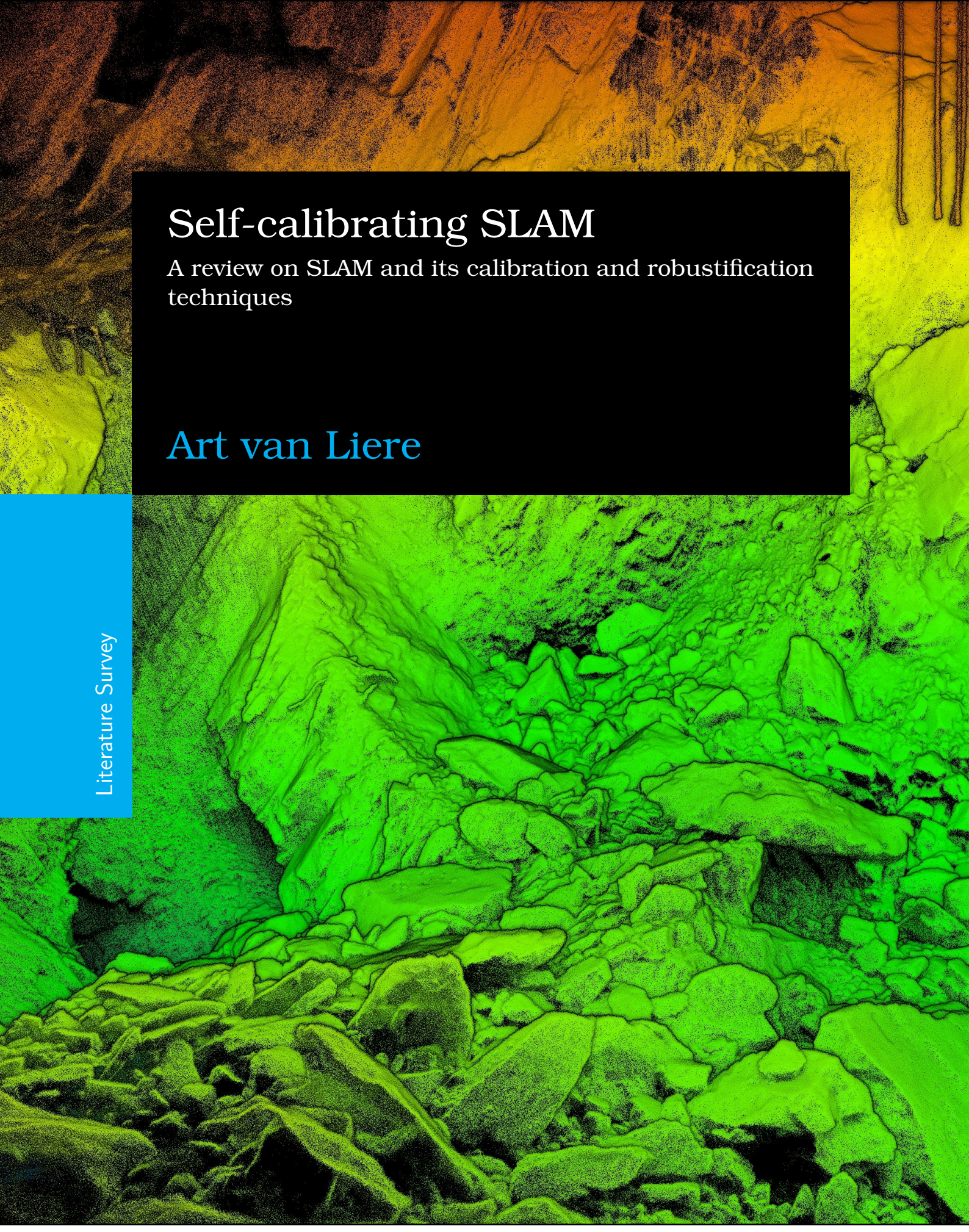

Self-calibrating SLAM

A review on SLAM and its calibration and robustification techniques

Art van Liere

Literature Survey

TU Delft Delft University of Technology

Delft Center for Systems and Control

Self-calibrating SLAM

A review on SLAM and its calibration and robustification techniques

Literature Survey

Art van Liere

6th January 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by Demcon Advanced Mechatronics Delft B.V. Their cooperation is hereby gratefully acknowledged.

Abstract

The goal of this research is to assist Demcon's Robotics Competence Group at gaining expertise in the field of robotic navigation. The problem of navigation is formalised as the Simultaneous Localisation and Mapping (or SLAM) problem, which incrementally builds a consistent map of an unknown environment, while simultaneously finding its own location within that map. Modern solutions to SLAM discretise the robot trajectory as a set of reference locations in space and time. A sensor model derives from each sensor measurement a soft constraint between poses, which defines a probabilistic distribution defined by a constraint value and a constrain covariance. The constraint value indicates the geometric relation that perfectly matches the observation, and the constraint covariance acts as a confidence metric for the accuracy of the constraint. Pose-graph optimisation finds the optimal compromise of poses over all constraints.

Setting up a SLAM approach is done via a lengthy configuration process of (assumed constant) tuning parameters: sensor models need to be calibrated, the constraint needs to be estimated. This requires insight in the system and can be a daunting task for a SLAM novice. By extending the pose-graph optimisation formulation to include both the sensor model calibration parameters and the constraint covariances, the importance of tuning is reduced, which simplifies the initialisation process. Furthermore, live-estimation could account for non-constant parameters.

By defining sensor models that relate the robot poses, calibration constraint and sensor measurements, these parameters can be included in the optimisation problem. Furthermore, robustification techniques allow for the constraint covariance to be included as well. Hence, an extended self-calibrating pose-graph optimisation problem is defined that will be examined in the subsequent thesis work.

Contents

List of Figures

List of Tables

Preface

This literature survey is the initial phase of my MSc-thesis project that is required for obtaining a Master of Science degree in Systems and Control at Delft University of technology. The purpose of this survey is to get acquainted with the concerned research subject and the prior work that has been conducted by the scientific community.

I would like to thank my supervisors prof.dr.ir. Tamás Keviczky (DCSC) and ir. Martijn Krijnen (Demcon Advanced Mechatronics B.V.) for their assistance and feedback throughout writing this literature survey.

Delft, University of Technology
6th January 2021

Art van Liere

Chapter 1

Introduction

A problem central to mobile robotics is that of *navigation*. Mobile robots have the capability to move around in their environment and are not fixed to one physical location. As such, they can act as a powerful extension to our capabilities by performing tasks that we, as humans, are not suited for. Mobile robotics can even be employed to *automate* a variety of tasks. This is where the problem of navigation comes in. Avoiding dangerous situations, such as collisions or unsafe conditions, is important, but reaching target locations might even be more so. Robust navigation requires the robot to have a sense of spatial awareness, and this is what the navigation problem is trying to solve.

Examining the methods that we as humans use to solve the navigation problem gives insight into the challenges robots face when encountering the same problem. We use the following tricks to navigate our surroundings; each of which has robotic navigation equivalent:

1. We use our stereoscopic exteroceptive sensors (or eyes) to recognise distinctive features in our environment (i.e., certain remarkable objects or landmarks) and estimate their distance from us. This gives us a sense of relative position and direction with respect to such a feature.
2. We track these recognised features in our field of view as we move around to build relations based between our current and previous position in space. This gives us a sense of change of position and orientation.
3. Similarly, we might count our steps to estimate our change in position and orientation.
4. We memorise views by recognising a distinctive set of features within the scene and observing the relations between it and its surroundings.
5. We build a mental map of our environment from which we could extrapolate observations, with predictions such as "If I move straight, I will probably encounter that same landmark that I came across a while ago".
6. Whenever we re-encounter past-seen landmarks, we establish a sense of our relative position and orientation with respect to our past self that made the previous sighting. We accept that our path has crossed itself, which strengthens the topology of our mental map.

The problem of navigation is formalised as the *Simultaneous Localisation and Mapping (SLAM)* problem [2, 3, 4, 5, 6, 7, 8, 9, 10]. The SLAM problem asks whether it is possible for the robot to incrementally build a consistent map of an unknown environment while simultaneously determining its location within this map.

Modern solutions to the SLAM problem are based on the *graph*-based formulation of the robot trajectory and environment [4, 7, 11, 8, 10]. For such an approach, the robot trajectory and/or environment is discretised as a set of reference locations in space (and time), which are represented as *nodes* in the graph. The geometric relations between nodes are represented by *edges*, where each is in the form of a soft constraint as derived from observations. By defining, for each constraint, a cost function that depends on the deviation of the proposed solution from the constraint, as well as the certainty of that constraint, the SLAM problem can be stated as an optimisation problem. Solving SLAM then means finding the set of robot poses (i.e., positions and orientations with respect to an inertial reference frame) and/or landmarks that minimises the total deviation from the constraints over all relations. That is, the solution is the optimal compromise of robot poses and/or landmarks over all constraints, and should best fit the set of observations.

