An alternative interpolation is that of quaternion SLERP, which we will define as f_{SLERP}. This interpolation transforms the rotational component of the pose along a shortest-path geodesic in the group of unit quaternions $SU(2)$. However, because the unit quaternions map *doubly* to $SO(3)$ (changing the sign of the quaternion yields the same rotation), f_{SLERP} may not yield the shortest *rotation*[4][9]. Because both interpolation schemes interpolate along a geodesic, they yield rotational interpolations that have constant linear and angular velocity.

Using the interpolation function for $SE(3)$ poses f, we can for any pose variable T_k in the pose-graph, compute a corresponding measurement $\tilde{\mathbf{z}}_k$ from a secondary modality as an interpolation between the two measurements closest in time to T_k. Let t_k be the timestamp corresponding to T_k. We can then obtain the measurements from the secondary modality immediately before and after t_k, which we will denote respectively as $\mathbf{z}^-$ and $\mathbf{z}^+$ with corresponding timestamps t^- and t^+. The interpolation parameter then becomes

$$t = \frac{t_k - t^-}{t^+ - t^-} \tag{3.3.9}$$

and $\tilde{\mathbf{z}}_k$ can be obtained as

$$\tilde{\mathbf{z}}_k = f(\mathbf{z}^-, \mathbf{z}^+, t). \tag{3.3.10}$$

Note that this assumes the latest measurement z^+ exists. Practical implementations must of course wait with adding any factors based on $\tilde{\mathbf{z}}_k$ until $\mathbf{z}^+$ arrives.

With the above method for obtaining interpolated measurements for any pose variable, we can create additional between-factors from the secondary modality between any two pose variables T_k and T_{k+1}[5]. To do this, we obtain corresponding interpolated measurements $\tilde{\mathbf{z}}_k$ and $\tilde{\mathbf{z}}_{k+1}$ and compute the transform between them, yielding a between measurement

$$\tilde{\mathbf{z}}_i = \tilde{\mathbf{z}}_k^{-1} \tilde{\mathbf{z}}_{k+1}. \tag{3.3.11}$$

This is then used to create a between factor $\phi_i(T_k, T_{k+1})$ which is added to the factor graph.

[4] It yields either the "short way" with a $< 180°$ rotation, or "the long way" with a $> 180°$ rotation.

[5] These need not be in immediate succession of one another. Such additional between-factors can be added between every second pose, every third pose or every n-th pose for that matter, depending on the application and the type of modality used.

Effect of Covariance Matrices

The covariance matrix $\Sigma_i \in \mathbb{R}^{6\times 6}$ for each $SE(3)$ between-factor ϕ_i determines how much each exteroceptive modality is weighted. We see this from looking at the error term, which is minimized to obtain the MAP estimate

$$\|\mathbf{e}\|^2_{\Sigma_i} = \mathbf{e}^\top \Sigma_i^{-1} \mathbf{e}. \tag{3.3.12}$$

It is clear from this that a smaller covariance is weighted *more* in the estimation problem. Consequently, when multiple exteroceptive modalities are added with between factors in a pose-graph, the fused estimates will tend to follow most closely whichever modality has the smallest covariance matrix.

3.3.3 Tightly Coupled Exteroceptive Modality Fusion with Multi-Modal Bundle-Adjustment

The other, more tightly coupled approach to the multi-modal fusion task is to pose the optimization as a *bundle adjustment* (BA) problem with landmarks extracted from both exteroceptive modalities. Bundle adjustment is most commonly applied with camera sensors and is the foundation for many visual SLAM algorithms based on nonlinear optimization [31, 36, 38]. Recently, a more general bundle adjustment has also been applied to LiDAR sensors [33, 49]. In its simplest form, bundle adjustment amounts to jointly optimizing for the robot poses and landmark locations by minimizing the *reprojection error* between reprojected landmarks and their measurements[24]. For cameras, the reprojection error is computed by projecting the 3-dimensional landmarks into the image using a known camera model. We can define a similar notion of reprojection error for other modality types as well. By generalizing also the landmarks to other variable types than simple 3D points, for instance lines and planes, we obtain a general form of bundle adjustment capable of multi-modal fusion.

Let $\mathsf{S} \subset \mathbb{Z}^+$ be the index set of modalities that we wish to fuse, and let L_s be the index set of landmarks for modality $s \in \mathsf{S}$. Now for a particular landmark $\mathbf{l}_m$, let further $\mathcal{T}_m$ be the set of robot poses with corresponding indices T_m in which $\mathbf{l}_m, m \in \mathsf{L}_s$ was observed. In general, a landmark can produce several observations in a single timestep, such as in LiDAR BA using plane landmarks and minimizing point-to-plane distance, because several points in the point cloud will be part of the same plane. We therefore define a set of measurement indices $\mathsf{Z}_{m,k}$ for measurements of $\mathbf{l}_m$ in pose k. In visual BA, this set will always be a singleton, but for LiDAR BA and in general, it may contain multiple measurements. For the single landmark $\mathbf{l}_m$, the bundle adjustment problem then amounts to

$$\mathbf{l}_m^*, \mathcal{T}_m^* = \underset{\mathbf{l}_m, \mathcal{T}_m}{\arg\min} \sum_{k\in\mathsf{T}_m} \sum_{i\in\mathsf{Z}_{m,k}} \|\pi^s(\mathbf{l}_m, T_k) - z_i\|^2_{\Sigma_i}, \tag{3.3.13}$$

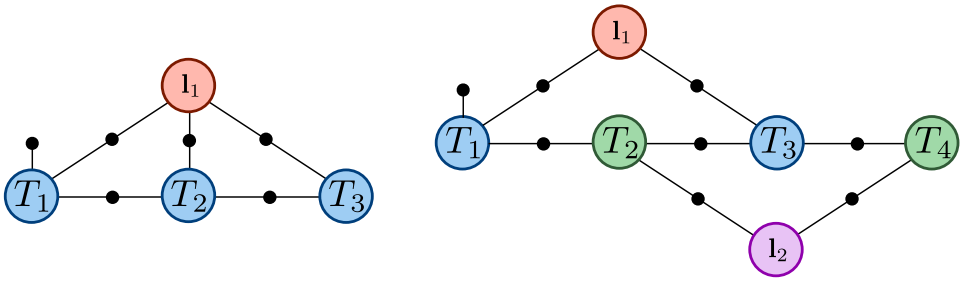


Figure 3.7: Basic example factor graph for a bundle adjustment problem with a single landmark observed from three poses **(left)** and a more general factor graph for a multi-modal bundle adjustment problem **(right)**. The different colored pose and landmark nodes correspond to different modalities. Both graphs also contains IMU factors and a prior factor on the first pose.

where Σ_i is the measurement covariance and π^s is the reprojection function for modality s which takes a landmark $\mathbf{l}_m$ and a pose T_k and produces a measurement prediction which we compare with z_i to obtain the reprojection error [24]. To extend this to the multi-modal context, we let $\mathcal{T}$ be the set of all poses and $\mathcal{L}$ be the set of all tracked landmarks, regardless of their modality. The full multi-modal bundle adjustment problem is then obtained by additionally summing (3.3.13) over the modalities and all the landmarks,

$$\mathcal{L}^*, \mathcal{T}^* = \underset{\mathcal{L},\mathcal{T}}{\arg\min} \sum_{s \in \mathsf{S}} \sum_{m \in \mathsf{L}_s} \sum_{k \in \mathsf{T}_m} \sum_{i \in \mathsf{Z}_{m,k}} \|\pi^s(\mathbf{l}_m, T_k) - z_i\|^2_{\Sigma_i}. \tag{3.3.14}$$

Since (3.3.13) and (3.3.14) are nonlinear least-squares optimization problems, they correspond nicely to nonlinear factor graphs. Figure 3.7 shows an example factor graph for both the basic version of the BA problem from (3.3.13) and the more general multi-modal BA in (3.3.14). The factors that connect landmarks to poses, $\phi_i(\mathbf{l}_m, T_k)$ all minimize the reprojection error $\pi^s(\mathbf{l}_m, T_k) - z_i$. In order include an exteroceptive modality with this scheme, three things must be defined: 1) the type of landmark $\mathbf{l}_m$ to store in the factor graph, e.g. 3D points for visual or planes or lines for LiDAR, 2) the type of measurements z_i received, e.g. 2D pixel locations for visual or 3D points for LiDAR[6], and 3) the reprojection function π^s as this is dependent on the landmark and measurement types.

Timing Considerations

The above scheme suffers from the same timing problems presented in section Section 3.3.2, in that different modalities are inherently asynchronous

[6]Minimizing point-to-plane or point-to-line distances is not the only way to formulate a LiDAR BA system. Extracting line and plane features from the raw point clouds in a pre-processing step and treating these as measurements has also proved successful [49].

in nature. The factor graph must therefore implement either an interleaved asynchronous pose graph or apply synchronization mechanisms to synchronize measurements to poses [49].

Resilience to Single-Modality Degeneracy

As explained in section Section 3.3.1, single-modality degeneracy can occur if not enough information from that modality is available to constrain the robot pose. In a loosely coupled context where each modality alone must be able to produce a valid robot trajectory, a degenerate modality must be turned off so not to deteriorate the fused result. In the multi-modal BA scheme presented here however, the measurements from the degenerate modality still provide useful information to the problem. In particular, the situation occurs if the set of landmarks for modality $s \in \mathsf{S}$ observed in pose $k \in \mathsf{T}$, $\mathcal{L}_{s,k}$ is non-empty, yet can not alone constrain all poses $\mathcal{T}_m$ for all landmarks $\mathbf{l}_m \in \mathcal{L}_{s,k}$ in (3.3.13) to a unique solution. Combined with the other modalities in S however, the combined set of visible landmarks $\mathcal{L}_k$ *does* constrain the solution. This results in a natural resiliancy to degeneracy without special treatment [49].

3.3.4 Inclusion of Inertial Measurements

So far, we have mentioned inertial measurements and IMUs (inertial measurement units) on several occasions without properly defining the characteristics of these sensors or how to include them into the factor graph. This section therefore explains the workings of IMUs and why they are beneficial to the multi-modal sensor fusion problem.

An IMU is a *proprioceptive* sensor, meaning it provides information about the internal motion of the robot directly, without having to infer it from exteroceptive measurements of the surroundings. It is common for IMUs to include an *accelerometer* and a *gyroscope* [50], so when referring to IMUs in this text we mean a unit comprising these two sensor types.

The reasons for including such proprioceptive measurements in the multi-modal localization problem are plentifold. Firstly, these sensors are complementary to the exteroceptice sensors commonly used in SLAM. During rapid motion for instance, cameras suffer due to motion blur and LiDARs suffer due to point cloud distortion. An IMU can complement both these challenging scenarios by making the rapid motion of the robot observable while not introducing considerable drift due to the short-term motion [31, 53]. A second benefit of IMUs is the high measurement frequency that they often provide. This is beneficial for control algorithms since they typically require high rate estimates [40]. A third and final benefit of IMUs is that, due to improvements in IMUs of the MEMS kind (microelectromechanical systems), they are now generally available at low cost and with low footprint [50],

making them ideal for handheld devices or small, highly mobile robots such as micro aerial vehicles (MAVs).

Motion Model Formulation and Integration

An IMU measures acceleration and angular velocity of the robot, so to obtain pose estimates, the measurements must be integrated over time. This integration is the major source of the rapid drift present in inertial-only navigation and minimizing this drift requires estimating the inherent *biases* associated with the accelerometer and gyroscope. Since these biases are unknown, slowly varying quantities, they too must be estimated in addition to the robot motion [50]. We now formulate the state representation, kinematic motion model and integration required to obtain robot pose estimates while accounting for the biases. The notation and derivation lends heavily from the seminal paper about on-manifold preintegration theory by Forster et al [19].

As in [19], we define the acceleration ${}_{\mathrm{W}}\mathbf{a}(t) \in \mathbb{R}^3$ of the body expressed in the world frame W and the angular velocity ${}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{WB}}(t) \in \mathbb{R}^3$ of the robot body frame B relative to W expressed in B. The IMU measures these quantities, but its readings are corrupted by the biases $\mathbf{b}^a(t), \mathbf{b}^g(t)$ and additive white noise components $\mathbf{w}^a(t) \sim \mathcal{N}(0, \Sigma^a), \mathbf{w}^g(t) \sim \mathcal{N}(0, \Sigma^g)$ (superscripts a and g respectively denote accelerometer and gyroscope), so that the measurements become

$$ {}_{\mathrm{B}}\tilde{\mathbf{a}}(t) = R_{\mathrm{WB}}^{\top}(t)({}_{\mathrm{W}}\mathbf{a}(t) - {}_{\mathrm{W}}\mathbf{g}) + \mathbf{b}^a(t) + \mathbf{w}^a(t), \tag{3.3.15} $$

$$ {}_{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{WB}}(t) = {}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{WB}}(t) + \mathbf{b}^g(t) + \mathbf{w}^g(t). \tag{3.3.16} $$

Here ${}_{\mathrm{W}}\mathbf{g}$ is the constant, known gravity vector and $R_{\mathrm{WB}}(t)$ is the rotation of the body B expressed in the world frame W [19]. Note that both measurements are expressed in B. This is because the IMU provides measurements in relation to its own frame, and we are assuming for simplicity that the IMU frame coincides with B.

We now define the motion model which relates the robot pose $T_{\mathrm{WB}} = (R_{\mathrm{WB}}, {}_{\mathrm{W}}\mathbf{p})$ to the acceleration and angular velocity. For the linear part of the pose, we have the familiar equations

$$ {}_{\mathrm{W}}\dot{\mathbf{v}}(t) = {}_{\mathrm{W}}\mathbf{a}(t), \quad {}_{\mathrm{W}}\dot{\mathbf{p}}(t) = {}_{\mathrm{W}}\mathbf{v}(t), \tag{3.3.17} $$

where ${}_{\mathrm{W}}\dot{\mathbf{p}}(t) \in \mathbb{R}^3$ is the velocity of the robot in frame W [19]. For the rotational component, the angular velocity ${}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{WB}}(t) = [\omega_1(t), \omega_2(t), \omega_2(t)]^{\top}$ is applied as a skew-symmetric matrix

$$ [{}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{WB}}(t)]_{\times} = \begin{bmatrix} 0 & -\omega_3(t) & \omega_2(t) \\ \omega_3(t) & 0 & -\omega_1(t) \\ -\omega_2(t) & \omega_1(t) & 0 \end{bmatrix}. \tag{3.3.18} $$

And the derivative is obtained by compounding with the current rotation, i.e. [15, 19]

$$\dot{R}_{\mathrm{WB}} = R_{\mathrm{WB}}[{}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{WB}}(t)]_{\times}. \tag{3.3.19}$$

The biases are also time varying quantities. They vary slowly and are typically modeled as a random-walk process, i.e.

$$\dot{\mathbf{b}}^{a}(t) = \mathbf{w}^{ba}(t), \quad \dot{\mathbf{b}}^{g}(t) = \mathbf{w}^{bg}(t), \tag{3.3.20}$$

where $\mathbf{w}^{ba}(t) \sim \mathcal{N}(0, \Sigma^{ba})$ and $\mathbf{w}^{bg}(t) \sim \mathcal{N}(0, \Sigma^{bg})$ are respectively the accelerometer and gyroscope continuous-time additive white noise components that drive the random-walk process [19, 50].

The motion model (3.3.17) and (3.3.19), can be discretized and integrated in order to obtain pose and velocity estimates for the robot. Assuming that ${}_{\mathrm{W}}\mathbf{a}(t)$ and ${}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{WB}}(t)$ are constant within the timespan of two IMU measurements, $[t, t+\Delta t]$, we can do as in [19] and define an Euler integration scheme

$$\begin{aligned} R_{\mathrm{WB}}(t+\Delta t) &= R_{\mathrm{WB}}(t)\exp({}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{WB}}(t)\Delta t). & {}_{\mathrm{W}}\mathbf{v}(t+\Delta t) &= {}_{\mathrm{W}}\mathbf{v}(t) + {}_{\mathrm{W}}\mathbf{a}(t)\Delta t \\ {}_{\mathrm{W}}\mathbf{p}(t+\Delta t) &= {}_{\mathrm{W}}\mathbf{p}(t) + {}_{\mathrm{W}}\mathbf{v}(t)\Delta t + \frac{1}{2}{}_{\mathrm{W}}\mathbf{a}(t)\Delta t^2 \end{aligned} \tag{3.3.21}$$

In (3.3.21), the integration is applied in the $SO(3)$ tangent space using the exponential map, but other formulations are also possible, e.g. using quaternions [31, 38]. Inserting (3.3.15) and (3.3.16) into the above, yields an integration scheme based on the IMU measurements

$$\begin{aligned} R_{\mathrm{WB}}(t+\Delta t) &= R_{\mathrm{WB}}(t)\exp(({}_{\mathrm{B}}\tilde{\boldsymbol{\omega}}_{\mathrm{WB}}(t) - \mathbf{b}^{g}(t) - \mathbf{w}^{gd}(t))\Delta t). \\ {}_{\mathrm{W}}\mathbf{v}(t+\Delta t) &= {}_{\mathrm{W}}\mathbf{v}(t) + R_{\mathrm{WB}}(t)({}_{\mathrm{W}}\tilde{\mathbf{a}}(t) - \mathbf{b}^{a}(t) - \mathbf{w}^{ad}(t))\Delta t \\ {}_{\mathrm{W}}\mathbf{p}(t+\Delta t) &= {}_{\mathrm{W}}\mathbf{p}(t) + {}_{\mathrm{W}}\mathbf{v}(t)\Delta t + \frac{1}{2}{}_{\mathrm{W}}\mathbf{g}\Delta t^2 \\ &\quad + \frac{1}{2}R_{\mathrm{WB}}(t)({}_{\mathrm{W}}\tilde{\mathbf{a}}(t) - \mathbf{b}^{a}(t) - \mathbf{w}^{ad}(t))\Delta t^2 \end{aligned}$$

Here, $\mathbf{w}^{ad}(t)$ and $\mathbf{w}^{gd}(t)$ are discrete-time variants of the continuous-time noise functions $\mathbf{w}^{a}(t)$ and $\mathbf{w}^{g}(t)$. See [19] for details of the relation between these.

The bias model can also be discretized to give

$$\begin{aligned} \mathbf{b}^{a}(t+\Delta t) &= \mathbf{b}^{a}(t) + \mathbf{w}^{bad}(t) \\ \mathbf{b}^{g}(t+\Delta t) &= \mathbf{b}^{g}(t) + \mathbf{w}^{bgd}(t). \end{aligned} \tag{3.3.22}$$

Again, $\mathbf{w}^{bad}(t)$ and $\mathbf{w}^{bgd}(t)$ are discretized versions of the corresponding continuous-time noise functions [19].

Exteroceptive Fusion with Preintegrated IMU-Factors

In the factor graphs for multi-modal exteroceptive fusion we have shown so far, such as e.g. Figure 3.5, we have included partial "IMU factors" between pose variables. We will now use the formulation above to formulate real IMU factors and show how they can be included into a multi-modal localization factor graph.

Including inertial measurements in the multi-modal factor graph requires defining new types of variables apart from the robot poses and support variables related to exteroceptive sensors. This is because the IMU integration also requires estimating the biases and velocities. This results in at least three variables for each timestep k: a pose $T_k \in SE(3)$, a velocity $\mathbf{v}_k \in \mathbb{R}^3$ and an IMU bias $\mathbf{b}_k \in \mathbb{R}^6$.

A naive way to include the IMU into the factor graph could be to add new variables for each incoming measurement into the graph, and then apply the integration scheme (3.3.22) to obtain a factor to connect with the previous set of variables. Considering however that IMUs can easily produce several hundred measurements every second, it becomes clear that this scheme would add too many variables into the graph and performance would suffer [19]. As suggested in [19, 34], a better approach is to summarize the IMU measurements between exteroceptive measurements, e.g. between camera frames.

Integrating (3.3.22) directly to form an IMU factor between frames k and $k+1$ is not sufficient. This is because, in a factor graph smoothing setting, past poses, velocities and biases could be changed as a result of the optimization. The integration in (3.3.22) would then have to be recomputed with the new values of $R_{\mathrm{WB}}(t)$, ${}_{\mathrm{W}}\mathbf{p}(t)$, ${}_{\mathrm{W}}\mathbf{v}(t)$, $\mathbf{b}^a(t)$ and $\mathbf{b}^g(t)$. This has prompted the need for a preintegrated IMU factor, which integrates the measurements between frames into a single factor that is *independent* of the linearization point. This amounts to defining the integration in a *local frame* [19]. One such scheme was first presented in [34] using Euler angles and then later refined to use on-manifold integration in [19]. Many SLAM algorithms have since made use of this on-manifold IMU preintegration scheme including [5, 38, 40, 42, 49].

For the sake of brevity, we will not include the details of Forster et. al.'s on-manifold preintegration scheme here and instead refer the reader to [19]. We will however state the resulting *preintegrated measurements* and the *preintegrated measurement model* and show schematically how the IMU factors are included in the factor graph.

Considering a set of IMU measurements obtained at times $k = i, ..., j-1$,

[19] defines the following preintegrated measurements:

$$\begin{aligned}
\Delta\tilde{R}_{ij} &= \prod_{k=i}^{j-1} \mathrm{Exp}((\tilde{\boldsymbol{\omega}}_k, \mathbf{b}_i^g)\Delta t) \\
\Delta\tilde{\mathbf{v}}_{ij} &= \sum_{k=i}^{j-1} \Delta\tilde{R}_{ik}(\tilde{\mathbf{a}}_k - \mathbf{b}_i^a)\Delta t \\
\Delta\tilde{\mathbf{p}}_{ij} &= \sum_{k=i}^{j-1} \frac{3}{2}\Delta\tilde{R}_{ik}(\tilde{\mathbf{a}}_k - \mathbf{b}_i^a)\Delta t^2.
\end{aligned} \tag{3.3.23}$$

Notice that the measurements are relative motion increments, and not defined at any particular rotation, position or velocity. The expression does however include the bias terms, which could change during the optimization. Instead of re-integrating the measurements to perform such updates, [19] applies bias updates with a first order Taylor expansion.

With the preintegrated measurements (3.3.23), [19] defines the following preintegrated measurement model:

$$\begin{aligned}
\Delta\tilde{R}_{ij} &= R_i^\top R_j \,\mathrm{Exp}(\delta\phi_{ij}) \\
\Delta\tilde{\mathbf{v}}_{ij} &= R_i^\top(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta t_{ij}) + \delta\mathbf{v}_{ij} \\
\Delta\tilde{\mathbf{p}}_{ij} &= R_i^\top(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i\Delta t_{ij} - \frac{1}{2}\mathbf{g}\Delta t_{ij}^2) + \delta\mathbf{p}_{ij}
\end{aligned} \tag{3.3.24}$$

This relates the actual poses, velocities and biases at $k = i$ and $k = j$, with the preintegrated measurements between them. The noise quantities $\delta\phi_{ij}$, $\delta\mathbf{v}_{ij}$ and $\delta\mathbf{p}_{ij}$ are obtained and propagated from the IMU noise characteristics Σ^a, Σ^g, Σ^{ba}, Σ^{bg}. See [19] for the details of this.

With the preintegrated measurement model (3.3.24), we can define a 15-DoF residual for fitting the involved poses, velocities and biases to the IMU measurements. In addition, a 15×15 covariance matrix can be defined from the propagated noise terms. Refer to [19] for the expressions for this. Together, the residual and covariance forms a 6-way combined IMU factor connecting six variables $\phi^{\mathcal{I}}_{k,k+1}(T_k, T_{k+1}, \mathbf{v}_k, \mathbf{v}_{,k+1}, \mathbf{b}_k, \mathbf{b}_{k+1})$. Figure 3.8 shows such factors connecting the poses in Figure 3.5.

3.4 Backend for Incremental Graph-Based SLAM

The theory we have presented so far has not addressed the incremental nature of SLAM. There are two aspects to this that must be considered. Firstly, because new information arrives with every timestep, the full factor graph grows ever larger. Memory and computing power is not infinite, so the backend needs a way to bound the problem size. Secondly, the SLAM problem is inherently *sparse* in nature. Indeed, in a robot exploration task, measurements in one part of the environment will not be highly correlated with

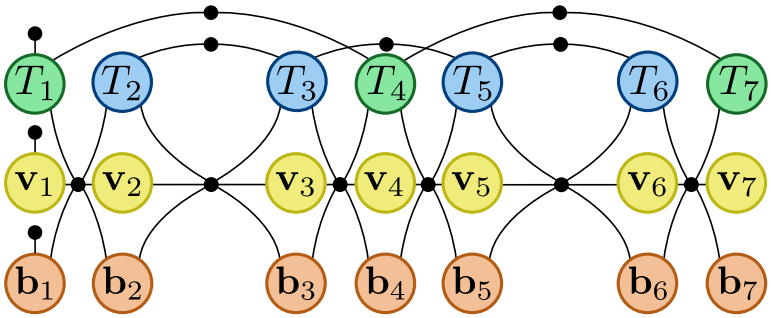


Figure 3.8: Figure 3.5 with preintegrated combined IMU factors. The factor graph includes both pose, velocity and bias variables, all connected with 6-way IMU factors. The figure is adapted from the previous work [47].

measurements from a completely different part of the environment. Exploiting this sparsity is key to achieving performant smoothing solutions to SLAM [13].

We will in this section look at two methods which facilitate incremental graph-based SLAM. First, we will consider a family of algorithms known as *incremental smoothing and mapping* algorithms which exploit sparsity in the SLAM problem to avoid relinearizing distant parts of the problem [26, 28]. Secondly, we will look at fixed-lag smoothing, a method for bounding the problem size by considering only a sliding window of states.

3.4.1 iSAM2 and the Bayes Tree

This section presents the iSAM2 algorithm by Kaess et al. [28], an algorithm which has become increasingly popular as a backend for many state-of-the-art SLAM systems based on factor graphs and nonlinear optimization [20, 40, 42, 49]. We present only a high level overview of iSAM2 and refer the reader to [28] and [13] for details. In addition, this section lends heavily from section 2.6 in [47] which presents the same topic.

Square Root SAM and iSAM with QR-Updating

To begin our discussion of iSAM2, we present its predecessor algorithms *square root SAM* (or simply SAM) [12] and iSAM [26].

The main idea in SAM lies in how the sparsity of the measurement Jacobian A for the linearized standard least-squares problem (3.2.27) is exploited to efficiently solve the normal equations (3.2.28) without directly inverting the information matrix $A^\top A$ [12]. Instead of linearizing, we factorize the information matrix as

$$A^\top A = R^\top R \tag{3.4.1}$$

where R is the upper-triangular *square root matrix*, sometimes called the *square root information matrix*, giving square root SAM its name. Such

a factorization can be obtained with e.g. *Cholesky factorization* or *QR-factorization*. With QR-factorization, instead of computing the information matrix, we factorize A itself. Assuming for now that A is $m \times n$, we obtain

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{3.4.2}$$

with $Q \in \mathbb{R}^{m \times m}$ an orthogonal matrix. Defining also $\mathbf{d} \in \mathbb{R}^n$, $\mathbf{e} \in \mathbb{R}^{m-n}$ as

$$\begin{bmatrix} \mathbf{d} \\ \mathbf{e} \end{bmatrix} = Q^\top \mathbf{b} \tag{3.4.3}$$

and exploiting the orthogonality of Q, the least squares cost becomes

$$\|A\Delta - \mathbf{b}\|_2^2 = \|Q^\top A\Delta - Q^\top \mathbf{b}\|_2^2 = \|R\Delta - \mathbf{d}\|_2^2 + \|\mathbf{e}\|_2^2. \tag{3.4.4}$$

To minimize this, we can thus ignore the separated $\|\mathbf{e}\|_2^2$ term and only solve by back-substitution the triangular system [12]

$$R\Delta^* = \mathbf{d}. \tag{3.4.5}$$

While the SAM scheme presented above benefits from being solved with simple back-substitution rather than forming and inverting the large information matrix $A^\top A$, it still suffers in an incremental setting because each iteration requires recomputing the R matrix from the augmented A matrix, and redoing the back-substitution. iSAM [26] improves on this by updating R directly using QR-updating. To add a new measurement to the system, A must be augmented with a new row $\mathbf{r}^\top$, according to

$$A' = \begin{bmatrix} A \\ \mathbf{r}^\top \end{bmatrix}. \tag{3.4.6}$$

However, noting that we can append the same row to the QR factorization,

$$A' = \begin{bmatrix} A \\ \mathbf{r}^\top \end{bmatrix} = \begin{bmatrix} Q & \\ & 1 \end{bmatrix} \begin{bmatrix} R \\ \mathbf{r}^\top \end{bmatrix}, \tag{3.4.7}$$

thus obtaining an almost triangular, Hessenberg matrix $[R, \mathbf{r}^\top]^\top$. To make this triangular again, and hence obtain a proper triangular factor matrix R', we can apply a sequence of *Givens rotations* to $[R, \mathbf{r}^\top]^\top$,

$$R' = J_n ... J_1 \begin{bmatrix} R \\ \mathbf{r}^\top \end{bmatrix}. \tag{3.4.8}$$

Informally, each Givens rotation "moves" one element from below the diagonal into the elements above the diagonal. At most n such Givens rotations

are required, but as noted in [26], the number is usually much lower since $\mathbf{r}^\top$ usually only contains a few non-zero values at the end. The notable exception to this is when a robot undergoes a loop closure by returning to a place already visited. In this case, correlations spanning far back into the history are introduced. This both requires a number of Givens rotations in general proportional to n, and introduces *fill-in* into the R matrix, making it significantly less sparse. To minimize the fill-in, a good variable *ordering* is required, and the variables must be reordered periodically to accomodate changes in the factor graph structure. In iSAM, after reordering the variables, the QR-factorization must be recomputed from scratch [26]. We will return to variable ordering while discussing iSAM2 in the following.

The QR-updating scheme presented above that represents the main workings of iSAM only apply to the linearized version of the otherwise nonlinear least-squares problem (3.2.18). As a result, the problem must be periodically relinearized and R recomputed from scratch [26]. Not only is it sub-optimal to only relinearize periodically (instead of every iteration), but it is slow when it is finally done [28]. This has led to the development of the iSAM2 algorithm [28], which is able to relinearize and reorder partially, thus avoiding periodic relinearization and reordering steps.

iSAM2 and the Bayes Tree

iSAM2 addresses the shortcomings of iSAM by departing from the realm of sparse matrix factorization and instead working directly on graphical models. The graphical model underlying iSAM2 is the *Bayes tree*, a tree representation of a Bayes net. By operating on the Bayes tree directly, relinearization and reordering can be done partially, allowing for fast computation at every iteration [28].

The Bayes tree structure is obtained by first converting the factor graph into a chordal Bayes net, using the *elimination algorithm*, then converting the result into a Bayes tree by discovering its *cliques*. The elimination algorithm, as described in [28], recursively converts a factor graph $\phi(\mathcal{X}) = \prod_i \phi_i(\mathcal{X}_i)$ into a Bayes net. Recalling from Section 3.2.1, a Bayes net defines a product of conditional and prior densities. The resulting Bayes net only contains densities on the variables $\mathcal{X}$, unlike the Bayes nets we saw in Section 3.2.1 which also contained the measurements $\mathcal{Z}$.

The elimination algorithm begins by considering the first variable in the ordering, $\mathbf{x}_j$. We define the *separator* $\mathcal{S}_j$ as the set of variables adjacent[7] to $\mathbf{x}_j$ in the graph and the joint factor $\psi(\mathbf{x}_j, \mathcal{S}_j)$ as the product of all factors involving $\mathbf{x}_j$. The joint factor $\psi(\mathbf{x}_j, \mathcal{S}_j)$ is then factorized into a density and

[7] Adjacent variables are here taken to mean variables which share a factor with $\mathbf{x}_j$, rather than the general graph-theoretic term of node adjacency. This is because as explained in Section 3.2.2, variable nodes can strictly speaking only be adjacent to factor nodes in a factor graph.

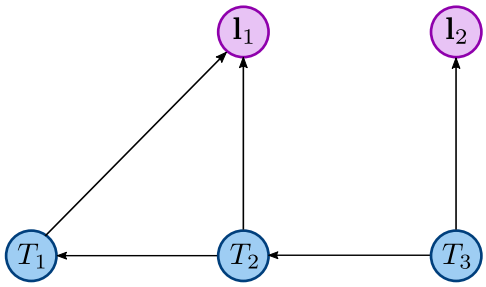


Figure 3.9: The Bayes net for the toy SLAM problem resulting from eliminating the factor graph in Figure 3.3 with the ordering $\mathbf{l}_1, \mathbf{l}_2, T_1, T_2, T_3$. The directed arrows indicate dependence relationships between variables, e.g. $\mathbf{l}_1$ depends on T_1 and T_2. The figure is adapted from [28].

a new factor on the separator $\tau(\mathcal{S}_j)$,

$$\psi(\mathbf{x}_j, \mathcal{S}_j) = p(\mathbf{x}_j|\mathcal{S}_j)\tau(\mathcal{S}_j). \tag{3.4.9}$$

The density $p(\mathbf{x}_j|\mathcal{S}_j)$ is here the first density in the Bayes net. The elimination algorithm then recursively calls on the factor graph formed by removing the factors making up $\psi(\mathbf{x}_j, \mathcal{S}_j)$ from $\phi(\mathcal{X})$ and adding in the new factor $\tau(\mathcal{S}_j)$, starting this time on the next variable in the ordering. The process completes when reaching the last variable in the ordering, at which point the separator $\mathcal{S}_j$ is the empty set, and the elimination returns a single prior on the last variable [28].

As an example, consider the toy SLAM problem presented in Figure 3.3. This factor graph, if eliminated with the ordering $\mathbf{l}_1, \mathbf{l}_2, T_1, T_2, T_3$ produces the Bayes net

$$p(\mathcal{X}) = p(\mathbf{l}_1|T_1, T_2)p(\mathbf{l}_2|T_3)p(T_1|T_2)p(T_2|T_3)p(T_3). \tag{3.4.10}$$

Figure 4 in [28] shows graphically how each step in the elimination algorithm gradually converts the factor graph for this example into the Bayes net, one variable at a time. In addition, Figure 3.9 shows the resulting Bayes net.

The Bayes net resulting form the elimination algorithm is *chordal*. This means that each cycle of more than three nodes has a *chord*, i.e. an extra edge connecting two variables in the cycle [13]. Any chordal graph can be converted to a *clique tree* or *junction tree*, by discovering its cliques [28]. The fundamental property of clique trees is that they have one node for each clique in the Bayes net. The Bayes tree is a directed variant of this type of tree that preserves information about elimination order and conditioning.

For the Bayes tree, we define three sets of variables for each node: the clique $\mathcal{C}_k$, the separator $\mathcal{S}_k$ which is the intersection $\mathcal{C}_k \cap \Pi_k$ between the clique $\mathcal{C}_k$ and its parent clique Π_k, and the frontal variables $\mathcal{F}_k = \mathcal{C}_k \backslash \mathcal{S}_k$, which are those that remain. Each node in the Bayes tree defines a conditional density

$p(\mathcal{F}_k|\mathcal{S}_k)$ and the complete Bayes tree then defines the following joint density

$$p(\mathcal{X}) = \prod_k p(\mathcal{F}_k|\mathcal{S}_k), \tag{3.4.11}$$

where for the root node, with $\mathcal{S}_k$ empty, the conditional becomes only a simple prior $p(\mathcal{F}_k)$ [28].

The constructed Bayes tree is actually equivalent to the QR-factorization performed in iSAM. As a result, the solution Δ^* can be obtained in the same way: by back-subtitution. In the Bayes tree, this amounts to one pass up from the leaf nodes to the root, in elimination ordering to define the conditional densities, and one pass down again to retrieve the optimal assignments for Δ^*. The upwards pass is already done as part of the elimination, while the downward pass is equivalent to solving the triangular system (3.4.5) with back-substitution. For that, instead of beginning at the bottom of an upper triangular matrix R and moving upwards, using the results from previous rows for subsequent computations, the back-substitution in the Bayes tree works by moving downwards from the root node, using results from previous nodes for subsequent nodes conditioned on them [28].

The Bayes tree datastructure shines in the context of incremental SLAM because it allows incremental inference and reordering, as well as fluid relinearization of variables and effective variable caching. We have already seen regular iSAM perform incremental inference. As new measurements come in, the R matrix must not be recomputed from scratch, but can instead be QR-updated to introduce the new measurements. The same applies to iSAM2. Here, only the variables affected by a new factor and those *above* them in the tree need to be reeliminated. All remaining variables downwards in the branches remain unchanged, as these were eliminated before the affected variables in the ordering [28]. This usually results in only a few variables being modified with every increment, except when a factor is added between variables in separate branches, such as in case of a loop closure. In such a case, both branches must be re-eliminated eliminated to account for the new correlation [28].

As new variables are added to the problem, or new factors between existing variables, the Bayes tree may need reordering. This is because the elimination order determines how much fill-in[8] results in the Bayes tree. Both iSAM and iSAM2 therefore makes use of the *column approximate minimum degree* ordering algorithm (COLAMD)[10] to obtain an ordering that reduces fill-in. Because in SLAM, new measurements are more likely to affect recent states, they both impose constraints on the ordering, forcing the most recent states to the end of the ordering and hence the top of the tree. This constrained COLAMD (CCOLAMD) algorithm leads to faster inference times at the cost of slightly more fill-in [26, 28]. The benefit of the

[8]The concept of fill-in here is equivalent to fill-in in the R matrix for iSAM and a denser (more connected) Bayes net for iSAM2, hence the shared terminology

Bayes tree for ordering, is that reordering can be performed on only a subset of the variables that need it, leaving the remaining variables untouched. iSAM2 does this along with the re-elimination in the incremental inference steps. That way, those variables that need reordering to better reflect new dependencies between them, are reordered to reduce fill-in, while the cost of doing a complete batch reordering step is avoided [28].

A major benefit of the Bayes tree is that linearization can be performed in a fluid manner, rather than a batch step as in iSAM. This can be done because iSAM2 keeps track of the linear least-squares update vector Δ_j, for each variable $\mathbf{x}_j$ in the factor graph. The linearization of the nonlinear factors involving $\mathbf{x}_j$ grow increasingly inaccurate the larger Δ_j grows. For Δ_j close to zero however, the linearization is still good enough. This suggests only relinearizing cliques with a Δ_j above a certain threshold. iSAM2 therefore defines a relinearization threshold β. Cliques containing a Δ_j with $|\Delta_j| \geq \beta$ are relinearized, along with any cliques above them in the tree. All cliques downwards towards the leaves however are left unmodified [28].

The final benefit of the Bayes tree is that updates to variable values tend to only have a local effect, affecting mostly variables in close proximity in the tree. New variables and factors are usually added to the top of the tree, and this may not significantly affect variables far down in the branches. iSAM2 exploits this for computing the update vector Δ, where only the Δ_j vectors for variables that significantly change are computed and used for updating the linearization point or providing estimates, while the rest can keep their cached values. This is achieved through a *wildfire* scheme, in which computation of the Δ vector begins from the top of the tree and moves downwards into the branches. If a clique in a branch contains no variables for which the *change* in Δ_j is larger than the wildfire threshold α, propagation down that branch stops, and the remaining variables down to the leaves keep their cached values. That way, large parts of Δ are obtained from cached values, with no computation needed.