Each of the intuitive 'navigation tricks' mentioned above has an equivalent in graph-based SLAM:

1. Exteroceptive sensors (e.g. laser-based or vision-based range sensors) are used to measure the distance to and direction of features in the environment. This positional estimate establishes a relation between the robot pose at the point of observation and the observed landmark in the form of a *feature constraint*.
2. The same exteroceptive sensors are used to track surrounding objects between different poses. The estimates that follow from these observations establish a relation between subsequent poses in the form of a *proximity constraint*, and, more strictly, between consecutive poses in the form of *odometry constraints*.
3. Interoceptive sensors (e.g. encoders for wheel rotation, accelerometers for acceleration or gyroscopes for turn rate) are used to predict the next pose from the current pose. These estimates establish a relation between consecutive poses in the form of an *odometry constraint*.
4. All observations are stored in *memory*, such that, when a similar observation is made, relations can be established between the poses from which these similar observations have been made.
5. A virtual *map* is created, and predictions are made, based on an estimate of the current robot location regarding which features are likely to be observed again soon.
6. Based on a prediction of the current robot location, new observations of features are matched to previously seen features in order to establish a spatial relation between poses in the form of a *loop-closure constraint*.

All of these terms will be elaborated upon in subsequent chapters, but this analogy between robot and human provides intuition behind some of the actions performed during SLAM. Furthermore, this line of thinking can act as a handy tool with which new methodologies can be derived.

In essence, the SLAM problem and its solution are fully defined by the relations that exist

between the poses. In conventional approaches to SLAM, the certainty of a constraint acts as a linear scaling factor for the constraint cost. As such, the constraint certainty acts as a ratio between constraint types with respect to which the optimal compromise is searched for (assuming a strong correlation between constraint type and constraint certainty). Since this ratio defines the SLAM problem, it cannot be simply included in the standard SLAM optimisation problem, making it a hard quantity to estimate.

The constraint certainty is represented by a covariance matrix and is usually considered as a tuning parameter. Its value is derived from the uncertainty (or noise model) of the sensors that make the observations that correspond to the constraint. Not only can it be difficult to estimate, but it might also change over time or depending on the environment. In light of its problem-defining nature, having a way to estimate its value on-the-go might make the approach more robust to initialisation errors, as well as potentially improve the accuracy of the solution.

Another class of parameters that could benefit from on-the-go estimation relate to the models that derive the relations between poses from the incoming sensor measurements. These models are dependent on calibration parameters, such as wheel radii, the sensor position on the robot, or the baseline distance between the cameras in a stereoscopic setup. The accuracy of these calibration parameters directly impacts the accuracy of the constraints, and therefore also the solution. These quantities might be prone to change over time and a constant representations might (therefore) be difficult to estimate.

1-1 SLAM history

The earliest and, until more recently, the most popular approaches to SLAM were filtering-based approaches. The most common approaches were based on the Extended Kalman Filter (EKF) and the Particle Filter [12, 13]. These filtering approaches estimate the Gaussian density [14, 15, 16] over the current pose and the positions of all landmarks, which are all included in a state-vector. Filtering approaches have shown to be inconsistent when applied to inherently non-linear SLAM problems, since they make predictions by linearising the state equation at the current best state estimate. The problem is that the updating state-vector results in an inability to modify past linearisation choices when future observations render this linearisation point inaccurate [17].

This opens the door to *smoothing* approaches to SLAM (commonly referred to as SAM: Smoothing and Mapping [18, 19, 20]), which does not just involve estimation of the most current robot location, but the entire robot trajectory up to the current point [21], along with all landmarks. This solves the problem of permanent linearisation, since the point of linearisation can be re-chosen during optimisation. The problem of optimally estimating the entire set of sensor poses along with the parameters of all features in the environment is described by the *full SLAM problem* [22]. The graph-based formulation of SLAM that is discussed above is widely studied as a tool to solve the full SLAM problem.

As a modification, pose graph SLAM (or pose SLAM [23]) formulates the problem as estimating solely over the robot poses from where the measurements were taken. In pose SLAM, the sensor observations are used to obtain an estimate of the relationship between the robot poses from which these observations were taken, without using an explicit geometric representation

of the environment [13].

1-2 Demcon

In an effort to gain expertise in the field of robotic navigation, Demcon Advanced Mechatronics Delft B.V. has assembled the *Robotics Competence Group*. To supplement this development, the company has built a test platform based on the BigBot wheeled robot. BigBot is equipped with the following equipment:

- Velodyne Puck (VLP-16) LiDAR sensor that outputs a depth map of its ($360° \times 30°$) field-of-view, consisting of a 300 000 point-cloud .
- Intel RealSense (D435) RGB-D camera that outputs a wide-angle FullHD RGB image at ($69.4° \times 42.5°$) with corresponding HD depth data af ($86° \times 57°$).
- XSens MTi 1-series IMU that outputs accelerometer, gyroscope and magnetometer data.
- Differential drive system, consisting of two motors and corresponding wheel encoders.

Additionally, a ceiling-mounted camera tracks the ground-truth position of the robot within the test environment. This ground truth data can be used to quantify the accuracy of the SLAM algorithm.

Its SLAM approach runs on top of the *Robot Operating System* (ROS) [24], which is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms. ROS implements a peer-to-peer network of processes (or *ROS nodes*) in a computation graph. These nodes communicate with each other by passing standardised messages, which are routed via a transport system with publish/subscribe semantics. Due to the modularity of the ROS architecture, alternate processes can be written and swapped out without much hassle.

The goal of the expertise gained by the Robotics Competence Group is to assist customers in building custom navigation and mapping systems. Due to the maturity of SLAM research, very capable open-source off-the-shelf approaches are available to be used and extended. However, using such an off-the-shelf algorithm is not as simple as downloading the codebase and running it on the robot of choice. Each system requires an extensive setup (or configuration), which requires in-depth understanding of the problem at hand. This problem motivates the direction of research.

1-3 Proposed work

This document has been written with the aim of finding a research topic in the field of graph-based SLAM. By taking interest in an approach to SLAM that requires minimal set-up, a form of *self-calibrating SLAM* is decided upon for research.

Two types of calibration parameters (which are assumed constant and set by the user in traditional SLAM) are considered for on-the-go estimation in this new form self-calibrating SLAM:

- The *sensor model parameters*, which range from geometric constants (such as the sensor position on a rigid frame, or baseline in stereoscopic systems) to variables that are not measurable a priori (such as the sensor bias, or magnetic field perturbations).

  Traditionally, raw sensor data is processed and fed into a sensor model, which then calculates, based on the raw data and the sensor model parameters, the *constraint value* — this is the geometric relation that perfectly matches the observation. As such, this value defines the global minimum of the cost function term corresponding to this constraint. The solution will have a natural tendency towards satisfying this relation exactly. Therefore, having a better estimate of these parameters naturally improves the constraint accuracy, and, therefore, the solution quality.

- The *constraint covariance*, which defines the confidence that the optimiser should have in the corresponding constraint. A low constraint covariance indicates an accurate constraint. In that case, the optimal solution should therefore not deviate much from the constraint value. This is stimulated by penalising deviations from the constraint value with a cost that in weighted linearly with the inverse of the constraint covariance.

  Since the actual cost value is not of importance during optimisation, the ratio of covariances essentially indicates relative constraint 'strength' (where constraints with low covariance are considered stronger constraints that assert more power on the solution). This ratio defines which constraints are more lenient when it comes to the optimal 'compromise' that defines the solution. Therefore, having an accurate estimate of the constraint covariances (and more specifically the ratio of constraint covariances) should improve the solution quality.

A constraint is fully defined by the *constraint value* and *constraint covariance*. Since the SLAM solution is the optimal compromise over the set of all constraints, improving the quality of constraint through live estimation should improve the quality of the quality of the solution. Furthermore, self-calibration allows for a simpler setup process and alleviates the user from performing a lengthy calibration process.

The calibration parameters, on which the sensor models are dependent, can be treated as optimisation parameters. These parameters are considered to have a ground-truth value that is sought by the optimisation problem.

A similar ground-truth goal can be taken into consideration for the estimation of the covariance matrices. For instance, an adaptive Kalman filter could be employed to find a moving estimate of the covariances, albeit independently of the solution cost. Alternatively, the covariance matrices can be considered tuning parameters, resulting in 'low-cost' solutions not necessarily being close to the true value. In that case, the covariance matrix can be treated as an optimisation variable. This, however, requires the modification of the optimisation problem to prohibit the trivial solution of infinite covariance.

This document proposes improvements of self-calibrating SLAM by the application of:

- A new formulation for sensor models used to highlight the dependence on calibration parameters.
- An approach to estimate the covariance matrices of each sensor system on-the-go.
- Saving and visualising the calibrated parameters topologically with a map.

1-4 Report outline

The report is structured as follows:

1. Chapter 2 introduces Lie theory and an accompanying mathematical formulation of kinematics and dynamics. This mathematical foundation serves as a basis on which the majority of notation is built and allows for the elegant description of otherwise complicated sensor models.
2. Chapter 3 discusses the modern graph-based SLAM framework. It describes the data flow from sensor to optimiser and describes the division between the front-end and back-end systems. The definition of the front-end as an abstraction layer between sensor system and optimiser neatly defines the scope of the back-end, and, with that, the scope of research.
3. Chapter 4 explains the graph-based formulation and derives the SLAM problem as a non-linear least-squares optimisation problem.
4. Chapter 5 discusses modern optimisation techniques.
5. Chapter 6 discusses techniques that make the back-end more robust to erroneous loop-closure constraints. These techniques offer insight into alternative optimisation schemes that might prove usable for live covariance calibration.
6. Chapter 7 discusses on-the-go calibration. For sensor model parameters, estimation is done by including the sensor models in the optimisation. As such, the sensor models are derived in terms of the Lie theory formulation. For covariance estimation, multiple techniques are discussed, of which the performance is to be researched in the thesis.
7. Chapter 8 summarises the problem statement and discusses the thesis work.