3.4.2 Fixed Lag Smoothing

While the Bayes tree of iSAM2 significantly improves the feasibility of solving large real-life SLAM problems, it too can not grow forever, given finite computing resources. A standard way to constrain the problem size is to apply *marginalization*, by marginalizing out old variables, hence bounding the size of the factor graph. When this is applied in an incremental smoothing setting, it leads to an approach known as *fixed-lag-smoothing*, where we keep only a sliding window of poses and marginalize out the rest [13]. In this section we describe this approach in general as well as how it can be applied within the framework of iSAM2 to yield an *incremental fixed-lag smoother*.

Marginalization

Marginalization is the process of obtaining a *marginal* probability distribution from a joint distribution. In general, given two random variables x and y with a joint probability density function $p(x, y)$, the marginal distribution over y can be obtained by integrating over x, i.e.

$$p(y) = \int_x p(x, y). \tag{3.4.12}$$

We often refer to this as marginalizing *out* x. In the Gaussian case, (3.4.12) is easy to compute. If $p(x, y)$ has mean $\boldsymbol{\mu}$ and covariance Σ, according to

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{xy}^\top & \Sigma_{yy} \end{bmatrix}, \tag{3.4.13}$$

then the marginal distribution can be read directly from the matrix, [13]

$$p(x) = \mathcal{N}(\mu_x, \Sigma_{xx}). \tag{3.4.14}$$

As we will now see, marginalizing out variables is simple also in the square root factor matrix form used in iSAM2 or the equivalent Bayes net produced by the elimination algorithm in iSAM2, but—crucially—only for *some* of the variables. Which variables are easy to marginalize depends on the variable ordering. To see why, notice that in the *information* representation of the Gaussian probability distribution,

$$p(x, y) = \mathcal{N}(\Lambda^{-1}\boldsymbol{\eta}, \Lambda^{-1}) = \mathcal{N}\left(\Lambda^{-1} \begin{bmatrix} \eta_x \\ \eta_y \end{bmatrix}, \begin{bmatrix} \Lambda_{xx} & \Lambda_{xy} \\ \Lambda_{xy}^\top & \Lambda_{yy} \end{bmatrix}^{-1}\right), \tag{3.4.15}$$

the information matrix of the marginal distribution $p(y)$ is given by the *Schur complement* of Λ_{xx}, i.e.

$$\Lambda_{yy} - \Lambda_{xy}^\top \Lambda_{xx}^{-1} \Lambda_{xy}. \tag{3.4.16}$$

Recall from Section 3.4.1 however that Λ can be factorized into a square root factor matrix R according to $\Lambda = R^\top R$. Because R has an upper-triangular structure, i.e.

$$R = \begin{bmatrix} R_{xx} & S_{xy} \\ 0 & R_{yy} \end{bmatrix}, \tag{3.4.17}$$

computing the Schur complement becomes trivial and the marginal information matrix of $p(y)$ is simply R_{yy}. Marginalizing out x is hence a simple matter of discarding the first row and column of R. This even generalizes to multiple variables, where marginalizing out n variables requires removing n rows from the top and n columns from the left. This however all rests on the variables to be marginalized being at the *beginning of the ordering* [13].

Another way to understand the above is to realize that R is equivalent to the Bayes net resulting from eliminating a factor graph with a given variable ordering, and that discarding rows and columns in R is the same as dropping variables from the Bayes net. As we discussed in Section 3.4.1, the Bayes net defines a joint probability distribution factorized into a set of conditional and prior distributions. For instance, the Bayes net for the toy SLAM problem shown in Figure 3.9 is defined as

$$p(\mathcal{X}) = p(\mathbf{l}_1|T_1, T_2)p(\mathbf{l}_2|T_3)p(T_1|T_2)p(T_2|T_3)p(T_3). \tag{3.4.18}$$

The variable $\mathbf{l}_1$ can be trivially marginalized out of this expression by dropping the conditional distribution $p(\mathbf{l}_1|T_1, T_2)$, because that is the only term that includes $\mathbf{l}_1$:

$$p(\mathcal{X} \setminus \mathbf{l}_1) = \int_{\mathbf{l}_1} p(\mathcal{X}) = p(\mathbf{l}_2|T_3)p(T_1|T_2)p(T_2|T_3)p(T_3). \tag{3.4.19}$$

The same applies to $\mathbf{l}_2$ and in fact any variable in a Bayes net resulting from elimination, so long as no other variables depend on it (there are no directed edges pointing *from* it) [7]. It is clear therefore why the variable ordering is important: if $\mathbf{l}_1$ was not eliminated before T_1 and T_2, the dependence relation would go the other way, and some of the terms in (3.4.18) would be conditioned on $\mathbf{l}_1$ which would make the integration in (3.4.19) non-trivial.

The operations above may give the impression that we are discarding information when marginalizing out a variable, but this is far from the case. Rather, when a variable x is marginalized out of the joint density, what remains is a density with the information from x "baked in". This makes marginalization an effective way to constrain the problem size while retaining information from old measurements. It is important to emphasize however, that the marginalization operations shown above only apply to the linear Gaussian case. In a nonlinear SLAM setting, the R matrix or the equivalent Bayes net are obtained from the linearized version of the factor graph with Gaussian densities. By marginalizing out states from them, we therefore effectively commit to a linearization point. If the Bayes net is converted back again into a nonlinear factor graph, the factors connecting to the marginalized variable will be replaced with Gaussian *marginal factors* on the remaining variables. This retains information from the marginalized variables and its involved factors, but results in an approximation of the original factor graph, meaning that the exact MAP solution can no longer be recovered [13].

Fixed-Lag Smoothing

The marginalization scheme above can be applied for SLAM in order to smooth over a sliding window of poses instead of an ever-growing factor-graph. Indeed, to implement such a fixed-lag smoother, we would marginalize

out old variables when new ones are added and the old get pushed out of the lag. The marginalization could be computed easily with the scheme described above, if we keep a representation of the factor graph as a Bayes net or a factor matrix R. This requires that the variables are ordered by time of arrival, with new variables placed at the end of the ordering. The oldest variable in the window is therefore always the first in the ordering, and can be readily marginalized. Such an ordering additionally works well in iSAM where new measurements will be added to the end of the R matrix using Givens rotations, or in iSAM2 where adding new variables at the top of the Bayes tree requires minimal re-elimination [13].

iSAM2 presents an interesting case for fixed-lag smoothing as the Bayes net is structured in form of the Bayes tree datastructure. In this case, *leaf nodes* contain the variables that can be trivially marginalized. Constraining the oldest variables to the beginning of the variable ordering has the effect of constraining them to the leaf nodes. This results in an incremental fixed-lag smoothing algorithm [7].

4 Proposed Method and Implementation

To enable resilient SLAM in sensor degraded environments, we propose a system for fusing LiDAR, visual and inertial measurements in a semi-tightly coupled factor graph. The factor graph combines odometry constraints from a *LiDAR odometry and mapping* (LOAM [53]) system, together with depth-enriched visual landmarks and preintegrated IMU measurements. The solution can detect and handle degeneracy in the LiDAR odometry to allow continued operation even in failure conditions if enough visual features are available. In this section, we present the method and its implementation.

4.1 High-Level Overview

Our system comprises two subsystems as shown in Figure 4.1 implemented as ROS (Robotic Operating System) nodes. The first is an existing research implemenation of LOAM [53], while the second system, containing our visual frontend and factor graph backend are new implementations. The visual frontend detects and tracks visual image features on which we can perform bundle adjustment. In addition, it projects LiDAR pointclouds into the image to enrich the features with depth. The backend fuses these in an iSAM2-based fixed-lag smoother implemented in GTSAM (Georgia Tech Smoothing and Mapping [11]) together with odometry measurements from LOAM and preintegrated IMU measurements. We therefore refer to this approach as having a semi-tight coupling between the sensor modalities. This is because LOAM operates entirely on its own but the bundle adjustment receives considerable aiding from the odometry and IMU measurements.

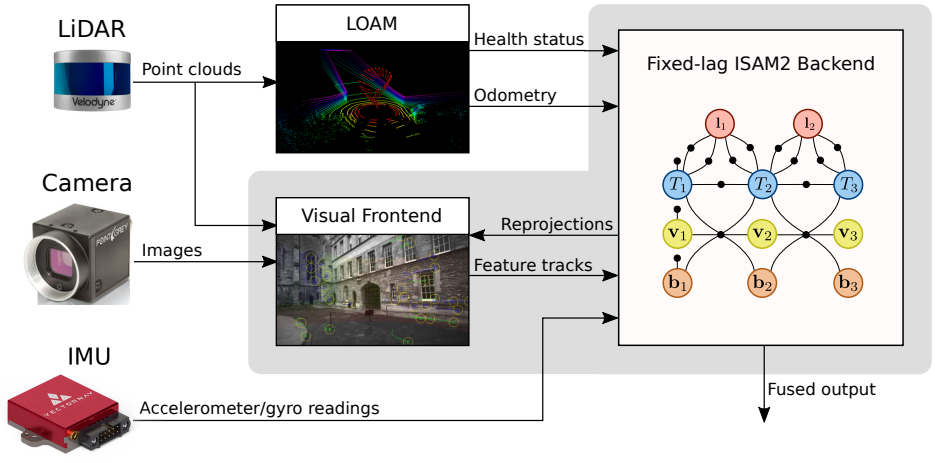


Figure 4.1: High-level overview of the complete system. The shaded area includes the part of the system that are implemented for this project.

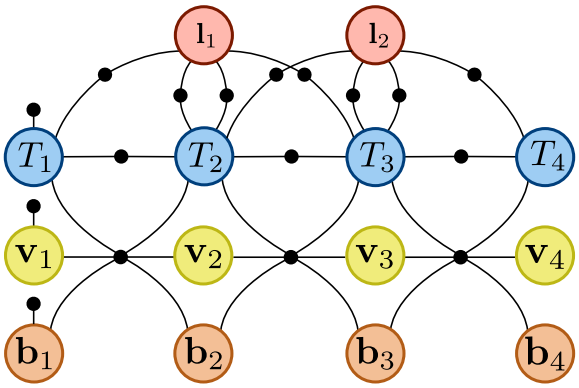


Figure 4.2: The factor graph for our multi-modal localization pipeline. Poses (in blue) are connected with both preintegrated IMU factors and between-factors from LOAM. Landmarks (in red) are connected by projection factors to every pose in which they are observed, along with optional range-factors whenever they have a depth measurement in a particular frame. IMU factors are also connected with velocity values (in yellow) and bias values (in orange).

Figure 4.2 shows the factor graph structure used by our backend. In addition to prior factors on the first pose, velocity and bias, it contains four different types of factors:

- $\phi^{\pi}(\mathbf{l}_m, T_k)$: landmark projection factors,
- $\phi^{d}(\mathbf{l}_m, T_k)$: landmark range factors,
- $\phi^{T}(T_k, T_{k+1})$: $SE(3)$ between-factors, and
- $\phi^{\mathcal{I}}(T_k, T_{k+1}, \mathbf{v}_k, \mathbf{v}_{k+1}, \mathbf{b}_k, \mathbf{b}_{k+1})$: preintegrated IMU-factors.

The graph is synchronized to camera rate, with a pose T_k added for every camera frame and LOAM odometry estimates interpolated to add as between-factors between the poses. The preintegrated IMU factors are computed according to the preintegration scheme from [19] which we presented in Section 3.3.4. IMU measurements are buffered as they arrive and preintegrated on the fly whenever a new camera frame is added. The frontend tracks image features and enriches them with depth from LiDAR pointclouds when available. These tracks are added to the graph as landmarks $\mathbf{l}_m \in \mathbb{R}^3$ when they pass a maturity test. On the landmarks we add projection factors ϕ^{π} for all pixel measurements, and range factors ϕ^{d} when depth is available. The factor graph gives rise to the following nonlinear least squares cost:

$$\sum_{k \in \mathsf{K}} \left(\sum_{m \in \mathsf{M}} \left(\|\mathbf{e}^{\pi}_{m,k}\|^2_{\Sigma^{\pi}} + \|\mathbf{e}^{d}_{m,k}\|^2_{\Sigma^{d}} \right) + \|\mathbf{e}^{T}_{k,k+1}\|^2_{\Sigma^{T}} + \|\mathbf{e}^{\mathcal{I}}_{k,k+1}\|^2_{\Sigma^{d}} \right) + \|\mathbf{e}_0\|^2_{\Sigma^0}, \tag{4.1.1}$$

where $\mathbf{e}^{\pi}_{m,k}$, $\mathbf{e}^{d}_{m,k}$, $\mathbf{e}^{\mathcal{I}}_{k,k+1}$, $\mathbf{e}^{T}_{k,k+1}$ and $\mathbf{e}_0$ respectively correspond to projection factors ϕ^{π}, range factors ϕ^{d}, between-factors ϕ^{T}, preintegrated IMU-factors $\phi^{\mathcal{I}}$ and prior factors ϕ^{0}. We will in turn describe each residual in more detail in Section 4.4.

The rest of this chapter is organized as follows: We first present our two frontends, starting with the visual in Section 4.2 and then with a brief explanation of LOAM in Section 4.3. After this, we formulate the multi-modal factor graph in greater detail in Section 4.4 and then finish with our iSAM2-based fixed-lag backend implementation in Section 4.5.

4.2 Visual Frontend

The visual frontend is responsible for taking raw images from the camera and producing accurate feature tracks that the backend can include as projection factors in its factor graph. In addition, the frontend enriches the feature tracks with depth measurements from LiDAR point clouds projected into the image.

4.2.1 Feature Extraction and Tracking

We extract strong image features using the Shi-Tomasi feature detector [43] and then track them using the Lucas-Kanade (LK) [2] feature tracker. To ensure good feature spread in the image, we perform the extraction in a grid, keeping a population of features for each grid cell and detecting new ones when a population goes below a certain threshold. Doing this is especially important since we are using the Shi-Tomasi implementation provided by OpenCV [3] which rejects features of bad quality *compared to the best quality feature in the image*. This may, in images with regions of wildly different

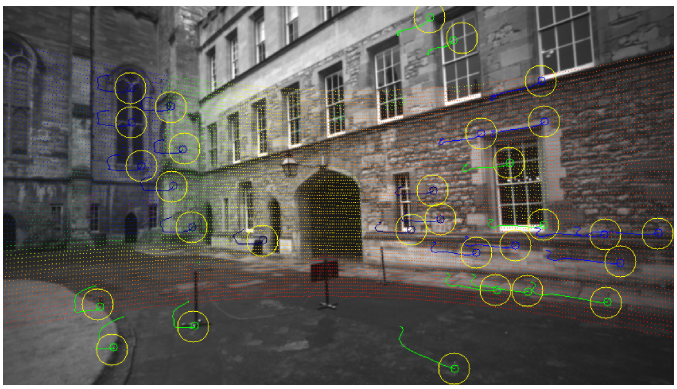


Figure 4.3: A visualization of data produced by the visual frontend showing tracked features and the LiDAR point cloud projected into the image. Features with depth are colored in blue, features without depth in green and yellow circles indicate IMU-propagated reprojected pixel predictions from the backend.

texture such as Figure 4.4, result in a high-texture region dominating the feature extraction and leaving entire regions without extracted features, unless using a gridded approach. As a final step in the extraction, we apply non-maximum suppression on feature quality to the combined set of existing and new features to achieve good spread also within each grid cell and to filter out any new features very close to existing features.

4.2.2 Outlier Rejection

The LK feature tracking may occasionally fail for some features and produce tracks that do not correspond to that feature's actual movement in image space. To combat this, we employ two outlier rejection mechanisms on the tracks: RANSAC and a reprojection test for features with a corresponding 3D landmark location $\mathbf{l}_m$.

RANSAC (RANdom SAmple Consensus) can be applied to a set of feature correspondences to reject bad LK predictions. We apply RANSAC to compute a *fundamental matrix* F for the feature correspondences, using the `findFundamenalMat` procedure in OpenCV. In addition to computing F, this rejects features that fall a certain threshold away from the epipolar line [3].

Our second outlier rejection scheme applies a simple reprojection test for landmarks which have been added to the backend. For every landmark $\mathbf{l}_m$ in the factor graph observed in frame $k+1$, we compute the simple metric

$$\|\pi(\mathbf{l}_m, \bar{T}_{k+1}) - \mathbf{z}^{\pi}_{m,k+1}\| \tag{4.2.1}$$

where $\mathbf{z}^{\pi}_{m,k+1} \in \mathbb{R}^2$ is the measured pixel and $\bar{T}_{k+1}$ is obtained from the most recent[9] pose estimate T_k by integrating IMU measurements between frames

[9] In practice the most recent pose is usually T_{k-1}, because the backend and frontend run in separate threads, so the backend is still processing T_k as the frontend requests $\bar{T}_{k+1}$.

Figure 4.4: An example image taken from the Newer College dataset [39] where some Shi-Tomasi features are vastly stronger than others. In this case, the overexposed sky along with the rectangular merlons cause nearly perfect corners with much higher quality scores than other feature candidates in the image.

k and $k+1$. We then compare the results with a predefined threshold and remove any tracks which fall outside the predicted radius.

4.2.3 Rotation-Compensated Parallax Computation

In order to triangulate a feature, it needs to have sufficient *parallax*. Common degenerate landmark configurations such as far-away points, points straight in front of the camera or points undergoing rotation-only motion, all manifest themselves in low parallax in the visual plane [24]. Projection factors can not alone constrain the 3D landmark location in such cases, and adding the track to the factor graph will hence result in an indeterminant linear system, which cannot be solved for a unique solution. Even near-degenerate cases should be avoided, as the solver may end up linearizing the factors involving the landmark at a bad linearization point which could lead the solution into a local minimum [24]. Computing the parallax of feature tracks is therefore a crucial ability for the visual frontend so it can hold back features from the backend until they have sufficient parallax.

A naive solution could compute parallax by directly computing raw pixel movement in the image plane, but this would fail to include the effect of rotations on the pixel movements. As the camera rotates, the features move across the image plane, yet this movement induces no real parallax and is of no aid to the triangulation. A real measure of track parallax should hence not include the effects of rotation. Our solution is to apply a rotation-compensation scheme similar to that of [38] to compute the parallax of feature correspondences between two camera frames. In short, the computation transforms the pixels from one frame into the previous using only the rotation between these frames. Whatever disparity remains between corresponding pixels is then only due to the translation. Far away points or points undergoing rotation-only motion will hence have very little disparity, and this is therefore a good measure for the parallax. We describe the details of this computation in the following.

Departing for a brief moment from our usual notation, we denote a 2D pixel as $\mathbf{x} \in \mathbb{R}^2$ and its corresponding 3D point as $X \in \mathbb{R}^3$. The relationship between the 2D pixel and its 3D point in the camera frame, ${}_{\mathrm{C}}X$, is given by the *intrinsic matrix* K, according to $\mathbf{x}' = K \cdot {}_{\mathrm{C}}X$ [24]. Note that $\mathbf{x}'$ is here given in *homogeneous coordinates* and must be normalized to obtain its two-parameter representation, whereas ${}_{\mathrm{C}}X$ is not, as we use the 3×3 form of the intrinsic matrix. If X was given in a different frame, e.g. the world frame W, we would have to apply the $SE(3)$ transform T_{CW} to obtain ${}_{\mathrm{C}}X = T_{\mathrm{CW}} \cdot {}_{\mathrm{W}}X$. Here, the $\cdot$ operator denotes the Lie-group action on $\mathbb{R}^3$, i.e. ${}_{\mathrm{C}}X = R_{\mathrm{CW}} \cdot {}_{\mathrm{W}}X + {}_{\mathrm{C}}\mathbf{t}_{\mathrm{CW}}$, where R_{CW} and ${}_{\mathrm{C}}\mathbf{t}_{\mathrm{CW}}$ are respectively the rotation and translation components of the transformation. Consequently, the relationship between pixels ${}_{\mathrm{A}}\mathbf{x}$ and ${}_{B}\mathbf{x}$ in camera frames A and B is given by ${}_{\mathrm{A}}\mathbf{x}' = KT_{\mathrm{AB}} \cdot K^{-1}{}_{\mathrm{B}}\mathbf{x}'$. Returning now to our usual notation, we can use

this to determine where a feature ${}_{\mathrm{B}}\mathbf{z}_m^{\pi}$ in camera frame B ends up in camera frame A if only the rotation R_{AB} is applied to it. This is given by setting ${}_{\mathrm{A}}\mathbf{t}_{\mathrm{AB}} = \mathbf{0}$, which yields

$$ {}_{\mathrm{A}}\tilde{\mathbf{z}}_m^{\pi\prime} = KR_{\mathrm{AB}}K^{-1}{}_{\mathrm{B}}\mathbf{z}_m^{\pi\prime}. \tag{4.2.2} $$

We finally compute the distance between these pixels

$$ \rho = \|{}_{\mathrm{B}}\mathbf{z}_m^{\pi} - {}_{\mathrm{A}}\tilde{\mathbf{z}}_m^{\pi}\|, \tag{4.2.3} $$

and use that as our parallax metric.

Unlike in [38] where the rotation R_{AB} is obtained from integrated IMU measurements, we obtain it from an essential matrix E computed from the feature correspondences. However, this may fail if the scene is not well explained by a fundamental matrix, a common case in scenes with low parallax. To add robustness to such situations, we apply the same heuristic as in [36] to determine if the scene is best described by a fundamental matrix F or a homography H. This amounts to computing both F and H using $RANSAC$, and then calculating a score for each respective matrix, S_F and S_H, based on how many outliers are flagged. From this we compute the comparative heuristic

$$ R_H = \frac{S_H + S_F}{S_H}, \tag{4.2.4} $$

where if $R_H > 0.45$, the scene is best described by the homography, and otherwise by the fundamental matrix [36]. If this check favors the homography, the scene is likely either planar, or all points have low parallax, so in that case we end the procedure without computing any parallaxes. To avoid increasing the computational burden too much with this scheme, we compute F and H in separate threads. The essential matrix is also obtained directly from F according to $E = K^{-1}FK$, and so does not require additional computation. Finally, since RANSAC is used to compute F and H, this doubles as the outlier rejection mechanism we described in Section 4.2.2. Even with this heuristic in place however, bad parallax estimates may slip through. Thus, for even greater robustness to bad parallax estimates, we compute the parallax for several consecutive frames, and take the median, thereby avoiding the most serious outliers. An example of this parallax computation scheme applied to feature tracks is shown in Figure 4.5 which shows far-away points being given much lower parallax than those close by.

4.2.4 Depth-Enriched Features

Bundle adjustment is a problem of nonlinear nature which, like any nonlinear optimization problem, requires a good initialization to converge. Initializing 3D landmarks by triangulating feature tracks is however non-trivial because a) tracks may not have sufficient parallax to constrain their location well and

(a) Rotation compensated parallax. **(b)** Naive uncompensated parallax.

Figure 4.5: Visualization of rotation-compensated feature parallax **(a)** vs uncompensated parallax **(b)** in a scene with forward camera motion. Green feature indicators are saturated to indicate levels of parallax: higher intensity green color means high parallax whereas grayer color means low parallax. Both computations take the median of computed values. The image is taken from the Newer College dataset [39] where many scenes have far away structures with very low parallax.

b) because triangulation requires a good estimate of the camera poses, which may not be available. A depth sensor such as LiDAR can aid the initialization in this case by enriching features with depth measurements in regions where the LiDAR point cloud overlaps with the image plane. This makes the 3D location of the landmarks in relation to the robot observable from only a single measurement, which allows better and more rapid initialization. Subsequent depth measurements can also be included as *range factors* between robot poses and landmarks to further constrain the landmark and aid convergence of the solution.

We use a simple method to associate depth measurements with features by considering a rectangular patch of LiDAR points projected into the image plane. In a separate thread, every incoming LiDAR point cloud $\mathcal{P} \subset \mathbb{R}^3$ is projected into the image frame according to

$$\mathbf{z}_i^{\mathcal{P}} = KT_{\mathrm{CL}} \cdot {}_{\mathrm{L}}\mathbf{p}_i, \quad \forall\ {}_{\mathrm{L}}\mathbf{p}_i \in \mathcal{P}, \tag{4.2.5}$$

where K is the intrinsic matrix and $T_{CL} \in SE(3)$ is the static camera-LiDAR transform. Corresponding depth measurements are computed as

$${}_{\mathrm{C}}d_i = \|{}_{\mathrm{C}}\mathbf{p}_i\| = \|T_{\mathrm{CL}} \cdot {}_{\mathrm{L}}\mathbf{p}_i\|, \quad \forall\ {}_{\mathrm{L}}\mathbf{p}_i \in \mathcal{P}. \tag{4.2.6}$$

Note our choice of computing the depth in the camera frame instead of in the body frame, i.e. ${}_{\mathrm{B}}d_i$. As we will discuss in Section 4.4.2, this makes landmark initialization easier, while we can still obtain ${}_{\mathrm{B}}d_i$ from ${}_{\mathrm{C}}d_i$ and $\mathbf{z}_m^{\pi}$. For each image feature $\mathbf{z}_m^{\pi}$, we then obtain a set of depth measurements $\mathcal{D}_m$, comprising all depth values ${}_{\mathrm{C}}d_i$ for which $\mathbf{z}_i^{\mathcal{P}}$ is within a rectangular patch centered at $\mathbf{z}_m^{\pi}$. To avoid spurious depth values, we check that $\mathcal{D}_m$ contains enough features, that they are spread out in all four quadrants of the patch and that the standard deviation is below a certain threshold. If all

these checks pass, we compute the median of $\mathcal{D}_m$ and store it as the depth measurement $\mathbf{z}_m^d$. Additionally, if some of the checks fail, we can still use the depth value as a crude rejection mechanism for far-away points, to add extra redundancy over the parallax mechanism.

In contrast to [49] we do not undistort the point cloud to the nearest camera frame, but instead simply use it as-is if the time delta is lower than some threshold (we use typical values of 100–150ms). This means that the point cloud and image may be slightly misaligned, which can lead to wrong depth measurements that the standard deviation check on $\mathcal{D}_m$ may not catch. This is especially prone to happen for features on a foreground edge, where we may sometimes get spurious depth measurements from the background, or vice versa. To combat this, we employ a simple check that consecutive depth measurements do not change by more than one meter, and require a track to have a certain number of depth measurements before being added to the graph.

4.3 Degeneracy-Aware LiDAR Odometry Frontend

To include LiDAR as a second exteroceptive modality in our factor graph, we employ the popular LOAM (lidar odometry and mapping) algorithm [53] to produce odometry estimates that we can add as between factors in the graph. While LOAM is a complete odometry system in itself, we use it as a frontend on top of our factor graph backend. The particular LOAM implementation we use comes with the capability to detect degenerate problem instances due to geometric ambiguity [29]. In this section we describe briefly how LOAM works, how it detects degenerate problem instances and how we obtain between-transforms from it which can be added to the factor graph.

4.3.1 LOAM

LOAM is a system for producing precise odometry and accurate 3D maps in real time using LiDAR. As inputs, LOAM takes raw point cloud scans together with optional IMU measurements, and outputs estimates of the LiDAR pose at two frequencies: *odometry* updates at scan rate (typically 10Hz) and higher-accuracy *mapping* updates at a lower rate (5Hz in our case). The latter also include an updated point cloud map.

The odometry updates are produced with a scan-to-scan matching method, which attempts to find the best alignment between consecutive LiDAR scans. In LOAM, this is performed as a form of ICP (iterated closest point) **[icp]** procedure that uses specific extracted geometric features in the point cloud for alignment, instead of considering the entire set of points. The features considered are sharp edges (lines) and planar surface patches (planes) and to perform the alignment, LOAM minimizes respectively a *point-to-line* metric and a *point-to-plane* metric. To obtain the odometry estimate between

two scans, LOAM first finds correspondences between LiDAR points in the new scan and the lines and planes in the previous scan. Then, it minimizes point-to-plane distance for each considered point to find the $SE(3)$ transform T^L_{k+1} between the scans. Each point $\mathbf{p}^L_{k+1,i}$ considered is transformed from the newest point cloud into the previous using T^L_{k+1}, according to

$$\tilde{\mathbf{p}}^L_{k+1,i} = (R^L_{k+1,i})^{-1}(\mathbf{p}^L_{k+1,i} - \mathbf{t}^L_{k+1,i}). \tag{4.3.1}$$

Here, $\tilde{\mathbf{p}}^L_{k+1,i}$ is $\mathbf{p}^L_{k+1,i}$ transformed into the k-th point cloud and $R^L_{k+1,i}$ and $\mathbf{t}^L_{k+1,i}$ are the rotation and translation components of $T^L_{k+1,i}$. The i subscripts here are needed because each point arrives as part of a LiDAR *sweep*, i.e. not at the same time. The i indicates that the point arrived at time t_i, and the pose $T^L_{k+1,i}$ is obtained as a *linear interpolation* of T^L_{k+1}. The resulting minimization problem is solved using the Levenberg-Marquardt (LM) algorithm. This means it repeatedly linearizes the problem and solves a resulting linear least squares problem using the damped normal equations given in (3.2.32) and restated here

$$A^\top A + \lambda \text{diag}(A^\top A)\Delta^* = A^\top \mathbf{b}, \tag{4.3.2}$$

in order to obtain an update step Δ^* for updating its current estimate. A is here the Jacobian matrix and $\mathbf{b}$ the RHS vector of the normal equations system.

In the mapping update, LOAM matches and registers the undistorted point cloud obtained from the odometry step into the world map. This is done in a way similar to the odometry update, by minimizing point-to-line and point-to-plane distances for feature-point correspondences, but this time it considers 10 times as many points, to obtain a more accurate estimate [53]. It does however not need as many iterations of the LM algorithm to converge.

4.3.2 Degeneracy Detection

LiDAR odometry will fail if the operating environment is geometrically self-similar, as is the case in e.g. long, straight tunnels or wide open fields [29, 42]. We show two such examples in Figure 4.6. The LOAM implementation we are using has built-in heuristic detection of such degeneracy when the geometry does not sufficiently constrain the scan-to-scan matching. This works by checking whether the $A^\top A$ matrix in the LM inner loop is close to singular. A singular, or near singular $A^\top A$ matrix, as we mentioned in Section 3.2.3, implies that the linearized system does not have a unique solution. The damping term $\lambda \text{diag}(A^\top A)$ added in LM means that the system can still be *solved*, but it does not mean that it will find the correct solution. LOAM then goes into a *degenerate* state, where its outputs are incorrect and should not be trusted. Thankfully, it is easy to detect a near singular matrix by checking if the eigenvalues are near zero. In the implementation we are using, a threshold of 30 is set on the eigenvalues of $A^\top A$ and if any of them

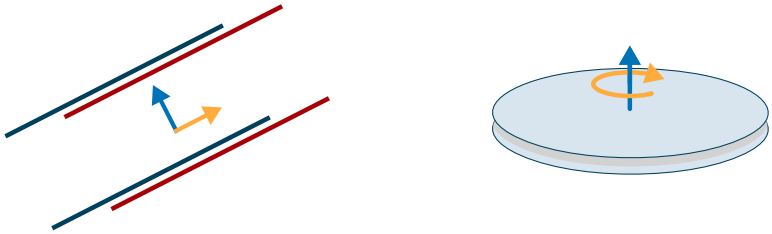


Figure 4.6: Degeneracy of the point cloud alignment problem due to self-similar geometry. Orange arrows indicate degenerate directions and blue indicate well conditioned directions. Straight corridors as shown to the **left** (top down view) are ill-conditioned along the direction of the corridor, while flat open environments as shown to the **right** have rotational ambiguity.

are lower, the odometry is marked as unhealthy. This is then broadcasted to all consumers of the odometry, so that we can avoid including it into our joint optimization problem.

4.3.3 Obtaining Interpolated Between Transforms

We use LOAM's high-accuracy mapping updates to provide estimates of the robot pose at 5Hz. To include this in our factor graph which runs at camera rate of 15Hz, we perform interpolation on the poses. In practice, this is done using the *tf2* library within ROS [18], which uses quaternion SLERP interpolation as we defined in Section 3.3.2. Our LOAM implementation makes the current pose of the LiDAR in the world frame available as a tf2 transform, which means we can query it for an interpolated pose at any timestamp, so long as we do not query past the latest mapping update.

Let us then consider a camera frame k and its subsequent frame $k+1$. We wish to obtain the $SE(3)$ transform $T_{\mathrm{B}_k\mathrm{B}_{k+1}}$ of the robot body frame B, from pose k to $k+1$. To do this, we query tf2 for the LiDAR poses T_{WL_k} and $T_{\mathrm{WL}_{k+1}}$. From this, we can obtain a between-transform of the body frame, by applying the static body-LiDAR transform T_{BL}, which we get from configuration. The between-transform we need is then obtained as

$$T_{\mathrm{B}_k\mathrm{B}_{k+1}} = T_{\mathrm{BL}}(T_{\mathrm{WL}_k})^{-1}T_{\mathrm{WL}_{k+1}}(T_{\mathrm{BL}})^{\top}. \tag{4.3.3}$$

To keep our notation consistent, we remind ourselves that this is a *measurement* in our context, and so we denote it as

$$\mathbf{z}^T_{k,k+1} = T_{\mathrm{B}_k\mathrm{B}_{k+1}}. \tag{4.3.4}$$

Note that the $(\cdot)^T$ superscript denotes that this is an $SE(3)$ pose measurement and should *not* be confused with the transpose, which we consistently denote as $(\cdot)^{\top}$. Finally, we note that we do not necessarily need to include these between transforms between every robot pose. We have for instance had good results inserting the transforms between every 5th pose instead.

4.4 Factor Graph Formulation

Now that we have seen how our two exteroceptive frontends produce measurements for the backend, we describe how the factor graph fuses all this together. Our factor graph consists of four types of variables: poses $T_k \in \mathcal{T} \subset SE(3)$, velocities $\mathbf{v}_k \in \mathcal{V} \subset \mathbb{R}^3$, IMU bias terms $\mathbf{b}_k \in \mathcal{B} \subset \mathbb{R}^6$ and landmarks $\mathbf{l}_m \in \mathcal{L} \subset \mathbb{R}^3$. Our complete state therefore becomes

$$\mathcal{X} = \Big\{\mathcal{T}, \mathcal{V}, \mathcal{B}, \mathcal{L}\Big\}. \tag{4.4.1}$$

The landmarks are connected to poses with projection factors $\phi^\pi(\mathbf{l}_m, T_k)$ and optional range factors $\phi^d(\mathbf{l}_m, T_k)$ from the visual frontend. Poses are connected with between-factors $\phi^T(T_k, T_{k+1})$ from LOAM. 6-way preintegrated IMU factors $\phi^\mathcal{I}(T_k, T_{k+1}, \mathbf{v}_k, \mathbf{v}_{k+1}, \mathbf{b}_k, \mathbf{b}_{k+1})$ connect the poses, velocities and biases. In addition, priors on the first pose, velocity and bias are necessary to eliminate gauge freedom by anchoring the trajectory to a fixed starting point in the world frame [13]. The factor graph then defines the following nonlinear least-squares optimization problem

$$\begin{aligned}\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_{k\in\mathsf{K}} \Bigg(\sum_{m\in\mathsf{M}} \Big(\|\mathbf{e}^\pi_{m,k}\|^2_{\Sigma^\pi} + \|\mathbf{e}^d_{m,k}\|^2_{\Sigma^d} \Big) \\ + \|\mathbf{e}^T_{k,k+1}\|^2_{\Sigma^T} + \|\mathbf{e}^\mathcal{I}_{k,k+1}\|^2_{\Sigma^\mathcal{I}} \Bigg) + \|\mathbf{e}_0\|^2_{\Sigma^0},\end{aligned} \tag{4.4.2}$$

We now define how variables are initialized, factors are constructed from the measurements and each residual is formed. We start with the IMU factors, before moving on to the landmarks and finally the LOAM between-factors, as we believe this is the natural steps we take when constructing the factor graph in an incremental setting.

4.4.1 Pose Initialization and IMU-Factors

We run the factor graph at camera rate, and hence initialize a pose T_{k+1}, a velocity $\mathbf{v}_{k+1}$ and a bias $\mathbf{b}_{k+1}$ for every new camera frame $k+1$. We use the preintegration scheme from [19] which we described in Section 3.3.4 to integrate our IMU measurements. We use the result of the preintegration for both the factor, and to initialize the new pose and velocity. The initial values for frame $k+1$ are hence obtained from the measurements and the previous values by setting the noise terms in (3.3.24) to zero, which yields

$$\begin{aligned}R_{k+1} &\leftarrow R_k \Delta\tilde{R}_{k,k+1}, \\ \mathbf{v}_{k+1} &\leftarrow \mathbf{v}_k + R_k \Delta\tilde{\mathbf{v}}_{k,k+1} + \mathbf{g}\Delta t_{k,k+1}, \\ \mathbf{p}_{k+1} &\leftarrow \mathbf{p}_k + R_k \Delta\tilde{\mathbf{p}}_{k,k+1} + \mathbf{v}_k \Delta t_{k,k+1} + \frac{1}{2}\mathbf{g}\Delta t^2_{k,k+1}.\end{aligned} \tag{4.4.3}$$

Here, $\Delta t_{k,k+1}$ is the time-delta between frames k and $k+1$, R_{k+1} and $\mathbf{p}_{k+1}$ are the respective rotation and translation components of T_{k+1} and $\Delta\tilde{\mathbf{v}}_{k,k+1}$, $\Delta\tilde{\mathbf{p}}_{k,k+1}$ and $\Delta\tilde{R}_{k,k+1}$, the preintegrated measurements for velocity, position, and rotation as defined in (3.3.23).

When we have inserted the new variables T_{k+1}, $\mathbf{v}_{k+1}$ and $\mathbf{b}_{k+1}$ into the graph, we connect them to the previous set of variables T_k, $\mathbf{v}_k$ and $\mathbf{b}_k$ using a 6-way combined IMU factor $\phi^{\mathcal{I}}_{k,k+1}(T_k, T_{k+1}, \mathbf{v}_k, \mathbf{v}_{k+1}, \mathbf{b}_k, \mathbf{b}_{k+1})$ as defined in [19]. This then results in a residual $\mathbf{e}^{\mathcal{I}}_{k,k+1}$ being added to the least squares problem (4.4.2). Furthermore, the covariance $\Sigma^{\mathcal{I}}_{k,k+1} \in \mathbb{R}^{15\times 15}$ for the factor is obtained from the continuous-time IMU noise characteristics Σ^a, Σ^g, Σ^{ba}, Σ^{bg} defined in Chapter 3, propagated incrementally with each timestep [19].

4.4.2 Landmark Initialization and Projection/Range Factors

After a new pose variable T_{k+1} has been added to the graph, we can initialize the landmarks and add observations for the features tracked in frame $k+1$. We use one of two mechanisms for initializing a landmark, depending on whether the landmark has depth measurements or not. This also determines what criteria we place on the landmark for performing initialization. In both cases, we require the track to be of a certain length, however we do not require tracks with depth to have parallax, as the depth makes the 3D location of it observable without triangulation.