Chapter 2

Lie groups for 2D and 3D transformations

The SLAM problem seeks to estimate the robot trajectory and a surrounding map that best matches the observations made over a given time period. Since this problem concerns the estimation of objects in multi-dimensional space, some mathematical tools are required that allow for expression and manipulation of these objects in said space. Note that SLAM problem can be considered in both 2D and 3D, where the problem dimensionality is represented by $n \in \{2, 3\}$. In most parts of this work nothing is assumed regarding the dimensionality of the SLAM problem. However, for deriving rigid body motion, the full 3D description is considered ($n = 3$), as the simpler 2D description can easily be derived from the full 3D description.

2-1 Rigid body motion

The robot is considered a rigid body, which is defined as a collection of particles such that the distance between any two particles remains fixed, regardless of any motions of the body or forces exerted on the body. Its continuous rigid motion is discretised as a set of rigid displacements, and each is described by a rigid body transformation. For such a transformation, distance is preserved between all points in $\mathbb{R}^3$, and the cross product is preserved between all vectors in $\mathbb{R}^3$ [25]. As such, particles in a rigid body can only rotate, but not translate, with respect to each other.

To keep track of the rigid body motion, a right-handed Cartesian coordinate frame is fixed at some point of the body (usually the odometric centre). The body-fixed coordinate frame is then expressed in terms of the inertial reference frame by a rigid body transformation, which defines a *rotation* and *translation*. This rigid body transformation is what is implied by the robot *pose*.

Describing a rotation of one reference frame relative to another requires special care. Planar rotation, for instance, is usually expressed with an angle $\theta \in \mathbb{R}$, and scalar addition is used to

compound multiple angles. This operation does not respect the circle topology; that is, the wrap-around property of angles that stipulates that $\theta \in [0, 2\pi)$ or $\theta \in [-\pi, \pi)$. Of course, θ can be converted to comply with the intended range of this planar rotation parameterisation, but this is an extra operation that is preferably avoided.

Rotation matrices are special matrices that can be used to express the relative orientation of one coordinate frame in terms of another. Compounding multiple rotation matrices by matrix multiplication results in another rotation matrix; i.e., no additional steps are required to make rotation matrices respect the required circle topology. By exploiting its special structure, a rotation matrix in $\mathbb{R}^3$ can be parameterised by 3 numbers. The most common choice is to make use of the vector of Euler angles: yaw, pitch, and roll. These angles represent a relative rotation via a series of rotations about each axis. Although intuitive, this representation suffers from singularities at gimbal lock [26], which occurs when two of the three rotational axes align. Alternatively, quaternions, which generalise complex numbers, can be used to represent rotations in much the same way as complex numbers on the unit circle can be used to represent planar rotations. Although these do not suffer from singularities, they parameterise rotations by 4 numbers, making them over-parameterised. In order to find an efficient parameterisation, rotation matrices have to be examined more closely.

The special structure of rotation matrices is described by a rotation manifold, which can be expressed using Lie theory. Lie theory allows for the representation of manifolds that are useful for the expression rigid motion and kinematics. Lie theory can be exploited to devise a coherent and robust framework for representing 3D transformation.

2-2 Lie theory

Lie theory (and the accompanying *Lie groups*) are useful for formulating estimating problems properly; that is, proper modelling of the states and measurements, the functions relating them, and their uncertainty. In particular, such modelling designs utilise manifolds, which are essentially the smooth topological surfaces of the Lie groups, where the state representation evolve.

2-2-1 Lie groups

The Lie group encompasses the concepts of *group* (1) and *smooth manifold* (2) in a unique body; that is, a Lie group $\mathcal{G}$ is a smooth manifold whose elements satisfy the group axioms [1]. These concepts are defined as:

1. A group $(\mathcal{G}, \circ)$ is a set, $\mathcal{G}$, equipped with a binary composition operation, $\circ$, that combines any two elements to form a third element in such a way that four conditions called group axioms are satisfied, namely:
   - *Closure under* $\circ$: If $g_1, g_2 \in \mathcal{G}$, then $g_1 \circ g_2 \in \mathcal{G}$;
   - *Associativity*: If $g_1, g_2, g_3 \in \mathcal{G}$, then $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$;
   - *Identity* e: There exists an identity element $e \in \mathcal{G}$, such that $e \circ g = g \circ e = g$, for every $g \in \mathcal{G}$;

- *Invertibility* g^{-1}: For each $g \in \mathcal{G}$, there exists a (unique) inverse $g^{-1} \in \mathcal{G}$ such that $g^{-1} \circ g = g \circ g^{-1} = e$.

2. A smooth (or differentiable) manifold is a topological space that locally resembles linear space. Its smoothness implies the existence of a unique tangent space at each point. It can be visualised as a curved, smooth (hyper)-surface, with no edges or spikes, embedded in a space of higher dimension.

In a Lie group, the manifold looks identical at every point. Lie groups join the local properties of smooth manifolds, which allow calculus to be performed, with the global properties of groups, which enable non-linear composition of distant objects.

2-2-2 Lie algebra

Every Lie group $\mathcal{G}$ has an associated Lie algebra $\mathfrak{g} := T_e\mathcal{G}$, which is defined as the *tangent space at the identity element* $e \in \mathcal{G}$. That is, the Lie algebra is the vector space generated by differentiating the group transformations along chosen directions in the space, at the identity transformation [27]. Its definition involves the identity element, as this is the only element that every group is guaranteed to have (i.e., the Lie algebra is automatically defined for every Lie group). The tangent spaces at all group elements are *isomorphic* (i.e., there exists an *isomorphism*, or structure-preserving mapping between two structures of the same type that can be reversed by an inverse mapping). Therefore, it is sufficient to study only the tangent space at the identity element.

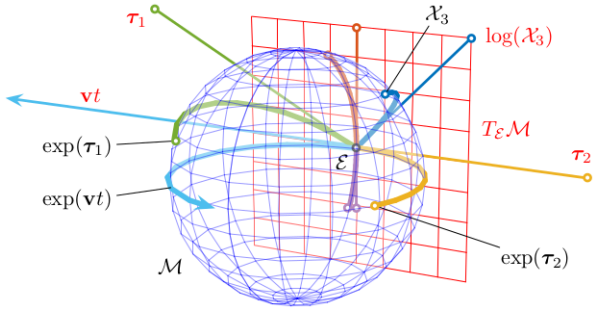

Figure 2-1: Representation of the relation between the Lie group $\mathcal{M}$ (blue sphere) and Lie algebra $\mathfrak{m}$ (red plane), which is the tangent space at the identity $\mathcal{E} \in \mathcal{M}$ [1]. Through the exponential map, each straight path $\mathbf{v}t$ through the origin on the Lie algebra produces a path $\exp(\mathbf{v}t)$ around the manifold.

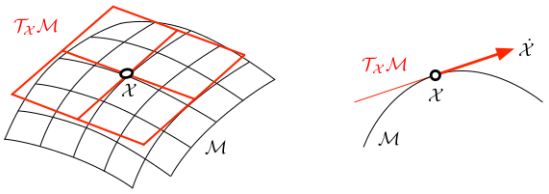

Figure 2-2: A manifold $\mathcal{M}$ and the vector space $T_\mathcal{X}\mathcal{M} \cong \mathbb{R}^2$ tangent at the point $\mathcal{X} \in \mathcal{M}$, and a convenient side-cut [1]. The velocity element, $\dot{\mathcal{X}} = \partial\mathcal{X}/\partial t$, does not belong to the manifold $\mathcal{M}$, but to the tangent space $T_\mathcal{X}\mathcal{M}$.

The Lie algebras are closely related to Lie groups, as can be seen in Figure 2-1. Each straight path through the origin on the Lie algebra produces a (curved) path around the manifold, which runs along the respective geodesic. Conversely, each element of the Lie group has an equivalent in the Lie algebra. This relation is so profound that (nearly) all operations in the Lie group, which is curved and non-linear, have an exact equivalent in the Lie algebra, which is a linear vector space. This correspondence allows one to study the structure and

classification of Lie groups in terms of Lie algebras.

Importantly, a tangent space associated with a Lie group $\mathcal{G}$ provides a convenient space in which to represent differential quantities related to the group. That is, velocities and Jacobians are well-represented in the tangent space around a transformation. This is due to the following principles [27]:

- The Lie algebra $\mathfrak{g}$ is a vector space with the same dimension $n_\mathcal{G}$ as the number of degrees of freedom of the group transformation $\mathcal{G}$. Its basis elements E_i, which are called *generators*, are each the time-derivative around the origin in the i-th direction. All tangent vectors represent linear combinations of the generators, and can therefore be identified by vectors in $\mathbb{R}^{n_\mathcal{G}}$.