If a track has depth, initialization proceeds as follows: We find the first feature in the track with a depth measurement, $(\mathbf{z}^{\pi}_{m,k}, {}_{\mathrm{C}}\mathbf{z}^{d}_{m,k})$, along with the pose T_k. To initialize the landmark, we then first obtain an up-to-scale vector along the ray from the camera center to the landmark

$$ {}_{\mathrm{C}}\tilde{\mathbf{l}}_m = K^{-1}\mathbf{z}^{\pi\prime}_{m,k}. \tag{4.4.4}$$

Here, we let $(\,\cdot\,)'$ denote a vector in homogeneous coordinates. The scale of ${}_{\mathrm{C}}\tilde{\mathbf{l}}_m$ will be arbitrary, depending on the third element of $\mathbf{z}^{\pi\prime}_{m,k}$. To scale it correctly, we first normalize it, and then scale it with ${}_{\mathrm{C}}\mathbf{z}^{d}_{m,k}$, i.e.

$$ {}_{\mathrm{C}}\mathbf{l}_m = {}_{\mathrm{C}}\mathbf{z}^{d}_{m,k} \cdot {}_{\mathrm{C}}\hat{\mathbf{l}}_m, \tag{4.4.5}$$

where we have let $\hat{(\cdot)}$ denote a unit vector. We can then initialize the landmark location in the world frame as

$$ {}_{\mathrm{W}}\mathbf{l}_m \leftarrow T_{\mathrm{WC}_k} \cdot {}_{\mathrm{C}}\mathbf{l}_m = T_{\mathrm{WB}_k} T_{\mathrm{BC}} \cdot {}_{\mathrm{C}}\mathbf{l}_m = T_k T_{\mathrm{BC}} \cdot {}_{\mathrm{C}}\mathbf{l}_m, \tag{4.4.6}$$

where T_{BC} is the static body-camera transform.

If the track has no depth available, we initialize it with triangulation instead. For that we use a standard DLT approach (direct linear transform) as this is built into GTSAM. We use the poses in which the landmark was observed, which we can get from the most recent iSAM2 estimate. To

limit the computational overhead, we pick a maximum of 10 poses and pixel measurements, spread out evenly across the entire track. To avoid obvious outliers or degenerate landmarks at this stage, we reject any triangulated landmarks with high average reprojection error (> 10px) and any for which the triangulation matrix is near singular (smallest singular value < 50).

After landmarks have been initialized, we can add projection and range factors to them. While a track is still active, we add projection factors at every timestep, however we only add range factors when depth is available. The projection factors minimize the reprojection error between measured pixel locations and the landmarks reprojected into the image. This gives rise to the residual

$$\mathbf{e}^{\pi}_{m,k} = \pi(\mathbf{l}_{m,k}, T_k) - \mathbf{z}^{\pi}_{m,k}, \tag{4.4.7}$$

where π is the standard projection function that computes the projection

$$\pi(\mathbf{l}_{m,k}, T_k)' = K T_{\mathrm{BC}}^{-1} T_k^{-1} \cdot \mathbf{l}_{m,k}, \tag{4.4.8}$$

normalizes the result by dividing by the last element and returns the resulting point in non-homogeneous form. This makes our approach an *indirect* approach, compared to *direct* visual odometry methods such as [1, 16, 17], which instead directly minimize pixel intensities.

To add the range factors, we must first transform the depth measurements from the camera frame into the body frame, as the range factors will minimize the distance from body poses to the landmarks. To obtain a depth measurement in the body frame ${}_{\mathrm{B}}\mathbf{z}^{d}_{m,k}$, from one in the camera frame, ${}_{\mathrm{C}}\mathbf{z}^{d}_{m,k}$, we first obtain the landmark in the camera frame ${}_{\mathrm{C}}\mathbf{l}_{m,k}$ using (4.4.4) and (4.4.5). This makes use of the two measurements $\mathbf{z}^{\pi}_{m,k}$ and ${}_{\mathrm{C}}\mathbf{z}^{d}_{m,k}$. Then we compute the range to the body pose as

$${}_{\mathrm{B}}\mathbf{z}^{d}_{m,k} = \| R_{\mathrm{BC}} \cdot {}_{\mathrm{C}}\mathbf{l}_{m,k} + {}_{\mathrm{B}}\mathbf{t}_{\mathrm{BC}} \|, \tag{4.4.9}$$

where R_{BC} and ${}_{\mathrm{B}}\mathbf{t}_{\mathrm{BC}}$ are the rotation and translation components of the static body-camera transform. The residual is then computed as

$$\mathbf{e}^{d}_{m,k} = \| R_k^{-1}(\mathbf{l}_{m,k} - \mathbf{t}_k) \| - {}_{\mathrm{B}}\mathbf{z}^{d}_{m,k}, \tag{4.4.10}$$

where R_k and $\mathbf{t}_k$ are the rotation and translation components of T_k.

For both projection and range factors, we use the Huber robust cost function [25] which adds robustness to outliers. This means, instead of minimizing the sum of squared Mahalanobis distances, $\|\mathbf{e}\|^2_{\Sigma}$ for the residuals, we minimize a sum of expressions $\rho_h(\|\mathbf{e}\|^2_{\Sigma})$, where ρ_h is defined as

$$\rho_h(x) = \begin{cases} \frac{x^2}{2} & \text{if } |x| \leq k \\ k(|x| - \frac{k}{2}) & \text{if } |x| > k, \end{cases} \tag{4.4.11}$$

for a fixed parameter k. This added robustness to outliers is especially helpful when optimizing over functions with several local minima (as is common in

bundle adjustment), because the convexity of the Huber cost function draws the optimizer towards the true minimum [24]. See [54] for details on this and other robust cost functions.

4.4.3 LiDAR Odometry Between-Factors

We include the interpolated odometry measurements from LOAM as between-factors ϕ^T in our graph. LOAM runs asynchronously and at a lower rate than the camera, but with interpolation we can still add LOAM constraints between arbitrary pose variables. We do not however want to interpolate these into higher frequency measurements than the natural frequency of 5Hz. This is because the interpolation assumes a constant-velocity model between the real LOAM updates, which is not in general true to the actual robot movement. Instead, we try to stay close to the real 5Hz frequency of LOAM, but interpolate the poses to the closest camera frames. In practice, we consider two camera frames $k = i$ and $k = j$, where i is the previous frame connected to a between-factor, and $k = j$ is the most recent frame we have a LOAM update for. We then obtain the odometry measurement $\mathbf{z}_{i,j}^T$ as described in Section 4.3.3. Finally, the residual for $\phi^T(T_i, T_j)$ is computed as

$$\mathbf{e}_{i,j}^T = \text{Log}(\mathbf{z}_{i,j}^T T_i^{-1} T_j). \tag{4.4.12}$$

The Log map here transforms to the $SE(3)$ tangent space, so we can optimize over vectors in $\mathbb{R}^6$. As with the projection and range factors, we also apply the Huber robust cost function to the between-factors, meaning that the true contribution to the minization problem is $\rho_h(\|\mathbf{e}_{i,j}^T\|_{\Sigma^T}^2)$ as defined in (4.4.11).

4.5 iSAM2-Based Fixed-Lag Smoother Backend

Our factor graph is formulated in the GTSAM library (Georgia Tech Smoothing and Mapping) [11] and we use its incremental fixed-lag smoother implementation based on iSAM2 [28] and the Bayes tree. This formulation, presented in [7, 27], applies the marginalization scheme we described in Section 3.4.2, where variables are marginalized by removing them from the leaf cliques in the Bayes tree. Marginalization occurs when variables fall out a sliding window $(t^{max} - t^l, t^{max})$ with a fixed temporal lag t^l. To keep track of the variables to be marginalized during an update, we therefore need to keep a set of timestamps corresponding to all variables in $\mathcal{X}$. These timestamps are then used to make ordering constraints for CCOLAMD, where all variables $\mathbf{x}_k$ with $t_k < t^{max} - t^l$, are placed in the beginning of the ordering, such that they end up in leaf cliques in the Bayes tree. After reordering, these variables are marginalized out by removing them from the leaf cliques. We also remove them from $\mathcal{X}$ and their factors from the nonlinear factor graph Φ. All remaining variables that were previously connected with a factor to a

now-marginalized variable, will get a Gaussian marginal factor instead. This three step procedure of ordering, updating and marginalizing completes one iteration of the incremental fixed-lag smoother.

Landmarks require special attention with this scheme however, because the lifetime of a track may exceed the lag of the smoother, and we do not wish to remove it while it is still tracked. We therefore update a landmark's timestamp to the most recent time with every new observation, until the track is finally lost. The landmark is thus only marginalized when its last connected pose is marginalized. We show in Figure 4.7 what happens to the factor graph during intermediary marginalizations as a result of this. The figure shows a slightly modified version of the toy SLAM problem we considered in Chapter 3. Figure 4.7a shows the original factor graph, with variables $\mathcal{X} = \{T_1, T_2, T_3, \mathbf{l}_1, \mathbf{l}_2\}$, and projection-factors, between-factors and a prior factor on T_1. This is of course a simplification of the actual structure of our factor graph, but the principle applies also with additional factors and variables. We consider a fixed lag of 3 poses, meaning that when a new pose T_4 is added to the graph in Figure 4.7c, T_1 is marginalized out. This results in a marginal factor $\phi(T_2, \mathbf{l}_1, \mathbf{l}_2)$ being added on all variables that T_1 is conditioned on in the Bayes net. We show in Figure 4.7b how this arises when eliminating T_1 as the first variable in the elimination algorithm. Here, we use the same notation as in [28] to denote a factor graph that is partially eliminated into a Bayes net. The elimination removes the factors involving T_1 from the graph, and summarizes them into a single joint factor $\phi_{joint}(T_1, T_2, \mathbf{l}_1, \mathbf{l}_2)$. This is then factorized into two parts: a conditional density $p(T_1|T_2, \mathbf{l}_1, \mathbf{l}_2)$ shown with dashed red arrows in Figure 4.7b and a marginal factor $\phi(T_2, \mathbf{l}_1, \mathbf{l}_2)$ on the separator $S = \{T_2, \mathbf{l}_1, \mathbf{l}_2\}$ shown as a red factor, i.e.

$$\phi_{joint}(T_1, T_2, \mathbf{l}_1, \mathbf{l}_2) = p(T_1|T_2, \mathbf{l}_1, \mathbf{l}_2)\phi(T_2, \mathbf{l}_1, \mathbf{l}_2). \tag{4.5.1}$$

Notice that since T_1 only shows up in the single conditional $p(T_1|T_2, \mathbf{l}_1, \mathbf{l}_2)$, it can be marginalized at no computational cost by simply dropping the

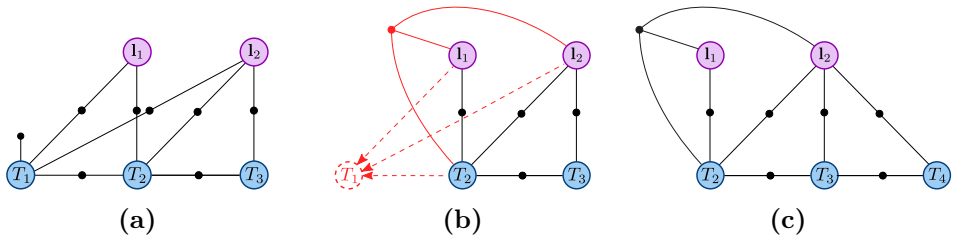


Figure 4.7: One step of the incremental fixed lag smoother with landmarks involved. The variable T_1 in **(a)** is marginalized out as a result of adding T_4 in **(c)**. This results in the marginal factor $\phi(T_2, \mathbf{l}_1, \mathbf{l}_2)$ being added. **(b)** shows the step in the elimination algorithm for eliminating the variable T_1, which results in the factor on the separator $S = \{T_2, \mathbf{l}_1, \mathbf{l}_2\}$, that later becomes the marginal factor.

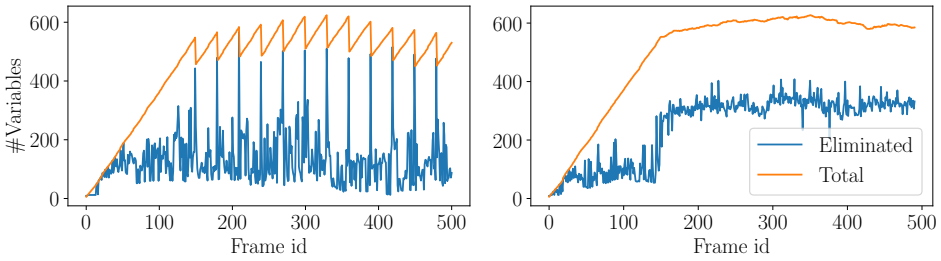


Figure 4.8: Number of re-eliminated variables resulting from marginalizing several variables simultaneously at key frames **(left)**, and from marginalizing the oldest variables at every frame **(right)**. This is using a lag of 10s and marginalization at every 30 frames (2s).

conditional. This is exactly what happens in Figure 4.7c. What remains then is only the marginal factor $\phi(T_2, \mathbf{l}_1, \mathbf{l}_2)$ on the separator S. This is then added into the nonlinear factor graph Φ to retain the information from T_1 for subsequent updates, and the marginalization of T_1 is complete.

While the marginalization of T_1 can be performed without cost, reordering the variables to place T_1 at the leaf nodes can be costly, because it may require re-eliminating a large number of variables. In the example in Figure 4.7, both landmarks $\mathbf{l}_1$ and $\mathbf{l}_2$ had to be re-eliminated to facilitate reordering. The same applies to all landmarks connected to a variable about to be marginalized. We limit the number of re-eliminations on every timestep by grouping together variables within a certain time frame, so that they are all marginalized together at the same time. We do this by only incrementing t^{max} on regular intervals, rather than at every frame (incrementing every 30 frames has shown good results). This leads to less total variable eliminations, except in key frames, where many variables are re-eliminated simultaneously, as can be seen in Figure 4.8.

5 Experimental Evaluation

We perform three experiments on two datasets to test the efficacy of the proposed method. First, in Section 5.2, we show the performance of the complete system on the Newer College dataset [39]. In Section 5.3 we perform a simulated degeneracy experiment on Newer College, where LOAM is made to fail artificially for parts of the dataset. Finally, in Section 5.4 we test the method in a degraded environment where LiDAR odometry fails due to self-similar geometry.

5.1 Evaluation Metrics

To quantify the accuracy of our trajectory estimates compared to ground truth, we report the *absolute pose error* (APE) and *relative pose error* (RPE) as defined in [45]. APE measures the global deviation of our estimate from the ground truth, and is computed as

$$\mathbf{e}_k^{APE} = (T_k^{gt})^{-1} T_k, \tag{5.1.1}$$

where T_k^{gt} is the ground truth pose at timestep k and T_k is the estimate. However for measuring drift of an odometry system, the APE can be less than ideal, because errors early on affect the rest of the trajectory, meaning that errors in the beginning contribute more to the APE than errors at the end [45]. We therefore also report the RPE, which measures the local error over a fixed time interval, computed as

$$\mathbf{e}_k^{RPE} = \left((T_i^{gt})^{-1} T_j^{gt}\right)^{-1} (T_i)^{-1} T_j, \tag{5.1.2}$$

for frames $k = i$ and $k = j$ which form an interval (i, j) of fixed length Δ_{RPE} [45]. For all our experiments we use a delta of 2 seconds, which is equivalent to $\Delta_{RPE} = 30$ when running at 15Hz. We report the magnitudes of the rotational and translational components of these errors, where the magnitude for the rotational component is defined as the unsigned angle of the angle-axis representation of the rotation. Finally, we report the *root mean square* of both these quantities, computed as

$$\begin{aligned} \mathbf{e}_{RMSE,trans} &= \sqrt{\frac{\sum_{k \in \mathsf{K}} \|trans(\mathbf{e}_k)\|^2}{|\mathsf{K}|}}, \\ \mathbf{e}_{RMSE,rot} &= \sqrt{\frac{\sum_{k \in \mathsf{K}} \|\angle rot(\mathbf{e}_k)\|^2}{|\mathsf{K}|}}, \end{aligned} \tag{5.1.3}$$

where the $\angle(\cdot)$ operator denotes taking the angle of the axis-angle representation. All computations of APE and RPE as well as synchronization of estimates with ground truth are performed using the *evo* library [23].

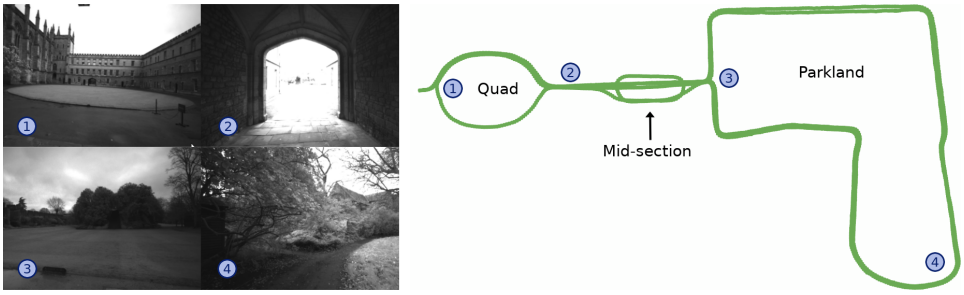


Figure 5.1: Overview of the Newer College dataset [39] with example images and their corresponding locations on the map.

5.2 Evaluation on the Newer College Dataset

5.2.1 Platform and Environment

The Newer College dataset [39] is a dataset of visual, inertial and LiDAR sequences of handheld motion together with an accurate ground truth. It is collected using an Ouster OS1-64 high-resolution 64-beam LiDAR and a RealSense D435i commodity-grade stereo camera with an embedded BOSCH BMI055 IMU. The LiDAR and camera produce measurements at 10Hz and 30Hz respectively, and the IMU at 650Hz[10]. Figure 5.2 shows an image of the platform.

The dataset traverses the campus grounds and parkland of the Newer College campus in Oxford, England and contains parts with dense foliage and periods of rapid illumination change. Figure 5.1 shows an overview of the traversed terrain with example images.

[10]The accelerometer in the RealSense runs at 250Hz and the Gyroscope at 400Hz. In the dataset, these are combined into a total of 650Hz [39].

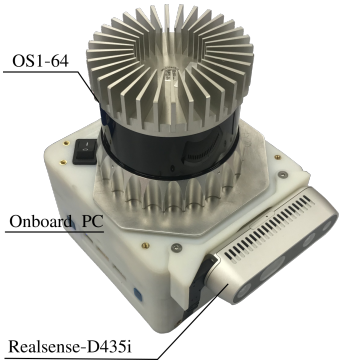


Figure 5.2: Data collection platform for the Newer College dataset [39]. The image is taken from the Newer College dataset home page https://ori-drs.github.io/newer-college-dataset/ and shared under a Creative Commons license (CC BY-NC-SA 4.0).

5.2.2 Depth-Enhanced Visual-Inertial Odometry Results

We first show our system running in depth-enhanced visual-inertial odometry mode. This includes both projection and depth factors into the factor graph, but excludes any between-factors from LOAM. For this, we run the frontend at 30Hz to aid feature tracking, while the backend runs at 15Hz. We use a smoother lag of 5s and keep 5 grid cells with 5-6 features in each. We run this on the first 900 seconds of the dataset, which includes one full traversal of the environment and report the results in Figures 5.3 and 5.4 and table 1. The

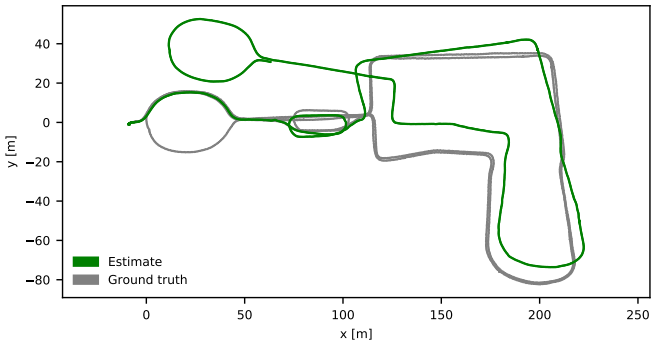
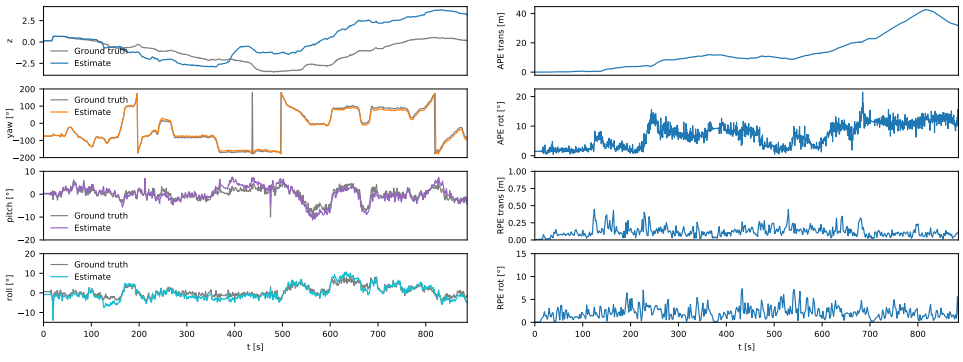


Figure 5.3: Resulting trajectory from running in depth-enhanced VIO-only mode compared to ground truth. The trajectory drifts significantly, especially when going through patches that are difficult for the frontend, such as when moving from the mid-section to the parkland as indicated in Figure 5.1.

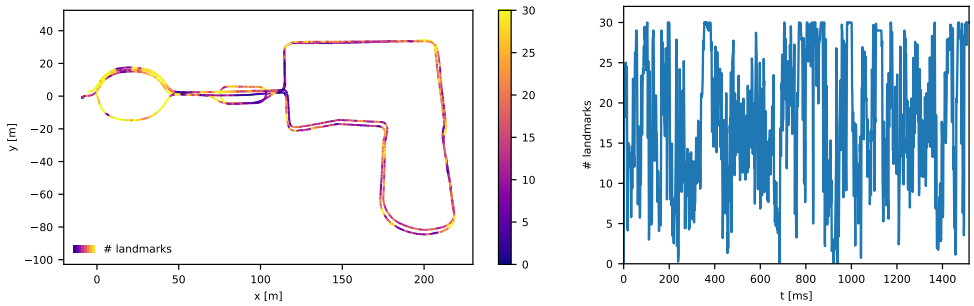


(a) Resulting z, yaw, pitch and roll. **(b)** Resulting APE and RPE.

Figure 5.4: Resulting statistics from running in depth-enhanced VIO-only mode on the Newer College dataset [39].

Table 1: APE and RPE RMSE from depth-enhanced VIO-only mode on the Newer College dataset [39].

$\mathbf{e}^{APE}_{RMSE,trans}$	$\mathbf{e}^{APE}_{RMSE,rot}$	$\mathbf{e}^{RPE}_{RMSE,trans}$	$\mathbf{e}^{RPE}_{RMSE,rot}$
18.18m	8.03°	0.14m	2.52°



(a) Landmarks tracked indicated on top of the Newer College trajectory.

(b) Landmarks tracked over time through the dataset shown as a 1s moving average.

Figure 5.5: Number of landmarks tracked in the backend through the Newer College sequence. For certain parts of the dataset, the landmark counts nearly drop to zero. In such situations, a VIO-only solution is likely to either drift, or lead the optimizer into a local minima, which could cause divergence and system failure. During these parts of the dataset, the addition of LOAM between-factors adds considerable robustness to failure, and if LOAM is not available, we can expect performance to suffer.

resulting trajectory is shown in Figure 5.3 compared to the ground truth and Figure 5.4a show how z, yaw, pitch and roll develops through the sequence. Figure 5.4b shows the APE and RPE through the dataset and table 1 the RMSE of these metrics.

The results indicate that our system struggles significantly without the aiding of between-factors. Parts of the sequence where few landmarks are tracked are particularily challenging, as this leaves very little information to constrain the localization problem. We indicate such problem-spots in Figure 5.5, which shows the number of landmarks tracked through one complete run of the dataset.

5.2.3 Full Lidar-Visual-Inertial Fusion Results

We then show the results of our system running in complete configuration, including between-factors from LOAM. For this, we weight the modalities with a 1mm and 0.005° standard deviation respectively for the translational and rotational part of the LOAM between-factors, 15px for the projection factors and 2m for the range factors. The reason for trusting LOAM the most in this case, is because the structured and large scale environment of the Newer College dataset constrains the LiDAR odometry well. Apart from that, the configuration is the same as for the VIO-only experiment. The results of this are shown in figures Figures 5.6 and 5.7 and table 2 with Figure 5.6 showing the resulting trajectory, Figure 5.7a showing z, yaw, pitch and roll and Figure 5.7b and table 2 showing APE and RPE. As the results

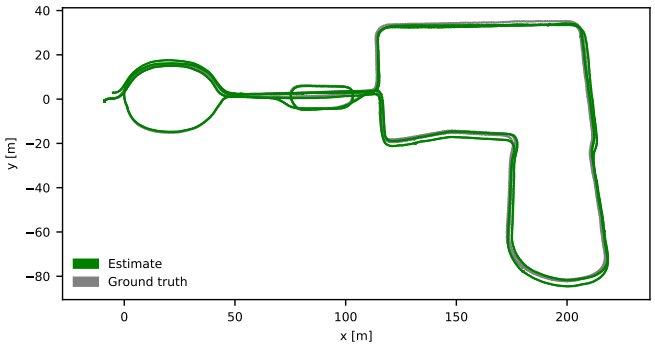
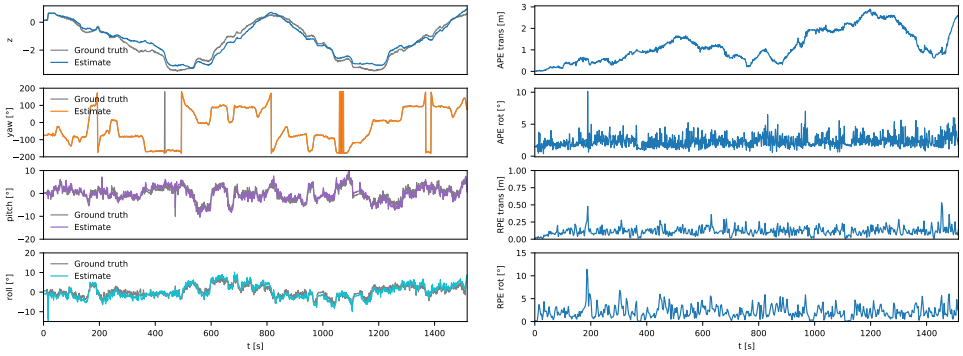


Figure 5.6: Resulting trajectory with full fusion compared to ground truth. With the aiding from LOAM between factors, the trajectory is much closer to the ground truth.



(a) Resulting z, yaw, pitch and roll. **(b)** Resulting APE and RPE.

Figure 5.7: Resulting statistics with full fusion on the Newer College dataset [39].

Table 2: APE and RPE RMSE with full fusion on the Newer College dataset [39].

$\mathbf{e}^{APE}_{RMSE,trans}$	$\mathbf{e}^{APE}_{RMSE,rot}$	$\mathbf{e}^{RPE}_{RMSE,trans}$	$\mathbf{e}^{RPE}_{RMSE,rot}$
1.4m	2.44°	0.13m	2.56°

indicate, the addition of LOAM between-factors results in an estimate that is much closer to the ground truth. This is expected due to the structured environment of the Newer College.

5.2.4 Timing Analysis

To show the real-time capabilities of our system, we report the processing times for a complete run of the 25 minute `01_short_experiment` sequence of the dataset. We run our system on the relatively low-power intel i7-10510U "mobile power efficient" laptop processor. Of the 4 cores and 8 threads available, we dedicate 2 threads to LOAM and the remaining 6 to

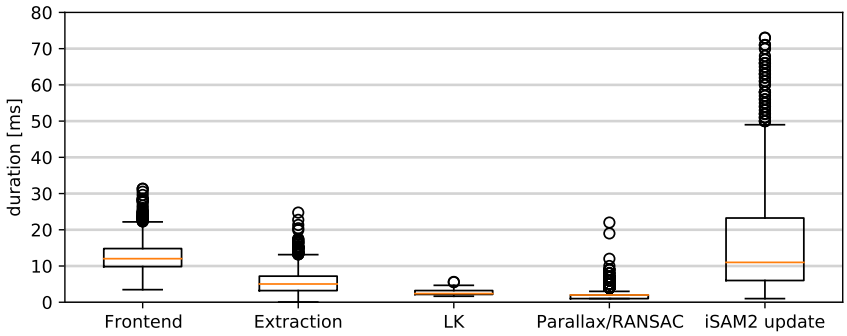
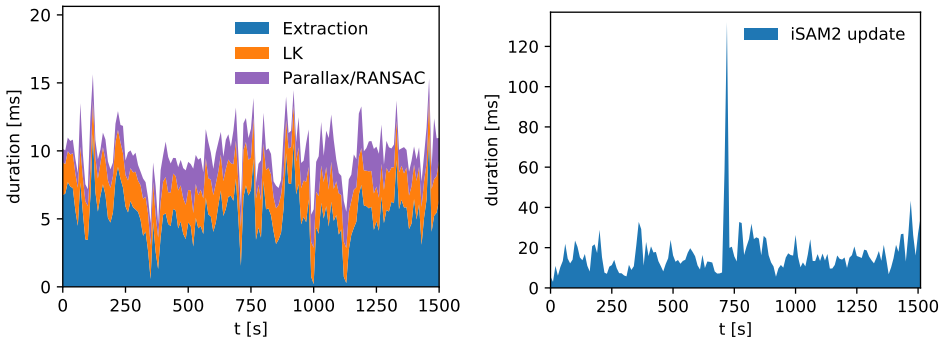


Figure 5.8: Timing breakdown for the visual frontend and backend. Frontend time includes feature extraction, Lucas-Kanade (LK) optical flow calculation and parallax computation/RANSAC outlier rejection. The iSAM2 update duration, which entirely dominates backend time, is for a full iteration of the fixed-lag smoother, including inference, linearization, reordering, elimination and marginalizationl. The boxes extend to the first quartile of the data and whiskers extend to show the range of the data, except outlier points which are shown as black circles. Not shown here are three outliers for the iSAM2 update duration at 225ms, 361ms and 366ms.



(a) Frontend timing. **(b)** Backend timing.

Figure 5.9: Processing times for frontend and backend over the entire dataset. The figures show the data smoothed with a 10s moving average. Notice how the outliers mentioned in Figure 5.8 appear as a spike in the iSAM2 update time.

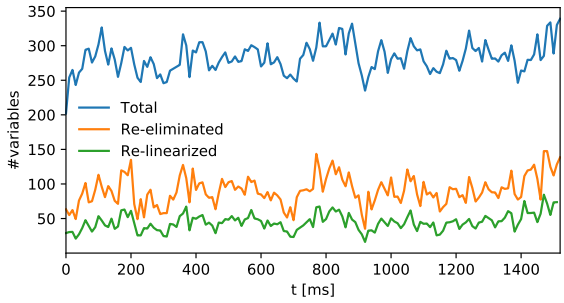


Figure 5.10: Variables in the fixed-lag smoother and re-elimination and re-linearization counts shown as a 10s moving average. Notice that the peaks shown in Figure 4.8 are not shown due to the moving average.

our frontend and backend. We use a smoother lag of 5s and keep 5 grid cells with between 3–5 features per cell. The results of this are shown in Figures 5.8 to 5.10. In Figure 5.8 we show a breakdown of the timing in box-plot form. Figure 5.9 shows the frontend and backend processing times through the dataset. Finally, Figure 5.10 shows a plot of the number of variables in the fixed-lag smoother, showing also how many variables are re-linearized and re-eliminated with every iSAM2 update. This indicates that the problem size is successfully bounded by the fixed-lag smoother and that the iSAM2 Bayes tree allows re-linearizing only a subset of the variables at every iteration, rather than the entire problem.

5.3 Simulated Degeneracy Study

5.3.1 Description of Experiment

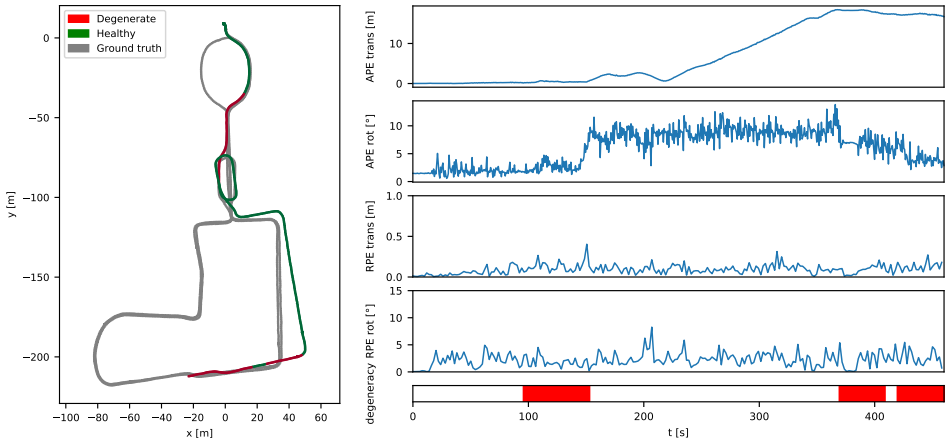
To test how our system handles degeneracy of the LiDAR odometry, we artificially disable the LOAM odometry measurements for certain segments of the Newer College sequence. We do this in a randomized fashion, selecting 4 points in the dataset at random, and then inserting a degeneracy period between 30 and 60 seconds. If any of the periods overlap, they are merged. We limit our runs to the first 900 seconds of the sequence, which amounts to one traversal of the entire environment. We perform 10 such experiments and report the qualitative and quantitative results. All tuning is performed *prior* to generation of the experiment data

Note that while we turn off the LiDAR odometry measurements from LOAM during the simulated degeneracy, we do not turn off LiDAR measurements all together. This is because in many degenerate situations, especially those resulting from self-similar geometry, the LiDAR measurements still contain valuable information that can aid the visual frontend. The LiDAR data alone however is not enough to produce reliable odometry estimates.

5.3.2 Results

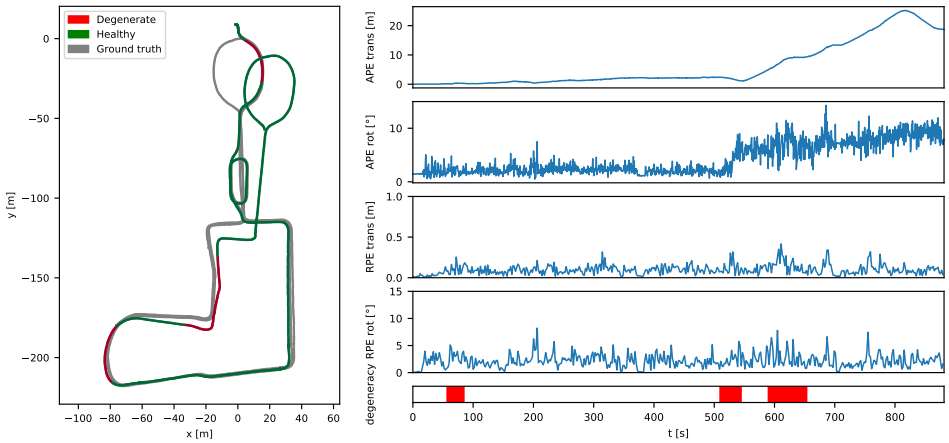
The results of the experiment are shown in Figures 5.11 to 5.20. For each run, we show a top-down view of the resulting trajectory with periods of degeneracy indicated on the trajectory in red. In addition, we plot the rotational and translational APE and RPE as they develop through the run. The period of degeneracy is indicated in red under these plots. Finally, we report the RMSE of the APE and RPE metrics for the whole run and whether the experiment was successful, i.e. it got through the entire sequence, or if the system crashed during the run.

As the results in Figures 5.11 to 5.20 indicate, the system can enable continued operation in times when LOAM becomes degenerate. The success of this however, both in terms of survival and accuracy of the result, depends greatly on the location of the degeneracy period. If LOAM fails at a point



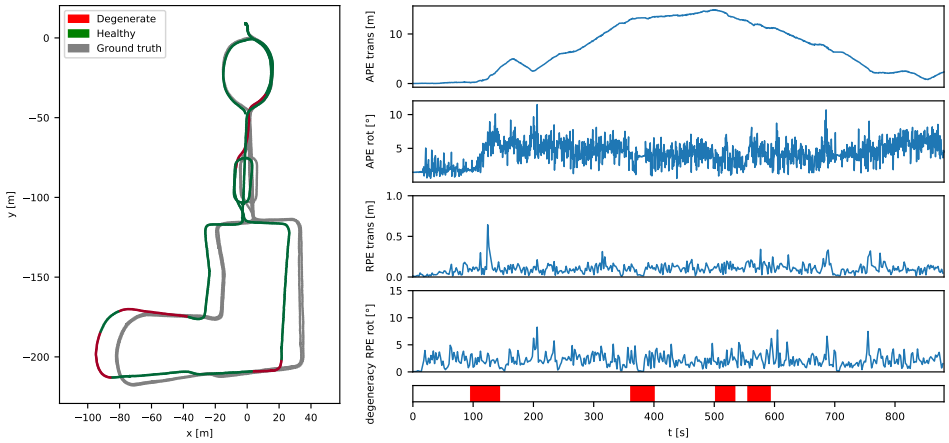
RMSE APE trans [m]	RMSE APE rot [°]	RMSE RPE trans [m]	RMSE RPE rot [°]
10.12m	6.79°	0.11m	2.61°

Figure 5.11: Run 1 of the simulated degeneracy experiment **(failure)**. **Left:** the resulting estimated trajectory compared with ground truth. **Right:** APE and RPE through the sequence. **Bottom:** RMSE statistics for APE and RPE over the entire run. Periods of LOAM degeneracy are indicated in red on the trajectory and on the color-bar underneath the APE and RPE plots. In this run, a rotation estimation error occured during the first degeneracy period, which can be seen as a sharp increase in rotational APE followed by a steadily increasing translational APE. The system finally crashed during the third degeneracy period. On this particular location we are typically tracking few features, which can be seen in Figure 5.5a.



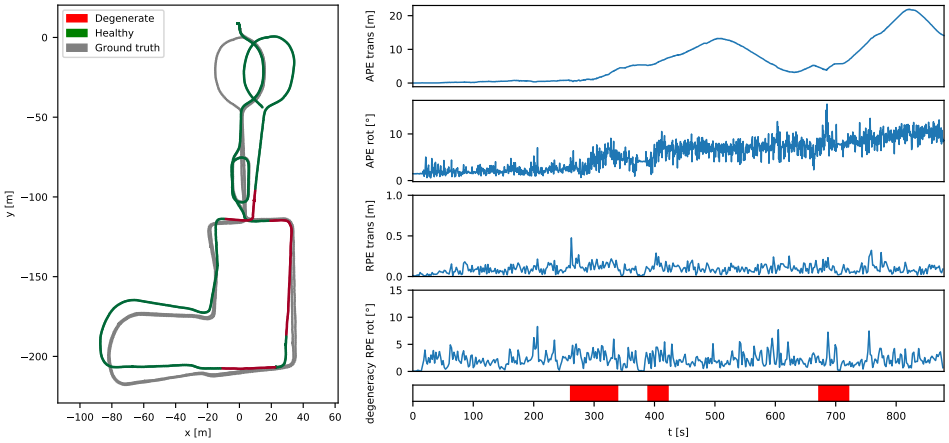
RMSE APE trans [m]	RMSE APE rot [°]	RMSE RPE trans [m]	RMSE RPE rot [°]
9.8m	5.18°	0.12m	2.55°

Figure 5.12: Run 2 of the simulated degeneracy experiment **(success)**. **Left:** the resulting estimated trajectory compared with ground truth. **Right:** APE and RPE through the sequence. **Bottom:** RMSE statistics for APE and RPE over the entire run. Periods of LOAM degeneracy are indicated in red on the trajectory and on the color-bar underneath the APE and RPE plots. In this run, we see a sharp increase in rotational APE during the second degeneracy period, which manifests itself in a growing translational APE through the rest of the run.