  If the vector $\boldsymbol{\tau} \in \mathbb{R}^{n_\mathcal{G}}$ identifies an element of Lie algebra $\mathfrak{g}$, the corresponding element is generally denoted with a 'hat' decorator (with $\mathbf{v}^\wedge$ for velocities and $\boldsymbol{\tau}^\wedge = (\mathbf{v}t)^\wedge = \mathbf{v}^\wedge t$ for general elements). To pass from $\mathfrak{g}$ to $\mathbb{R}^{n_\mathcal{G}}$ and vice versa, two mutually inverse linear maps (or isomorphisms), *vee* and *hat*, are defined as

$$
\begin{aligned}
(\cdot)^\wedge \text{ (hat)}: &\quad \mathbb{R}^{n_\mathcal{G}} \to \mathfrak{g}: \quad \boldsymbol{\tau} \mapsto \boldsymbol{\tau}^\wedge = \sum_{i=1}^{n_\mathcal{G}} \tau_i E_i, \\
(\cdot)^\vee \text{ (vee)}: &\quad \mathfrak{g} \to \mathbb{R}^{n_\mathcal{G}}: \quad (\boldsymbol{\tau}^\wedge)^\vee = \boldsymbol{\tau} = \sum_{i=1}^{n_\mathcal{G}} \tau_i \mathbf{e}_i,
\end{aligned}
$$

  with $\mathbf{e}_i$ the base vectors of $\mathbb{R}^{n_\mathcal{G}}$ (where $\mathbf{e}_i^\wedge = E_i$). A subscript can be added to specify the tangent space (e.g., similar to Figure 2-2, for moving point $g(t) \in \mathcal{G}$ with velocity $\dot{g} = \partial g / \partial t = \mathbf{v}_g^\wedge \in T_g\mathcal{G}$ expressed in the *local* tangent space of g). Consequently, it can be deduced that $\mathfrak{g}$ is isomorphic to the vector space $\mathbb{R}^{n_\mathcal{G}}$ — this is written as $\mathfrak{g} \cong \mathbb{R}^{n_\mathcal{G}}$ (or $\boldsymbol{\tau}^\wedge \cong \boldsymbol{\tau}$).

  Since the velocity of a point moving on a Lie group's manifold belongs to the tangent space, the structure of the Lie algebra can be found by time-differentiating the group constraint of invertibility (i.e., for each $g \in \mathcal{G}$, there exists a (unique) inverse $g^{-1} \in \mathcal{G}$ such that $g^{-1} \circ g = g \circ g^{-1} = e$) [1]. For multiplicative groups, this constraints results in $g^{-1}\dot{g} + (\dot{g^{-1}})g = 0$, which applies to elements tangent at g. The elements of the Lie group are therefore of the form

$$
\mathbf{v}^\wedge = g^{-1}\dot{g} = -(\dot{g^{-1}})g. \tag{2-1}
$$

- The *exponential map* converts any element $\boldsymbol{\tau}^\wedge \in \mathfrak{g}$ *exactly* into a transformation $g \in \mathcal{G}$. Intuitively, this can be thought of as wrapping the tangent element around the manifold following the geodesic. The inverse *logarithmic map* can then be though of as the unwrapping operation. Both are isomorphisms, defined as

$$
\begin{aligned}
\exp: &\quad \mathfrak{g} \to \mathcal{G}: \quad \boldsymbol{\tau}^\wedge \mapsto g = \exp(\boldsymbol{\tau}^\wedge), \\
\log: &\quad \mathcal{G} \to \mathfrak{g}: \quad g \mapsto \boldsymbol{\tau}^\wedge = \log(g).
\end{aligned}
$$

  Closed forms of the exponential in multiplicative groups are obtained by writing the absolutely convergent Taylor series,

$$
\exp\left(\boldsymbol{\tau}^\wedge\right) = \sum_{i=k}^{\infty} \frac{1}{k!} (\boldsymbol{\tau}^\wedge)^k. \tag{2-2}
$$

An additional exponential mapping can be defined that directly maps vector elements $\boldsymbol{\tau} \in \mathbb{R}^{n_\mathcal{G}} \cong \mathfrak{g}$ with elements $g \in \mathcal{G}$. These isomorphisms are denoted by the *capitalised exponential map* and defined as

$$\begin{aligned} \text{Exp}: &\quad \mathbb{R}^{n_\mathcal{G}} \to \mathcal{G}: \quad \boldsymbol{\tau} \mapsto g = \text{Exp}(\boldsymbol{\tau}) = \exp(\boldsymbol{\tau}^\wedge), \\ \text{Log}: &\quad \mathcal{G} \to \mathbb{R}^{n_\mathcal{G}}: \quad g \mapsto \boldsymbol{\tau} = \text{Log}(g) = (\log(g))^\vee. \end{aligned}$$

- The *adjoint linearly* and *exactly* transforms a tangent vector from one tangent space to another. That is, the adjoint of $\mathcal{G}$ at $x \in \mathcal{G}$ is defined as

$$\text{Ad}_x: \quad \mathfrak{g} \to \mathfrak{g}: \quad \boldsymbol{\tau}^\wedge \mapsto \text{Ad}_x(\boldsymbol{\tau}^\wedge) := x\boldsymbol{\tau}^\wedge x^{-1},$$

  and can be used to transforms $\boldsymbol{\tau}^\wedge$ from tangent space $T_x\mathcal{G}$ to tangent space $T_e\mathcal{G} = \mathfrak{g}$, with $\boldsymbol{\tau}_e^\wedge = \text{Ad}_x(\boldsymbol{\tau}_x^\wedge)$. Since the adjoint is a linear *homomorphism* (i.e., a structure-preserving map between two algebraic structures of the same type), an equivalent *adjoint matrix* operator can be found that maps the Cartesian tangent vectors $\boldsymbol{\tau}_e \cong \boldsymbol{\tau}_e^\wedge$ and $\boldsymbol{\tau}_x \cong \boldsymbol{\tau}_x^\wedge$:

$$[\text{Ad}_x]: \quad \mathbb{R}^{n_\mathcal{G}} \to \mathbb{R}^{n_\mathcal{G}}: \quad \boldsymbol{\tau}_x \mapsto \boldsymbol{\tau}_e = [\text{Ad}_x]\boldsymbol{\tau}_x$$

  The adjoint property is what ensures that the tangent spaces at all group elements are isomorphic (i.e., that the tangent space has the same structure at all points on the manifold), since any tangent vector can be transformed back to the tangent space around the identity.

To conclude, the Lie algebra can be considered as a linearisation of the Lie group (near the identity element), and the exponential map provides the "delinearisation". As a result of this isomorphism, the vector that identifies the Lie algebra efficiently parameterises the Lie group element.

2-2-3 Encapsulation

Encapsulation is a fundamental of object-oriented programming, which refers to the principle that data inside an object should only be accessed through a public interface, thereby preventing direct access by unauthorised parties. In this setting, encapsulation means that the the manifold can be treated as a black box with only two possibilities for access: via the *plus* and *minus* operators, denoted by $\oplus$ and $\ominus$ [28, 1]. As such, the encapsulation limits operations on the manifolds that might void the constraints internal to its definition.

The two operators combine one Exp/Log operation with one composition. They are defined as follows:

- The *plus* operator $\oplus$ defines a homomorphism that is used to find the updated group element $y \in \mathcal{G}$, obtained by addition of increment $\boldsymbol{\tau}_x \in \mathbb{R}^{n_\mathcal{G}}$ from $x \in \mathcal{G}$. The homomorphism is defined by

$$\oplus: \mathcal{G} \times \mathbb{R}^{n_\mathcal{G}} \to \mathcal{G}: x \times \boldsymbol{\tau}_x \mapsto y = x \oplus \boldsymbol{\tau}_x := x \circ \text{Exp}(\boldsymbol{\tau}_x) \in \mathcal{G}. \tag{2-3}$$

  The increment $\boldsymbol{\tau}_x \in \mathbb{R}^{n_\mathcal{G}}$ is expressed in the *local* tangent space at $x \in \mathcal{G}$, similarly to $\boldsymbol{\theta}$ in Figure 2-3.

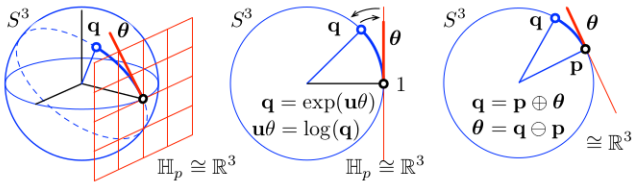

Figure 2-3: The $\mathcal{S}^3$ manifold is a unit 3-sphere (blue) in the 4-space of quaternions [1]. The Lie algebra $\mathbb{H}_p$ is isomorphic to the hyper-plane $\mathbb{R}^3$ (red grid). The tangent vector $\boldsymbol{\theta} \in \mathbb{R}^3$ (red segment) wraps the manifold over the geodesic (dashed). The centre and right figures show a side-cut through this geodesic. Mappings exp and log (arrows) map (wrap and unwrap) elements of the Lie algebra to/from elements of the Lie group. In the right figure, the increment $\boldsymbol{\theta} \in \mathbb{R}^3$ is defined in the local tangent space at $\mathbf{p}$.