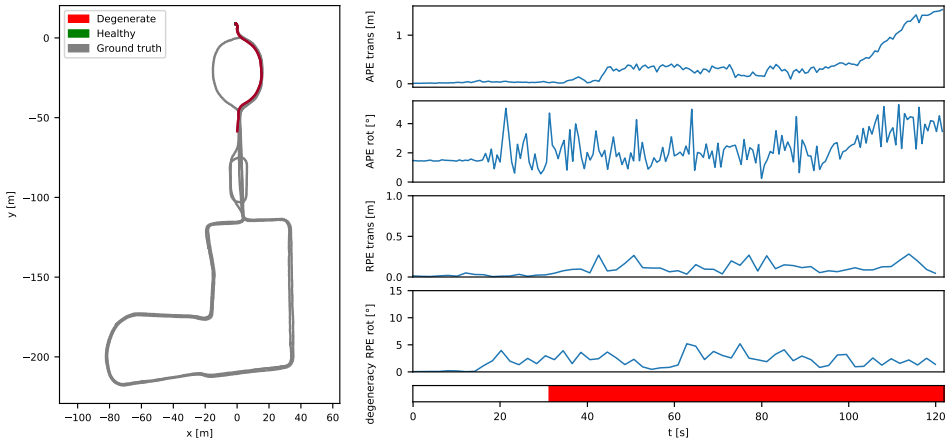
RMSE APE trans [m]	RMSE APE rot [°]	RMSE RPE trans [m]	RMSE RPE rot [°]
8.57m	4.56°	0.12m	2.52°

Figure 5.13: Run 3 of the simulated degeneracy experiment **(success)**. **Left:** the resulting estimated trajectory compared with ground truth. **Right:** APE and RPE through the sequence. **Bottom:** RMSE statistics for APE and RPE over the entire run. Periods of LOAM degeneracy are indicated in red on the trajectory and on the color-bar underneath the APE and RPE plots. In this run, we see a rotation error during the first degeneracy period with a clear spike in RPE and an increase in rotational APE. However, because the VIO-only system handled well the remaining degeneracy periods during traversal of the parkland, we finally return to the starting point with a low final translational APE.



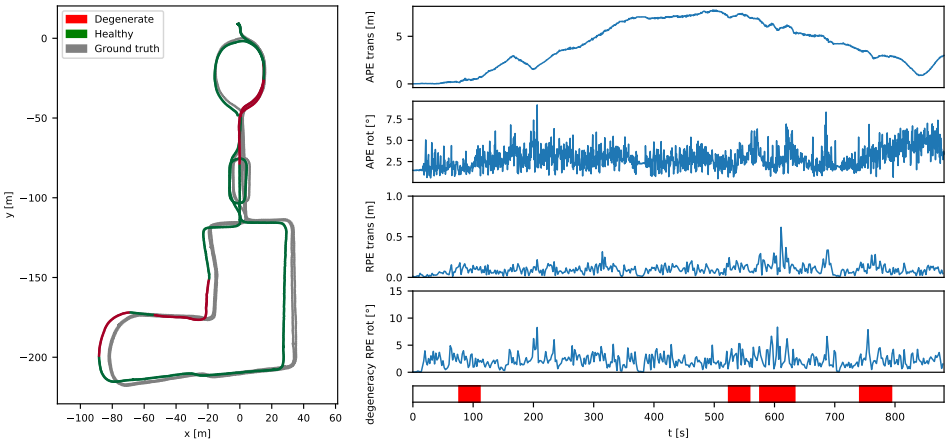
RMSE APE trans [m]	RMSE APE rot [°]	RMSE RPE trans [m]	RMSE RPE rot [°]
8.78m	6.4°	0.11m	2.56°

Figure 5.14: Run 4 of the simulated degeneracy experiment **(success)**. **Left:** the resulting estimated trajectory compared with ground truth. **Right:** APE and RPE through the sequence. **Bottom:** RMSE statistics for APE and RPE over the entire run. Periods of LOAM degeneracy are indicated in red on the trajectory and on the color-bar underneath the APE and RPE plots.



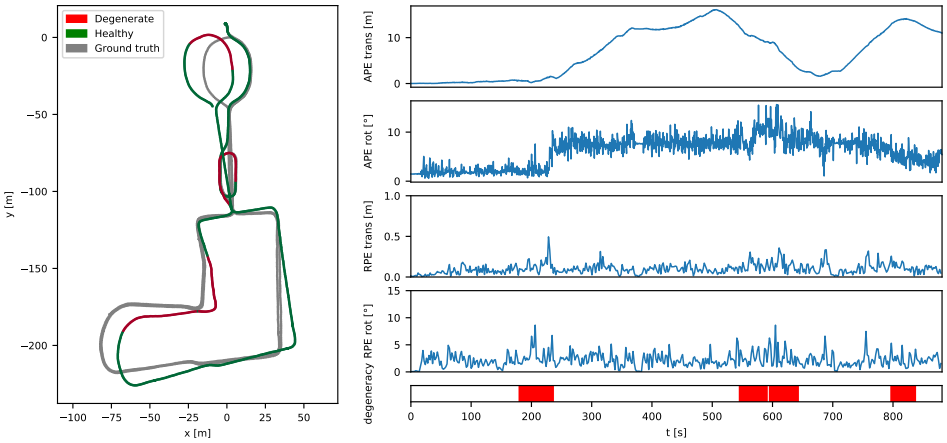
RMSE APE trans [m]	RMSE APE rot [°]	RMSE RPE trans [m]	RMSE RPE rot [°]
0.5m	2.44°	0.12m	2.42°

Figure 5.15: Run 5 of the simulated degeneracy experiment **(failure)**. **Left:** the resulting estimated trajectory compared with ground truth. **Right:** APE and RPE through the sequence. **Bottom:** RMSE statistics for APE and RPE over the entire run. Periods of LOAM degeneracy are indicated in red on the trajectory and on the color-bar underneath the APE and RPE plots. In this run, the the VIO-only solution was not able to survive the first 90s degenearcy period and crashed shortly after exiting the quad.



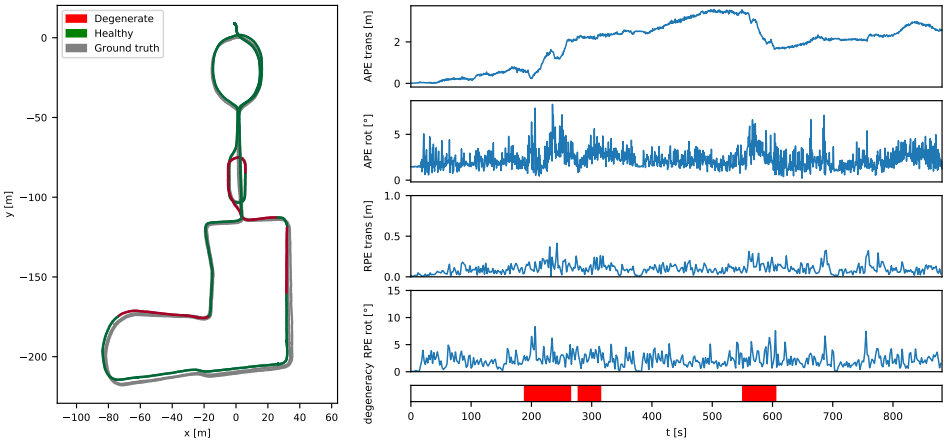
RMSE APE trans [m]	RMSE APE rot [°]	RMSE RPE trans [m]	RMSE RPE rot [°]
4.77m	3.14°	0.12m	2.57°

Figure 5.16: Run 6 of the simulated degeneracy experiment **(success)**. **Left:** the resulting estimated trajectory compared with ground truth. **Right:** APE and RPE through the sequence. **Bottom:** RMSE statistics for APE and RPE over the entire run. Periods of LOAM degeneracy are indicated in red on the trajectory and on the color-bar underneath the APE and RPE plots.



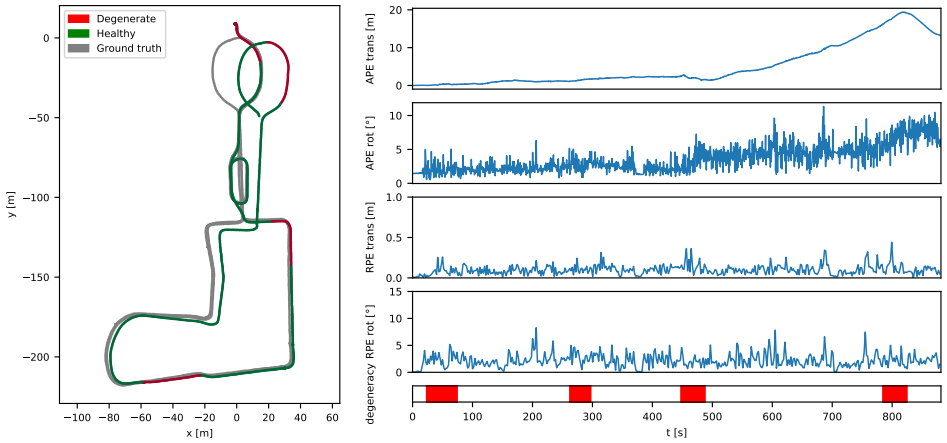
RMSE APE trans [m]	RMSE APE rot [°]	RMSE RPE trans [m]	RMSE RPE rot [°]
8.68m	6.85°	0.12m	2.57°

Figure 5.17: Run 7 of the simulated degeneracy experiment **(success)**. **Left:** the resulting estimated trajectory compared with ground truth. **Right:** APE and RPE through the sequence. **Bottom:** RMSE statistics for APE and RPE over the entire run. Periods of LOAM degeneracy are indicated in red on the trajectory and on the color-bar underneath the APE and RPE plots. In this run, rotational errors during the first, second and third degeneracy periods result in an increase in rotational APE, which manifests itself in a significant translation APE through the rest of the run.



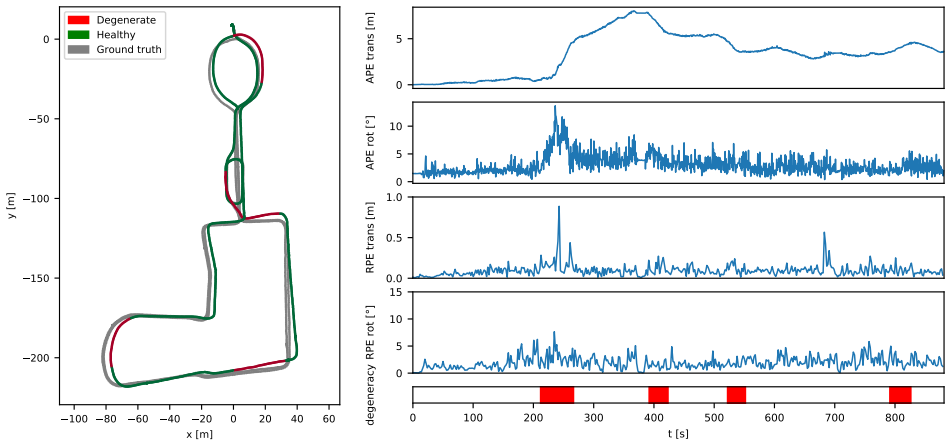
RMSE APE trans [m]	RMSE APE rot [°]	RMSE RPE trans [m]	RMSE RPE rot [°]
2.2m	2.5°	0.12m	2.54°

Figure 5.18: Run 8 of the simulated degeneracy experiment **(success)**. **Left:** the resulting estimated trajectory compared with ground truth. **Right:** APE and RPE through the sequence. **Bottom:** RMSE statistics for APE and RPE over the entire run. Periods of LOAM degeneracy are indicated in red on the trajectory and on the color-bar underneath the APE and RPE plots. In this run, the VIO-only solution was able to sustain good tracking through all degeneracy periods, resulting in a low overall RMSE APE of only 2.2m and 2.5° translation and rotational.



RMSE APE trans [m]	RMSE APE rot [°]	RMSE RPE trans [m]	RMSE RPE rot [°]
7.61m	4.12°	0.12m	2.53°

Figure 5.19: Run 9 of the simulated degeneracy experiment **(success)**. **Left:** the resulting estimated trajectory compared with ground truth. **Right:** APE and RPE through the sequence. **Bottom:** RMSE statistics for APE and RPE over the entire run. Periods of LOAM degeneracy are indicated in red on the trajectory and on the color-bar underneath the APE and RPE plots. In this run, we see an increase of rotational APE during the third degeneracy period, which manifests itself in a growing translational APE through the rest of the run.



RMSE APE trans [m]	RMSE APE rot [°]	RMSE RPE trans [m]	RMSE RPE rot [°]
4.12m	3.49°	0.12m	2.26°

Figure 5.20: Run 10 of the simulated degeneracy experiment **(success)**. **Left:** the resulting estimated trajectory compared with ground truth. **Right:** APE and RPE through the sequence. **Bottom:** RMSE statistics for APE and RPE over the entire run. Periods of LOAM degeneracy are indicated in red on the trajectory and on the color-bar underneath the APE and RPE plots.

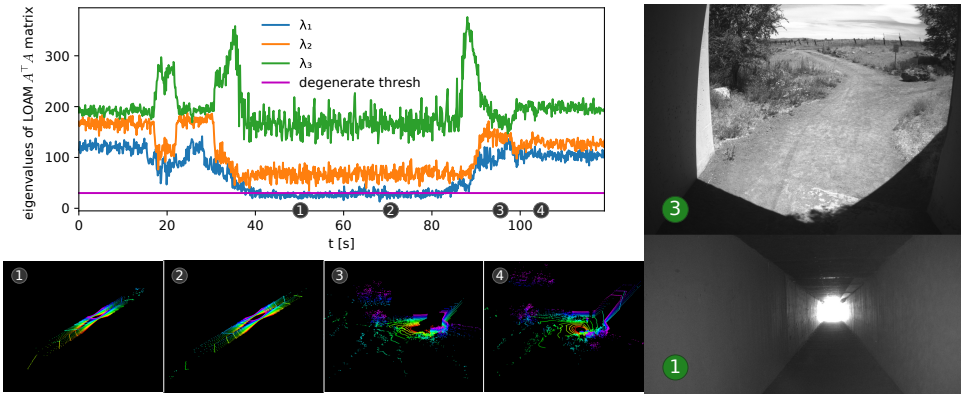


Figure 5.21: The three first eigenvalues of the LOAM $A^\top A$ matrix through the San Rafael tunnel sequence, along with example point clouds and images. The figure indicates the threshold where LOAM is considered degenerate according to the heuristic described in Section 4.3. The numbered examples indicate the environments that trigger the degeneracy (1 and 2) compared to the environments that do not (3 and 4). The example image (1) additionally shows the difficult low-texture nature of the tunnel, which makes it hard for our LK-based frontend to produce good feature tracks. The figure is is partially reused from our previous work [47].

where the visual frontend is also struggling, as indicated in the landmark counts in Figure 5.5, there is naturally little information to go on. This impacts both the accuracy, as we can tell by the spikes in RPE during some of the degeneracy periods, and the chance of survival as we can tell by the two failures during degeneracy.

5.4 Evaluation in a Geometrically Self-Similar Environment

5.4.1 Platform and Environment

As a final experiment, we test our system in a self-similar environment where the LiDAR odometry becomes degenerate. This dataset, which we refer to as the San Rafael tunnel sequence, is recorded in a highway underpass tunnel in the Rancho San Rafael Regional Park in Reno, Nevada. For most of the underpass, the LiDAR point clouds are entirely self-similar, which makes the point cloud alignment problem ill-conditioned and the robot translation unobservable along the direction of the tunnel. Figure 5.21 shows examples of these self-similar point clouds and indicates how LOAM successfully detects the degenerate problem instance from the eigenvalues of the $A^\top A$ matrix when traversing the tunnel. The dataset is recorded using a Velodyne Puck-LITE 10Hz LiDAR, a FLIR BlackFly 20Hz camera and a VectorNav VN-100 IMU running at 200Hz. We test our system running in full configuration and report the resulting LiDAR map.

5.4.2 Results

Our results of this experiment are shown in Figure 5.22. LOAM in this case produces a map that is much shorter than the actual tunnel, whereas our fused result is more closer in length to the ground truth. Our result however is not geometrically consistent. We attribute this mainly due to drift in the middle of the tunnel, where the darkness and low-texture tunnel walls make landmarks hard to track. Indeed, as we show in Figure 5.23, low landmark counts are prevelant through parts of the tunnel, and when this is paired

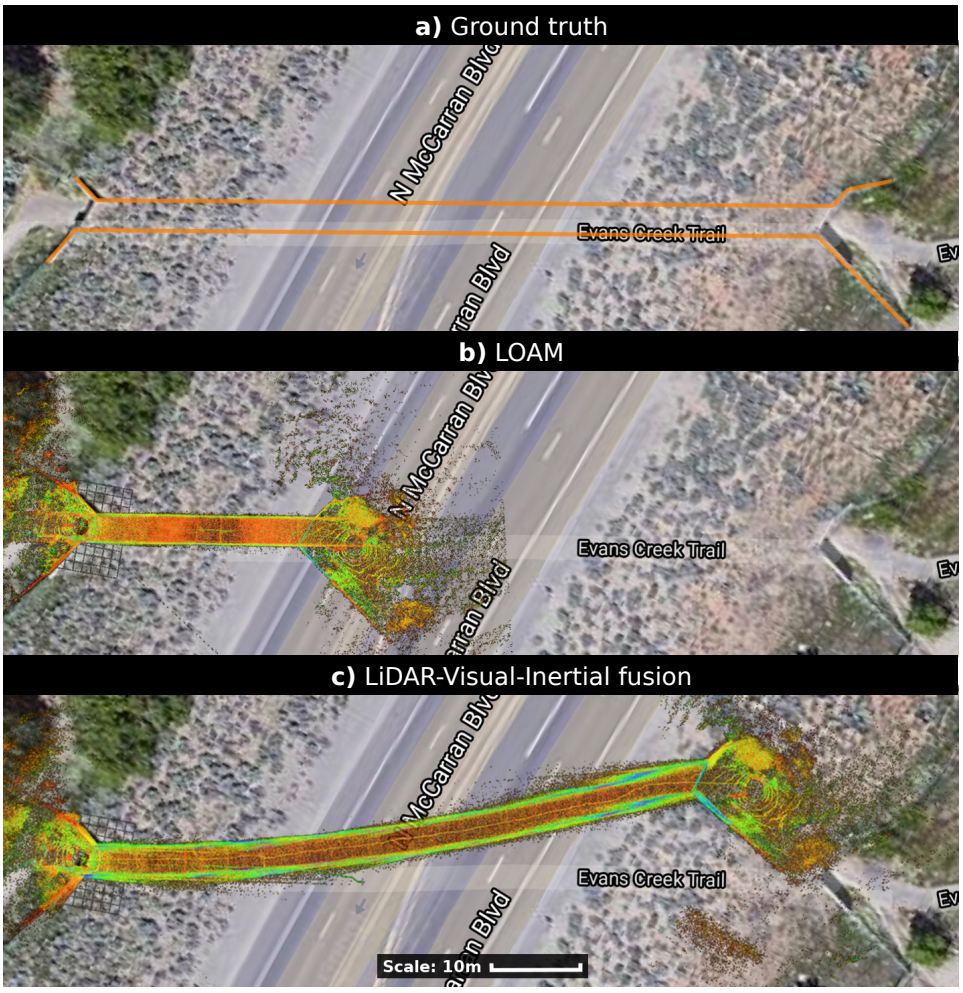


Figure 5.22: Qualitative results of running our system on the San Rafael tunnel dataset overlayed on Google Maps. The map of the tunnel produced by LOAM is much shorter than the ground truth, due to the geometric self-similarity of the tunnel. Our LiDAR-visual-inertial fusion gets comparatively closer to the real length, but also struggles in the middle of the sequence due to low landmark counts, and hence produces a geometrically inconsistent result.