- The *minus* operator $\ominus$ can be thought of as the inverse of $\oplus$ (for all $x, y \in \mathcal{G}$, it holds that $x \oplus (y \ominus x) = y$). That is, it finds the difference between two group elements, expressed in the local tangent space of the subtracted group element. The homomorphism is defined by

$$\ominus : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^{n_\mathcal{G}} : y \times x \mapsto \boldsymbol{\tau}_x = y \ominus x := \mathrm{Log}(x^{-1} \circ y) \in T_x\mathcal{G}. \tag{2-4}$$

  The difference $\boldsymbol{\tau}_x \in \mathbb{R}^{n_\mathcal{G}}$ is again expressed in the local tangent space at $x \in \mathcal{G}$, similarly to Figure 2-3.

As a result of this encapsulation, optimisation algorithms working on $\mathbb{R}^{n_\mathcal{G}}$ work essentially the same way on $\mathcal{G}$: state updates are performed with $\oplus$, while state differences are determined with $\ominus$.

Because of the non-commutativity of the composition, an alternative 'left' version can also be defined as

$$\begin{aligned} \text{left-}\oplus &: \mathbb{R}^{n_\mathcal{G}} \times \mathcal{G} \to \mathcal{G} : \boldsymbol{\tau}_e \times x \mapsto \quad y = \boldsymbol{\tau}_e \oplus x := \mathrm{Exp}(\boldsymbol{\tau}_e) \circ x \in \mathcal{G}, \\ \text{left-}\ominus &: \mathcal{G} \times \mathcal{G} \to \mathbb{R}^{n_\mathcal{G}} : y \times x \mapsto \quad \boldsymbol{\tau}_e = x \ominus y := \mathrm{Log}(y \circ x^{-1}) \in T_e\mathcal{G} = \mathfrak{g}. \end{aligned}$$

Here, the increment or difference is defined in the *global* tangent space. While left- and right-$\oplus$ are distinguished by operand order, the notation for $\ominus$ is ambiguous. Perturbations, however, are always expressed locally, and therefore the right forms of $\oplus$ and $\ominus$ are used by default.

The left- and right- versions can be utilised to reveal the structure of the adjoint. By identifying $y \in \mathcal{G}$ and setting both versions of $\oplus$ to be equal, the adjoint action is derived as

$$\boldsymbol{\tau}_e \oplus x = x \oplus \boldsymbol{\tau}_x \quad \Longrightarrow \quad \exp(\boldsymbol{\tau}_e^\wedge) = x \exp(\boldsymbol{\tau}_x^\wedge) x^{-1} = \exp(x \boldsymbol{\tau}_x^\wedge x^{-1}) \quad \Longrightarrow \quad \boldsymbol{\tau}_e^\wedge = x \boldsymbol{\tau}_x^\wedge x^{-1}.$$

2-3 Poses

As mentioned in Section 2-1, a robot *pose* is the rigid body transformation that describes the body-fixed coordinate frame as seen from the inertial reference frame in terms of a *rotation* and *translation*. Rotations and transformations are conveniently described by *matrix Lie groups*, which are closed subgroups of the *General Linear* group, $\mathcal{GL}(n)$, of $n \times n$ invertible matrices with entries in $\mathbb{C}$. For matrix Lie groups, the composition action is matrix multiplication.

2-3-1 Rotations

The orientation of a body is described by the relative orientation between a body-fixed coordinate frame and a fixed (or inertial) coordinate frame. Let B be the body-fixed coordinate frame and A be the inertial coordinate frame. The rotation matrix $R_{\mathsf{B}}^{(\mathsf{A})}$ describes the orientation of B (subscript) relative to A (superscript). For the 3-dimensional case, it is obtained by stacking the coordinate vectors of the principal axes $\mathbf{x}_{\mathsf{B}}^{(\mathsf{A})}, \mathbf{y}_{\mathsf{B}}^{(\mathsf{A})}, \mathbf{z}_{\mathsf{B}}^{(\mathsf{A})} \in \mathbb{R}^3$ of B as seen from A; that is,

$$R_{\mathsf{B}}^{(\mathsf{A})} = \begin{bmatrix} \mathbf{x}_{\mathsf{B}}^{(\mathsf{A})} & \mathbf{y}_{\mathsf{B}}^{(\mathsf{A})} & \mathbf{z}_{\mathsf{B}}^{(\mathsf{A})} \end{bmatrix} \in \mathcal{SO}(3),$$

where rotations matrices are represented by the *Special Orthogonal* group $\mathcal{SO}(n)$. With *Special* referring to unit determinant ($\det R = 1$), and *Orthogonal* referring to the orthogonality constraint ($R^\top R = I_n$), the rotation group is defined as

$$\mathcal{SO}(n) = \{R \in \mathbb{R}^{n \times n} : R^\top R = I_n, \det(R) = 1\} \subset \mathcal{GL}(n).$$

Every configuration of a rigid body that is free to rotate relative to a fixed frame can be identified with a unique $R_{\mathsf{B}}^{(\mathsf{A})} \in \mathcal{SO}(3)$. Let $\mathbf{q}^{(\mathsf{B})} = (q_1^{(\mathsf{B})}, q_2^{(\mathsf{B})}, q_3^{(\mathsf{B})}) \in \mathbb{R}^3$ be the coordinates of $\mathbf{q}$ relative to frame B. Since the elements $q_1^{(\mathsf{B})}, q_2^{(\mathsf{B})}, q_3^{(\mathsf{B})} \in \mathbb{R}$ are projections of $\mathbf{q}$ onto the coordinate axes of B, the coordinates of $\mathbf{q}$ relative to frame A are given by

$$\mathbf{q}^{(\mathsf{A})} = \mathbf{x}_{\mathsf{B}}^{(\mathsf{A})} q_1^{(\mathsf{B})} + \mathbf{y}_{\mathsf{B}}^{(\mathsf{A})} q_2^{(\mathsf{B})} + \mathbf{z}_{\mathsf{B}}^{(\mathsf{A})} q_3^{(\mathsf{B})} = R_{\mathsf{B}}^{(\mathsf{A})} \mathbf{q}^{(\mathsf{B})}.$$

Accordingly, $R_{\mathsf{B}}^{(\mathsf{A})}$ can be considered a linear map that rotates the coordinates of a point from frame B to A. The intuition behind this operation can be twofold:

- Frames A and B are misaligned. Rotation $R_{\mathsf{B}}^{(\mathsf{A})}$ encodes this misalignment and can be used to modify the expression of spatially-fixed $\mathbf{q}$ from coordinate frame B to coordinate frame A.
- Frames A and B are aligned, and $\mathbf{q}$ is fixed to frame B, but expressed in A. Rotation $R_{\mathsf{B}}^{(\mathsf{A})}$ moves B relative to A, thereby modifying the expression of $\mathbf{q}$ within A.

Rotation matrices can be defined for both the 3-dimensional and 2-dimensional case:

1. Rotations in 3D space are represented by elements of the spatial rotation Lie group $\mathcal{SO}(3)$. Its properties are described as follows:
   - The structure of the Lie algebra $\mathfrak{so}(3)$ follows from Equation (2-1); namely, for $R \in \mathcal{SO}(3)$ it follows that $R^\top \dot{R} = -(R^\top \dot{R})^\top = [\boldsymbol{\omega}]_\times \in \mathfrak{so}(3)$, where $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3) \in \mathbb{R}^3$ denotes the vector of angular velocities. This reveals that the Lie algebra $\mathfrak{so}(3)$ is the set of 3×3 skew-symmetric matrices, denoted by

     $$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \in \mathfrak{so}(3).$$

     A skew-symmetric matrix $[\mathbf{a}]_\times$ denotes the linear operator $\mathbf{b} \mapsto \mathbf{a} \times \mathbf{b}$, which is equivalent to the cross product with $\mathbf{a} \in \mathbb{R}^3$. This follows intuitively from the differential equation $\dot{\mathbf{q}}(t) = \boldsymbol{\omega} \times \mathbf{q}(t)$ describing the tip point trajectory of point $\mathbf{q}(t)$ due to a rotation $\boldsymbol{\omega}$.

The generators of $\mathfrak{so}(3)$ correspond to the derivatives of rotation around each of the standard axes, evaluated at identity:

$$E^{(1)}_{\mathcal{SO}(3)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \quad E^{(2)}_{\mathcal{SO}(3)} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad E^{(3)}_{\mathcal{SO}(3)} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

The Lie algebra is a vector space whose elements can be decomposed into

$$\boldsymbol{\theta}^\wedge = [\boldsymbol{\omega} t]_\times = \theta_1 E^{(1)}_{\mathcal{SO}(3)} + \theta_2 E^{(2)}_{\mathcal{SO}(3)} + \theta_3 E^{(3)}_{\mathcal{SO}(3)} \in \mathfrak{so}(3),$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3) = \boldsymbol{\omega} t \in \mathbb{R}^3$ is the identifying vector of integrated rotations over duration t (assuming constant angular velocity $\boldsymbol{\omega}$ for duration t).

- From the structure of the Lie algebra, the ordinary differential equation $\dot{R} = R[\boldsymbol{\omega}]_\times \in T_R\mathcal{SO}(3)$ can be derived. Its solution $R(t) = R_0 \exp([\boldsymbol{\omega} t]_\times)$, where $R_0 \in \mathcal{SO}(3)$ is the initial rotation. With $R_0 = I_3$, the spatial rotation matrix R follows from Equation (2-2) as

$$R(t) = \exp\left([\mathbf{u}]_\times \theta\right) = \sum_k \frac{\theta^k}{k!} \left([\mathbf{u}]_\times\right)^k \in \mathcal{SO}(3),$$

where $\mathbf{u}\theta := \boldsymbol{\theta} = \boldsymbol{\omega} t \in \mathbb{R}^3$ defines the integrated rotation in angle-axis form, with angle θ and unit axis $\mathbf{u}$. By taking advantage of the properties of skew-symmetric matrices, the exponential map series can be simplified to the *Rodrigues formula* [27, 25, 1], which is defined as

$$R(\boldsymbol{\theta}) = \exp([\mathbf{u}]_\times \theta) = I + [\mathbf{u}]_\times \sin(\theta) + [\mathbf{u}]^2_\times (1 - \cos(\theta)) \in \mathcal{SO}(3). \tag{2-5}$$

The exponential map can be inverted to give the logarithm, from $\mathcal{SO}(3)$ to $\mathfrak{so}(3)$, as

$$\boldsymbol{\theta}^\wedge = \log(R) = \frac{\theta}{2\sin(\theta)}(R - R^\top) \quad \text{with} \quad \theta = \cos^{-1}\Big(\frac{\mathrm{tr}(R) - 1}{2}\Big). \tag{2-6}$$

The identifying vector $\boldsymbol{\theta} \in \mathbb{R}^3$ is then formed by taking the off-diagonal elements of $\boldsymbol{\theta}^\wedge \in \mathfrak{so}(3)$.

2. Rotations in 2D space are represented by elements of the spatial rotation Lie group $\mathcal{SO}(2)$. Its properties are described as follows:

   - Similar to the case of 3D rotations, the Lie algebra $\mathfrak{so}(2)$ is the set of 2×2 skew-symmetric matrices. The single generator of $\mathfrak{so}(2)$ corresponds to the derivative of 2D rotation evaluated at the identity, which is given by

$$E_{\mathcal{SO}(2)} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

   An element of $\mathfrak{so}(2)$ is then any scalar multiple of the generator,

$$\theta^\wedge = [\theta]_\times = \theta E_{\mathcal{SO}(2)} \in \mathfrak{so}(2)$$

   where θ is the planar rotation angle.

- The exponential map that yields a planar rotation matrix by θ radians follows from Equation (2-2) as

$$R(\theta) = \exp([\theta]_\times) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \in \mathcal{SO}(2).$$

  The exponential map can inverted to yield the logarithmic map from $\mathcal{SO}(2)$ to $\mathfrak{so}(2)$ as

$$\theta = \text{Log}(R) = \tan^{-1}(R_{21}/R_{11}), \tag{2-7}$$

  where then $\theta^\wedge = \log(R) = \theta E_{\mathcal{SO}(2)}$.

Practical implementations of the exponential and logarithmic forms should use the Taylor series expansions of the coefficients where the numerator θ is small.

2-3-2 Transformations

Rigid motions are represented using rigid body transformations to describe the instantaneous orientation and position of a body coordinate frame relative to an inertial frame. That is, the position and orientation of a body-fixed coordinate frame B is described relative to an inertial frame A. For the 3-dimensional case, let $\mathbf{t}_\mathsf{B}^{(\mathsf{A})} \in \mathbb{R}^3$ be the position vector of the origin of frame B relative to frame A, and $R_\mathsf{B}^{(\mathsf{A})} \in \mathcal{SO}(3)$ the orientation of frame B relative to A. A configuration of frame B relative to A consists of the pair $(\mathbf{t}_\mathsf{B}^{(\mathsf{A})}, R_\mathsf{B}^{(\mathsf{A})})$ defined over the *Special Euclidean Group* $\mathcal{SE}(n)$, the (homogeneous) matrix form of which is defined as

$$\mathcal{SE}(n) = \left\{ T = \begin{bmatrix} R & \mathbf{t} \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)} : R \in \mathcal{SO}(n), \mathbf{t} \in \mathbb{R}^n \right\} \subset \mathcal{GL}(n).$$

Alternatively, the Special Euclidean group can be defined as a tuple of the translation and rotation:

$$\mathcal{SE}(n) = \{(\mathbf{t}, R) : \mathbf{p} \in \mathbb{R}^n, R \in \mathcal{SO}(n)\} = \mathbb{R}^n \times \mathcal{SO}(n). \tag{2-8}$$

Let $\mathbf{q}^{(\mathsf{A})}, \mathbf{q}^{(\mathsf{B})} \in \mathbb{R}^3$ be the coordinates of a point $\mathbf{q}$ relative to the frame A and B, respectively. The transformation of points and vectors by rigid transformations has a simple representation in terms of matrices and vectors in $\mathbb{R}^4$. In these *homogeneous coordinates*, the transformation $\mathbf{q}^{(\mathsf{A})} = T_\mathsf{B}^{(\mathsf{A})}(\mathbf{q}^{(\mathsf{B})})$ is an affine transformation that can be represented in the *linear* form

$$\underbrace{\begin{bmatrix} \mathbf{q}^{(\mathsf{A})} \\ 1 \end{bmatrix}}_{\bar{\mathbf{q}}^{(\mathsf{A})}} = \underbrace{\begin{bmatrix} R_\mathsf{B}^{(\mathsf{A})} & \mathbf{t}_\mathsf{B}^{(\mathsf{A})} \\ 0 & 1 \end{bmatrix}}_{\bar{T}_\mathsf{B}^{(\mathsf{A})}} \underbrace{\begin{bmatrix} \mathbf{q}^{(\mathsf{B})} \\ 1 \end{bmatrix}}_{\bar{\mathbf{q}}^{(\mathsf{B})}},$$

where $\bar{T}_\mathsf{B}^{(\mathsf{A})} \in \mathbb{R}^{4\times4}$ is the *homogeneous representation* of $T_\mathsf{B}^{(\mathsf{A})} \in \mathcal{SE}(3)$, and $\bar{\mathbf{q}}^{(\mathsf{A})}, \bar{\mathbf{q}}^{(\mathsf{B})} \in \mathbb{R}^4$ are the *homogeneous coordinates* of $\mathbf{q}^{(\mathsf{A})}, \mathbf{q}^{(\mathsf{B})} \in \mathbb{R}^3$.

As with rotations, rigid transformations can be defined for both the 3D and 2D case:

1. Rigid transformations in 3D space are represented by elements of the spatial rigid transformation Lie group $\mathcal{SE}(3)$. Such transformations are described using *screw theory*, which postulates that a rigid body can be moved from any one position to any other position by a movement consisting of rotation followed by translation parallel to the rotation axis, called a *screw motion* [25]. The differential equation

$$\dot{\mathbf{q}} = \boldsymbol{\omega} \times \mathbf{q}(t) + \mathbf{v} \tag{2-9}$$

describes the twist as the tip point trajectory of point $\mathbf{q}(t)$ due to rotation $\boldsymbol{\omega}$ through the origin with velocity $\mathbf{v}$ parallel to the axis of rotation, as seen from an inertial frame. Screw theory allows an an elegant, rigorous and geometric treatmetnt of spatial rigid body motion. For instance, a revolute joint through $\mathbf{p}$ can be modelled by $\dot{\mathbf{q}}(t) = \boldsymbol{\omega} \times (\mathbf{q}(t) - \mathbf{p})$ (where $\mathbf{v} = -\boldsymbol{\omega} \times \mathbf{q}$) whereas a prismatic joint can be modelled by $\mathbf{q}(t) = \mathbf{v}$.