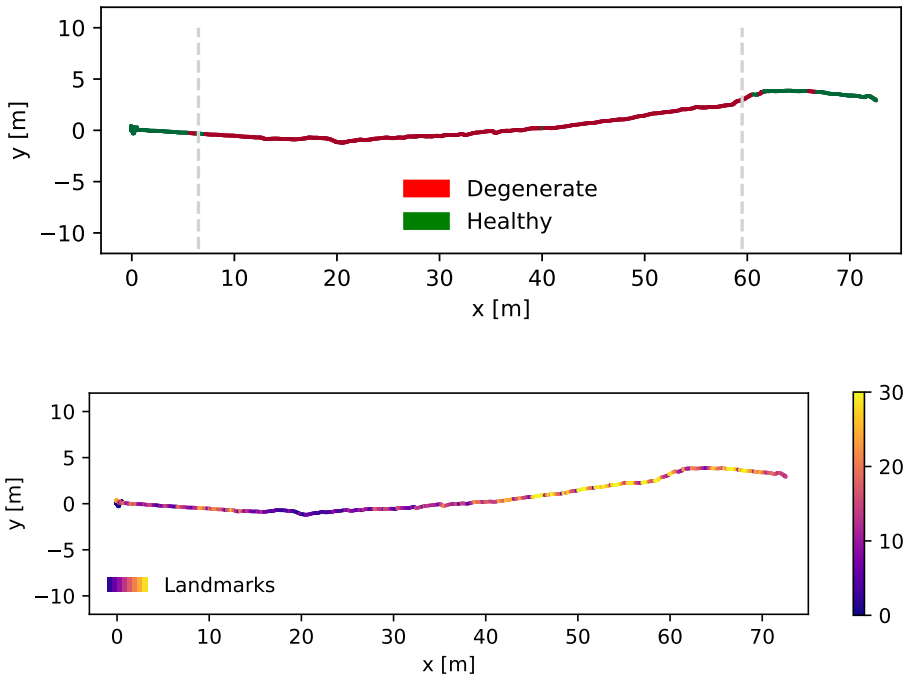


Figure 5.23: LOAM Degeneracy **(top)** and visual landmark counts **(bottom)** through the San Rafael tunnel sequence. The top figure shows that LOAM is degenerate throughout most of the tunnel. In addition, as shown in the bottom figure, some parts of the tunnel have low landmark counts which makes the VIO-only solution struggle as well.

with LOAM degeneracy, it is natural to expect drift, as neither between factors nor landmarks truly constrain the pose estimates. Nevertheless, it should be noted that LiDAR-visual-inertial fusion can indeed work on this dataset, as shown in [29] and in our preliminary work [47]. Both these methods however use the ROVIO visual-inertial odometry system, which, being a direct method, provides more robust feature tracking in low-texture scenes [1].

6 Discussion and Concluding Remarks

As our results in Chapter 5 indicate, the combination of the complementary LiDAR, visual and inertial sensor modalities can add resilience to situations where a single-modality solution would otherwise fail. As our benchmark and simulated degeneracy study on the Newer College dataset shows, the complete system is resilient to low landmark counts when LOAM between-factors are available. Conversely, when the backend is tracking enough landmarks, the system can also withstand failure of the LOAM subsystem. When both degradations occur simultaneously however, with a degenerate LOAM in addition to low landmark counts, the system struggles, which typically manifests itself in high drift or even divergence and failure. The experiment in the San Rafael underpass confirm these findings as well, as the visual-inertial solution is able to keep the tracking going through the structurally self-similar tunnel, but does so with considerable drift due to the severely low landmark counts in the dark, low-texture parts of the tunnel.

It is important to emphasise however, that with improvements to the visual frontend, we should expect better results, with less drift and less chance of divergence. Indeed, as mentioned in Section 5.4, the ROVIO VIO system used in [29, 47] is able to track through the low-texture tunnel without drift. ROVIO is however a special case, because its feature tracking scheme is particularly robust to low-texture scenarios, and was therefore specifically chosen as the visual subsystem in [29]. This nonetheless proves that a visual-inertial method can achieve a good result on this dataset, without aiding from LiDAR. The same goes for the Newer College sequence, where monocular vision-only methods such as ORB-SLAM [36] have been proven to work well, although with the help of loop closures. We consider two reasons for our issues on this front. The first is the aforementioned use of optical-flow-based feature tracking instead of a more direct method, which is generally considered superior in terms of robustness to low-texture scenes [1, 17]. The second reason is the non-trivial problem of feature selection. The system must include enough features in the bundle adjustment to constrain the poses, but at the same time avoid including outliers and low-parallax features that can not be triangulated well. A successful implementation must hence find the right balance between these two extremes and neither be too stringent nor too lenient in the feature selection process. We believe finding this balance requires a certain finesse and considerable efforts on tuning, trial and error, but it is by no means out of reach.

The San Rafael tunnel sequence exemplifies an additional opportunity for further work on this system, namely investigating the gains in robustness from adding infinite line and plane factors to the factor graph as is done in [49]. Because the tunnel sequence is a well defined structural environment, alignment of infinite lines and planes extracted from consecutive point clouds should constrain the rotational component of the robot pose with

great accuracy. We hypothesise that this could aid the overall optimization and lead to a more accurate and geometrically consistent result than a VIO-solution alone. Testing this tighter LiDAR-visual coupling on this dataset could therefore be of great research interest. On a more general note, it would be of great interest to investigate how to exploit the measurements of any sensor modality that has one or more unobservable degrees of freedom. The line and plane factors presented in [49] is just one way to do this, and our current method is in fact already incorporating information from the degenerate LiDAR sensor stream in the form of depth enriched visual landmarks. An alternative approach could be to go the route of [51] and detect and separate the degenerate degrees of freedom from the well-conditioned degrees of freedom. These approaches, can in theory be expected produce better estimates than a boolean healthy/degenerate state due to not turning the partially failing modality completely off in the case of degeneracy.

All in all, the method and the theory presented in this work can be seen as a step in the direction of a tighter LiDAR-visual-inertial coupling and a tighter multi-modality exteroceptive fusion paradigm in general. The system we have presented to facilitate this is a semi-tight fusion framework for LiDAR-visual-inertial odometry based on sliding-window factor graphs and the Bayes tree. While the evaluation of our proposed system shows that a loosely coupled LiDAR-visual-inertial odometry can offer improved resilience to both LiDAR and visual degeneracy, our results also indicate that further investigation into a more unified tight-fusion factor-graph formulation could be of great research interest. In any case, we have shown that the use of complementary exteroceptive sensor modalities is key to achieve resilience in challenging environments and yet is an open and active research field with many exciting prospects. Indeed, looking out at the broader field of robotic perception and autonomous systems, we see a flurry of recent and ongoing work on multi-modal perception that is pushing the boundaries of what is possible for resilient autonomous systems [29, 42, 49, 55]. We therefore expect to see many great complementary sensor-fusion systems—tightly coupled or not—in the years to come.

7 Abbreviations

Table 3: Abbreviations

Abbreviation	Meaning
APE	Absolute Pose Error
COLAMD	Column Approximate Minimum Degree
EKF	Extended Kalman Filter
GN	Gauss-Newton
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GTSAM	Georgia Tech Smoothing and Mapping
IMU	Inertial Measurement Unit
iSAM	Incremental Smoothing and Mapping
KF	Kalman Filter
LiDAR	Light Detection and Ranging
LM	Levenberg-Marquardt
MAP	Maximum a posteriori
RANSAC	RANdom SAmple Consensus
RHS	Right Hand Side
RMS	Root Mean Square
RMSE	Root Mean Square Error
RPE	Relative Pose Error
SAM	Smoothing and Mapping
SE(n)	Special Euclidean Group n
SLAM	Simultaneous Localization and Mapping
SLERP	Spherical Linear Interpolation
SO(n)	Special Orthogonal Group n
SU(n)	Special Unitary Group n
VIO	Visual-Inertial Odometry
VO	Visual Odometry

8 List of Figures

List of Figures

References

[1] M. Bloesch et al. "Robust visual inertial odometry using a direct EKF-based approach." In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 298–304.

[2] Jean-Yves Bouguet et al. "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm." In: *Intel corporation* 5.1-10 (2001), p. 4.

[3] G. Bradski. "The OpenCV Library." In: *Dr. Dobb's Journal of Software Tools* (2000).

[4] C. Cadena et al. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age." In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[5] Carlos Campos et al. "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM." In: *arXiv preprint arXiv:2007.11898* (2020).

[6] Chang Chen et al. "A Review of Visual-Inertial Simultaneous Localization and Mapping from Filtering-Based and Optimization-Based Perspectives." In: *Robotics* 7.3 (2018). URL: https://www.mdpi.com/2218-6581/7/3/45.

[7] Han-Pang Chiu et al. "Robust vision-aided navigation using Sliding-Window Factor graphs." In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 46–53.

[8] Chee-Yee Chong et al. "Architectures and algorithms for track association and fusion." In: *IEEE Aerospace and Electronic Systems Magazine* 15.1 (2000), pp. 5–13.

[9] Erik B. Dam, Martin Koch, and Martin Lillholm. *Quaternions, Interpolation and Animation*. Tech. rep. University of Copenhagen, 1998.

[10] Tim Davis et al. "A column approximate minimum degree ordering algorithm." In: *ACM Trans. Math. Softw.* 30 (Sept. 2004), pp. 353–376.

[11] Frank Dellaert. *Factor graphs and GTSAM: A hands-on introduction*. Tech. rep. Georgia Institute of Technology, 2012.

[12] Frank Dellaert and Michael Kaess. "Square Root SAM: Simultaneous localization and mapping via square root information smoothing." In: *The International Journal of Robotics Research* 25.12 (2006), pp. 1181–1203.

[13] Frank Dellaert, Michael Kaess, et al. "Factor graphs for robot perception." In: *Foundations and Trends® in Robotics* 6.1-2 (2017), pp. 1–139.

[14] Ethan Eade. “Lie groups for 2d and 3d transformations.” In: ().

[15] Olav Egeland and Jan Tommy Gravdahl. *Modeling and simulation for automatic control*. Marine Cybernetics Trondheim, Norway, 2002.

[16] Jakob Engel, Vladlen Koltun, and Daniel Cremers. “Direct sparse odometry.” In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2017), pp. 611–625.

[17] Jakob Engel, Thomas Schöps, and Daniel Cremers. “LSD-SLAM: Large-Scale Direct Monocular SLAM.” In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 834–849.

[18] Tully Foote. “tf: The transform library.” In: *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*. Open-Source Software workshop. Apr. 2013, pp. 1–6.

[19] C. Forster et al. “On-Manifold Preintegration for Real-Time Visual–Inertial Odometry.” In: *IEEE Transactions on Robotics* 33.1 (2017), pp. 1–21.

[20] Christian Forster et al. “SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems.” In: *IEEE Transactions on Robotics* 33.2 (2017), pp. 249–265.

[21] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite.” In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[22] Johannes Graeter, Alexander Wilczynski, and Martin Lauer. “LIMO: Lidar-Monocular Visual Odometry.” In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 7872–7879.

[23] Michael Grupp. *evo: Python package for the evaluation of odometry and SLAM.* https://github.com/MichaelGrupp/evo. 2017.

[24] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. USA: Cambridge University Press, 2003.

[25] Peter J. Huber. “Robust Estimation of a Location Parameter.” In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101. URL: https://doi.org/10.1214/aoms/1177703732.

[26] M. Kaess, A. Ranganathan, and F. Dellaert. “iSAM: Incremental Smoothing and Mapping.” In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1365–1378.

[27] Michael Kaess et al. “Concurrent filtering and smoothing.” In: *2012 15th International Conference on Information Fusion*. 2012, pp. 1300–1307.

[28] Michael Kaess et al. “iSAM2: Incremental smoothing and mapping using the Bayes tree.” In: *The International Journal of Robotics Research* 31.2 (2012), pp. 216–235. eprint: https://doi.org/10.1177/0278364911430419. URL: https://doi.org/10.1177/0278364911430419.

[29] S. Khattak et al. “Complementary Multi–Modal Sensor Fusion for Resilient Robot Pose Estimation in Subterranean Environments.” In: *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2020, pp. 1024–1029.

[30] Georg Klein and David Murray. “Parallel Tracking and Mapping for Small AR Workspaces.” In: *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*. Nara, Japan, Nov. 2007.

[31] Stefan Leutenegger et al. “Keyframe-based visual–inertial odometry using nonlinear optimization.” In: *The International Journal of Robotics Research* 34.3 (2015), pp. 314–334.

[32] KENNETH LEVENBERG. “A METHOD FOR THE SOLUTION OF CERTAIN NON-LINEAR PROBLEMS IN LEAST SQUARES.” In: *Quarterly of Applied Mathematics* 2.2 (1944), pp. 164–168. URL: http://www.jstor.org/stable/43633451.

[33] Zheng Liu and Fu Zhang. “BALM: Bundle Adjustment for Lidar Mapping.” In: *CoRR* abs/2010.08215 (2020).

[34] Todd Lupton and Salah Sukkarieh. “Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions.” In: *IEEE Transactions on Robotics* 28.1 (2011), pp. 61–76.

[35] E. Mouragnon et al. “Real Time Localization and 3D Reconstruction.” In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 1. 2006, pp. 363–370.

[36] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system.” In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[37] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[38] Tong Qin, Peiliang Li, and Shaojie Shen. “VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator.” In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

[39] Milad Ramezani et al. “The Newer College Dataset: Handheld LiDAR, Inertial and Vision with Ground Truth.” In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 4353–4360.

[40] Antoni Rosinol et al. "Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping." In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2020. URL: https://github.com/MIT-SPARK/Kimera.

[41] Tixiao Shan et al. "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 5135–5142.

[42] Tixiao Shan et al. "LVI-SAM: Tightly-coupled Lidar-Visual-Inertial Odometry via Smoothing and Mapping." In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021.

[43] Jianbo Shi and Tomasi. "Good features to track." In: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 1994, pp. 593–600.

[44] Joan Solà, Jeremie Deray, and Dinesh Atchuthan. *A micro Lie theory for state estimation in robotics*. Tech. rep. IRI-TR-18-01. Barcelona: Institut de Robòtica i Informàtica Industrial, 2018.

[45] Jürgen Sturm et al. "A benchmark for the evaluation of RGB-D SLAM systems." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 573–580.

[46] Sebastian Thrun. "Probabilistic Robotics." In: *Commun. ACM* 45.3 (Mar. 2002), pp. 52–57. URL: https://doi.org/10.1145/504729.504754.

[47] Sigurd Vatn Totland. *Resilient Graph-based Multi-modal SLAM for Sensor-degraded environments*. Tech. rep. Norwegian University of Science and Technology, 2020.

[48] Vladyslav Usenko et al. "Visual-Inertial Mapping With Non-Linear Factor Recovery." In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 422–429.

[49] David Wisth et al. "Unified Multi-Modal Landmark Tracking for Tightly Coupled Lidar-Visual-Inertial Odometry." In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 1004–1011.

[50] Oliver J Woodman. *An introduction to inertial navigation*. Tech. rep. University of Cambridge, Computer Laboratory, 2007.

[51] J. Zhang, M. Kaess, and S. Singh. "On degeneracy of optimization-based state estimation problems." In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 809–816.

[52] J. Zhang and S. Singh. "Visual-lidar odometry and mapping: low-drift, robust, and fast." In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 2174–2181.

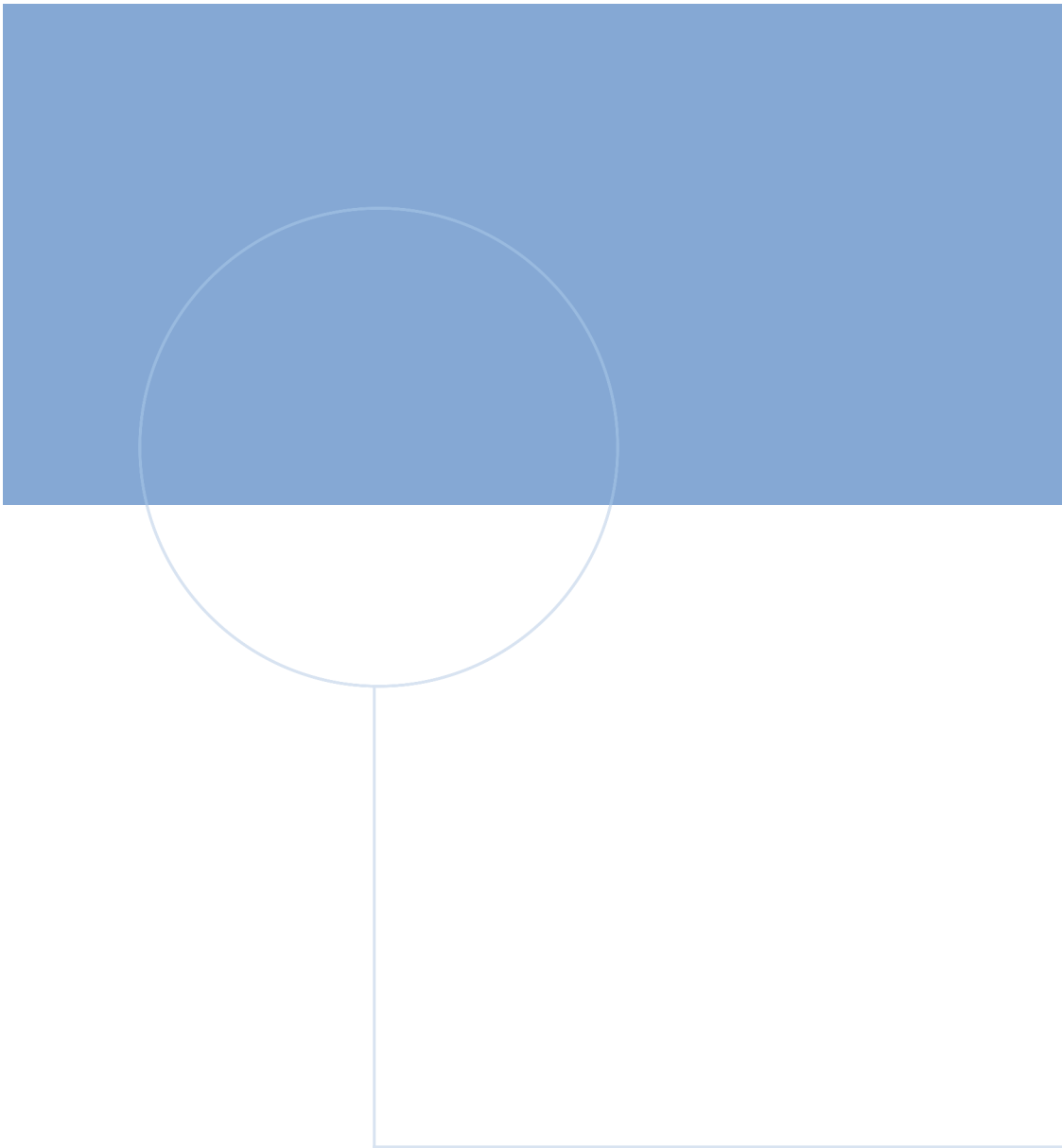
[53] Ji Zhang and Sanjiv Singh. "LOAM: Lidar Odometry and Mapping in Real-time." In: *Robotics: Science and Systems*. Vol. 2. 9.

[54] Zhengyou Zhang. "Parameter estimation techniques: A tutorial with application to conic fitting." In: *Image and vision Computing* 15.1 (1997), pp. 59–76.

[55] Shibo Zhao et al. "Super Odometry: IMU-centric LiDAR-Visual-Inertial Estimator for Challenging Environments." In: *CoRR* abs/2104.14938 (2021). arXiv: 2104.14938. URL: https://arxiv.org/abs/2104.14938.

[56] Xingxing Zuo et al. "LIC-Fusion 2.0: LiDAR-Inertial-Camera Odometry with Sliding-Window Plane-Feature Tracking." In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5112–5119.

[57] Xingxing Zuo et al. "LIC-Fusion: LiDAR-Inertial-Camera Odometry." In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 5848–5854.



NTNU

Kunnskap for en bedre verden