The properties of the spatial rigid transformation group are described as follows:

- The structure of the Lie algebra $\mathfrak{se}(3)$ follows from Equation (2-9) in homogeneous coordinates. That is, $\mathfrak{se}(3)$ is the set of 4×4 matrices corresponding to the differential translations and rotations (i.e., $\mathbf{s} \in \mathbb{R}^3$ and $\boldsymbol{\theta} \in \mathbb{R}^3$, respectively) around the identity, as in $\mathfrak{so}(3)$. Thus, the six generators are described by

$$E^{(i)}_{\mathcal{SE}(3)} = \begin{bmatrix} 0 & \mathbf{e}_i \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad E^{(3+j)}_{\mathcal{SE}(3)} = \begin{bmatrix} E^{(j)}_{\mathcal{SO}(3)} & \mathbf{0} \\ 0 & 0 \end{bmatrix} \quad \text{for } i, j \in \{1, 2, 3\},$$

  where $\mathbf{e}_i$ are the Euclidean base vectors and $E^{(j)}_{\mathcal{SO}(3)}$ are the generators of $\mathcal{SO}(3)$. The Lie algebra is then the vector space whose elements are represented by multiples of the generators as

$$\boldsymbol{\xi}^\wedge = s_1 E^{(1)}_{\mathcal{SE}(3)} + s_2 E^{(2)}_{\mathcal{SE}(3)} + s_3 E^{(3)}_{\mathcal{SE}(3)} + \theta_1 E^{(4)}_{\mathcal{SE}(3)} + \theta_2 E^{(5)}_{\mathcal{SE}(3)} + \theta_3 E^{(6)}_{\mathcal{SE}(3)},$$

  where $\boldsymbol{\xi} = (\mathbf{s}, \boldsymbol{\theta}) = (s_1, s_2, s_3, \theta_1, \theta_2, \theta_3) = (\mathbf{v}, \boldsymbol{\omega})t \in \mathbb{R}^6 \cong \mathfrak{se}(3)$ is the identifying vector of integrated translations and rotations over duration t (assuming constant linear velocity $\mathbf{v}$ and angular velocity $\boldsymbol{\omega}$ over duration t).
- The closed-form expression of the exponential map that maps the Lie algebra element to the rigid transformation matrix follows from Equation (2-2) as

$$T(\boldsymbol{\xi}) = \exp(\boldsymbol{\xi}^\wedge) = \exp\Big(\begin{bmatrix} [\boldsymbol{\theta}]_\times & \mathbf{s} \\ 0 & 0 \end{bmatrix}\Big) = \begin{bmatrix} R(\boldsymbol{\theta}) & V(\boldsymbol{\theta})\mathbf{s} \\ 0 & 1 \end{bmatrix} \tag{2-10}$$
$$\text{with} \quad V(\boldsymbol{\theta}) = I + [\mathbf{u}]_\times(1 - \cos(\theta)) + [\mathbf{u}]^2_\times\Big(1 - \frac{\sin(\theta)}{\theta}\Big),$$

  where $R(\boldsymbol{\theta}) = \exp([\boldsymbol{\theta}]_\times)$ is determined using the Rodrigues formula of Equation (2-5). The logarithmic map is found by using Equation (2-6) to determine $\boldsymbol{\omega}^\wedge$ and computing $\mathbf{s} = V^{-1}\mathbf{t}$ with

$$V^{-1} = I - \frac{\theta}{2}[\mathbf{u}]_\times + \Big(1 - \frac{\theta \sin(\theta)}{2(1 - \cos(\theta))}\Big)[\mathbf{u}]^2_\times.$$

2. Rigid transformations in 2D space are represented by elements of the planar translation Lie group $\mathcal{SE}(2)$. Its properties are described as follows:

- The Lie algebra $\mathfrak{se}(2)$ is the set of 3×3 matrices corresponding to differential translations and rotation (i.e., $\mathbf{s} \in \mathbb{R}^2$ and $\theta \in \mathbb{R}$, respectively) around the identity. Thus, the three generators are

$$E^{(i)}_{\mathcal{SE}(2)} = \begin{bmatrix} 0 & \mathbf{e}_i \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad E^{(3)}_{\mathcal{SE}(2)} = \begin{bmatrix} E_{\mathcal{SO}(2)} & \mathbf{0} \\ 0 & 0 \end{bmatrix} \quad \text{for } i \in \{1, 2\},$$

  where $\mathbf{e}_i$ are two Euclidean base vectors and $E_{\mathcal{SO}(2)}$ is the generator of $\mathcal{SO}(2)$. The algebra is the vector space whose elements are represented by multiples of the generators as

$$\boldsymbol{\xi}^\wedge = s_1 E^{(2)}_{\mathcal{SE}(2)} + s_2 E^{(2)}_{\mathcal{SE}(2)} + \theta E^{(3)}_{\mathcal{SE}(2)}$$

  where $\boldsymbol{\xi} = (\mathbf{s}, \theta) = (s_1, s_2, \theta) \in \mathbb{R}^3 \cong \mathfrak{se}(2)$ is the vector of translations and rotation that identifies the Lie algebra.
- The closed-form expression of the exponential map is similar to Equation (2-10), and is given by

$$T(\boldsymbol{\xi}) = \exp(\boldsymbol{\xi}^\wedge) = \exp\Big(\begin{bmatrix} [\theta]_\times & \mathbf{s} \\ 0 & 0 \end{bmatrix} \Big) = \begin{bmatrix} R(\theta) & V(\theta)\mathbf{s} \\ 0 & 1 \end{bmatrix}$$

$$\text{with} \quad R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad \text{and} \quad V(\theta) = \frac{1}{\theta} \begin{bmatrix} \sin(\theta) & -(1-\cos(\theta)) \\ 1-\cos(\theta) & \sin(\theta) \end{bmatrix}.$$

  The exponential map can simply be inverted to yield the logarithmic map from $\mathcal{SE}(2)$ to $\mathfrak{se}(2)$, with $\log(R(\theta)) = \theta$ per Equation (2-7) and $\mathbf{s} = V^{-1}\mathbf{t}$.

2-4 Kinematics

With the tools in place to describe rigid body motion, the velocity is left to be described. Again, the notion of twists is utilised to derive a proper representation of rigid body velocity. In this section, only the 3-dimensional case is considered.

2-4-1 Rotational velocity

Let $R^{(\mathsf{A})}_{(\mathsf{B})}(t) \in \mathcal{SO}(3)$ be a curve representing the trajectory of an object frame B, with origin at the origin of frame A, and rotating relative to fixed A. The velocity of any point $\mathbf{q}_\mathsf{B}$ fixed to B as seen from A is given by

$$\dot{\mathbf{q}}^{(\mathsf{A})}_{(\mathsf{B})}(t) = \frac{\mathrm{d}}{\mathrm{d}t}(R^{(\mathsf{A})}_{(\mathsf{B})}(t)\mathbf{q}^{(\mathsf{B})}_{(\mathsf{B})}) = \dot{R}^{(\mathsf{A})}_{(\mathsf{B})}(t)\mathbf{q}^{(\mathsf{B})}_{(\mathsf{B})}$$

where $\dot{R}^{(\mathsf{A})}_{(\mathsf{B})}(t)$ maps the body coordinates of a point to the spatial velocity of that point. The *instantaneous spatial angular velocity* $\boldsymbol{\omega}^{(\mathsf{A})}_{(\mathsf{B})} \in \mathbb{R}^3$ and *instantaneous body angular velocity* $\boldsymbol{\omega}^{(\mathsf{B})}_{(\mathsf{B})} \in \mathbb{R}^3$ are defined as

$$\left.\begin{aligned} (\boldsymbol{\omega}^\wedge)^{(\mathsf{A})}_{(\mathsf{B})} &:= \dot{R}^{(\mathsf{A})}_{(\mathsf{B})}(R^{(\mathsf{A})}_{(\mathsf{B})})^{-1} \\ (\boldsymbol{\omega}^\wedge)^{(\mathsf{B})}_{(\mathsf{B})} &:= (R^{(\mathsf{A})}_{(\mathsf{B})})^{-1}\dot{R}^{(\mathsf{A})}_{(\mathsf{B})} \end{aligned}\right\} \quad \Longrightarrow \quad (\boldsymbol{\omega}^\wedge)^{(\mathsf{B})}_{(\mathsf{B})} = (R^{(\mathsf{A})}_{(\mathsf{B})})^{-1}(\boldsymbol{\omega}^\wedge)^{(\mathsf{A})}_{(\mathsf{B})}R^{(\mathsf{A})}_{(\mathsf{B})}.$$

Both are related via $\boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{B})} = \left(R_{\mathsf{B}}^{(\mathsf{A})}\right)^{-1}\boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{A})}$. With these definitions, the velocity of $\mathbf{q}_{\mathsf{B}}$ can be expressed as

$$\begin{aligned}\dot{\mathbf{q}}_{\mathsf{B}}^{(\mathsf{A})}(t) &= \boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{A})}(t) \times \mathbf{q}_{\mathsf{B}}^{(\mathsf{A})}(t),\\ \dot{\mathbf{q}}_{\mathsf{B}}^{(\mathsf{B})}(t) &= \boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{B})}(t) \times \mathbf{q}_{\mathsf{B}}^{(\mathsf{B})}.\end{aligned}$$

2-4-2 Rigid body velocity

Let $T_{\mathsf{B}}^{(\mathsf{A})}(t) \in \mathcal{SE}(3)$ be the time-parameterised curve representing the trajectory of a rigid body; more specifically, the rigid body motion of body-fixed frame B relative to fixed inertial frame A. The velocity of any point $\mathbf{q}_{\mathsf{B}}$ fixed to B as seen from A is given by

$$\dot{\mathbf{q}}_{\mathsf{B}}^{(\mathsf{A})} = \frac{\mathrm{d}}{\mathrm{d}t}(T_{\mathsf{B}}^{(\mathsf{A})}\mathbf{q}_{\mathsf{B}}^{(\mathsf{B})}) = \dot{T}_{\mathsf{B}}^{(\mathsf{A})}\mathbf{q}_{\mathsf{B}}^{(\mathsf{B})} = \dot{T}_{\mathsf{B}}^{(\mathsf{A})}\left(T_{\mathsf{B}}^{(\mathsf{A})}\right)^{-1}\mathbf{q}_{\mathsf{B}}^{(\mathsf{A})}.$$

By analogy of the rotational velocity, the *instantaneous spatial velocity* $\mathbf{V}_{\mathsf{B}}^{(\mathsf{A})} \in \mathbb{R}^6$ and *instantaneous body velocity* $\mathbf{V}_{\mathsf{B}}^{(\mathsf{B})} \in \mathbb{R}^6$ are defined as

$$(\mathbf{V}^\wedge)_{\mathsf{B}}^{(\mathsf{A})} := \dot{T}_{\mathsf{B}}^{(\mathsf{A})}T_{\mathsf{A}}^{(\mathsf{B})} = \begin{bmatrix}\dot{R}_{\mathsf{B}}^{(\mathsf{A})}R_{\mathsf{A}}^{(\mathsf{B})} & -\dot{R}_{\mathsf{B}}^{(\mathsf{A})}R_{\mathsf{A}}^{(\mathsf{B})}\mathbf{t}_{\mathsf{B}}^{(\mathsf{A})} + \dot{\mathbf{t}}_{\mathsf{B}}^{(\mathsf{A})}\\ 0 & 0\end{bmatrix} = \begin{bmatrix}(\boldsymbol{\omega}^\wedge)_{\mathsf{B}}^{(\mathsf{A})} & \mathbf{v}_{\mathsf{B}}^{(\mathsf{A})}\\ 0 & 0\end{bmatrix} \quad\Longrightarrow\quad \mathbf{V}_{\mathsf{B}}^{(\mathsf{A})} = \begin{bmatrix}\mathbf{v}_{\mathsf{B}}^{(\mathsf{A})}\\ \boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{A})}\end{bmatrix},$$

$$(\mathbf{V}^\wedge)_{\mathsf{B}}^{(\mathsf{B})} := T_{\mathsf{B}}^{(\mathsf{A})}\dot{T}_{\mathsf{A}}^{(\mathsf{B})} = \begin{bmatrix}R_{\mathsf{A}}^{(\mathsf{B})}\dot{R}_{\mathsf{B}}^{(\mathsf{A})} & R_{\mathsf{A}}^{(\mathsf{B})}\dot{\mathbf{t}}_{\mathsf{B}}^{(\mathsf{A})}\\ 0 & 0\end{bmatrix} = \begin{bmatrix}(\boldsymbol{\omega}^\wedge)_{\mathsf{B}}^{(\mathsf{B})} & \mathbf{v}_{\mathsf{B}}^{(\mathsf{B})}\\ 0 & 0\end{bmatrix} \quad\Longrightarrow\quad \mathbf{V}_{\mathsf{B}}^{(\mathsf{B})} = \begin{bmatrix}\mathbf{v}_{\mathsf{B}}^{(\mathsf{B})}\\ \boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{B})}\end{bmatrix}.$$

In either case, the velocity is written as,

$$\begin{aligned}\dot{\mathbf{q}}_{\mathsf{B}}^{(\mathsf{A})} &= (\mathbf{V}^\wedge)_{\mathsf{B}}^{(\mathsf{A})}\mathbf{q}_{\mathsf{B}}^{(\mathsf{A})} = \boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{A})} \times \mathbf{q}_{\mathsf{B}}^{(\mathsf{A})} + \mathbf{v}_{\mathsf{B}}^{(\mathsf{A})},\\ \dot{\mathbf{q}}_{\mathsf{B}}^{(\mathsf{B})} &= (\mathbf{V}^\wedge)_{\mathsf{B}}^{(\mathsf{B})}\mathbf{q}_{\mathsf{B}}^{(\mathsf{B})} = \boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{B})} \times \mathbf{q}_{\mathsf{B}}^{(\mathsf{B})} + \mathbf{v}_{\mathsf{B}}^{(\mathsf{B})}.\end{aligned}$$

Intuitively, the angular component $\boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{A})}$ is the instantaneous angular velocity of the body as viewed from the spatial frame A, and $\mathbf{v}_{\mathsf{B}}^{(\mathsf{A})}$ is the velocity of the point on the body-fixed frame B travelling through the origin of A. Furthermore, $\mathbf{v}_{\mathsf{B}}^{(\mathsf{B})}$ can be interpreted as the velocity of the origin of the body coordinate frame B relative to the spatial frame A, and $\boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{B})}$ the angular velocity of B viewed in the same frame.

The instantaneous velocities are related via the similarity transform

$$(\mathbf{V}^\wedge)_{\mathsf{B}}^{(\mathsf{A})} = T_{\mathsf{B}}^{(\mathsf{A})}(\mathbf{V}^\wedge)_{\mathsf{B}}^{(\mathsf{B})}\left(T_{\mathsf{B}}^{(\mathsf{A})}\right)^{-1}.$$

Alternatively,

$$\begin{aligned}\boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{A})} &= R_{\mathsf{B}}^{(\mathsf{A})}\boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{B})}\\ \mathbf{v}_{\mathsf{B}}^{(\mathsf{A})} &= \underbrace{\mathbf{t}_{\mathsf{B}}^{(\mathsf{A})} \times \left(R_{\mathsf{B}}^{(\mathsf{A})}\boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{B})}\right)}_{-\boldsymbol{\omega}_{\mathsf{B}}^{(\mathsf{A})}\times\mathbf{t}_{\mathsf{B}}^{(\mathsf{A})}} + \underbrace{R_{\mathsf{B}}^{(\mathsf{A})}\mathbf{v}_{\mathsf{B}}^{(\mathsf{A})}}_{\dot{\mathbf{t}}_{\mathsf{B}}^{(\mathsf{A})}}\end{aligned} \quad\Longrightarrow\quad \mathbf{V}_{\mathsf{B}}^{(\mathsf{A})} = \begin{bmatrix}R_{\mathsf{B}}^{(\mathsf{A})} & (\mathbf{t}^\wedge)_{\mathsf{B}}^{(\mathsf{A})}R_{\mathsf{B}}^{(\mathsf{A})}\\ 0 & R_{\mathsf{B}}^{(\mathsf{A})}\end{bmatrix}\mathbf{V}_{\mathsf{B}}^{(\mathsf{B})}. \quad (2\text{-}11)$$

Thus, the spatial and body velocity of a rigid motion are related via the *adjoint transformation* associated with T, $\mathrm{Ad}_T : \mathbb{R}^6 \to \mathbb{R}^6$, given as

$$\mathrm{Ad}_T = \begin{bmatrix}R & (\mathbf{t}^\wedge)R\\ 0 & R\end{bmatrix} \quad\text{and}\quad \mathrm{Ad}_T^{-1} = \begin{bmatrix}R^\top & -(R^\top\mathbf{t})^\wedge R^\top\\ 0 & R^\top\end{bmatrix} = \begin{bmatrix}R^\top & -R^\top(\mathbf{t}^\wedge)\\ 0 & R^\top\end{bmatrix} = \mathrm{Ad}_{T^{-1}}.$$

Chapter 3

SLAM framework

As discussed in Chapter 1, the SLAM problem asks whether it is possible for the robot to incrementally build a consistent map of an unknown environment while simultaneously determining its location within this map. Modern solutions to the SLAM problem are based on the *graph*-based formulation of the robot trajectory and environment, where the optimum is found as the set of robot poses and/or landmarks that minimises the total deviation from the constraints over all relations between them.

A SLAM framework essentially processes raw sensor data to create a graph that can be optimised to yield the SLAM solution. The architecture of a SLAM system includes two main components:

1. The *front-end*, which abstracts sensor data into models that are amenable for estimation.
2. The *back-end*, which performs inference on the abstracted data produced by the front-end.

Basically, the front-end constructs the graph from raw sensor data, and the back-end optimises the graph to find the SLAM solution.

Section 3-1 gives an overview of the different SLAM methodologies. Section 3-2 introduces RTAB-Map, which is the SLAM framework of choice for this research. RTAB-Map implements the front-end of the SLAM system, and relegates the back-end to a set of off-the-shelf optimisation frameworks. These frameworks are briefly discussed in Section 3-3

3-1 Methodologies

Two main SLAM methodologies exist, each of which yields a different optimisation problem [10]. The difference in these approaches stems from the chosen metric representation (or metric map) that encodes the geometry of the environment.

- In *point-feature SLAM* (or feature-based SLAM), the environment is represented by a set of sparse features (or landmarks) corresponding to discriminative features in the environment (e.g. lines or corners). The assumption underlying this representation is that the

landmarks are distinguishable.

- In *pose SLAM*, only the poses are estimated and the positions of non-pose features can be trivially computed given the robot trajectory since they become conditionally independent [29]. It is assumed that, when having a good estimation of the robot pose, the map can be retrieved by simply referring the relative measurements which can come from a large variety of sparse or dense sensors. Landmarks are then used only to derive relative measurements linking pairs of poses.

Feature-based SLAM was the first approach to be implemented and therefore quite mature. However, with the advent of more information-dense sensors (such as LiDAR and RGB-D), the amount of landmarks increased beyond the point of practicality. As such, more recently, the more compact representation of pose SLAM is found to have more active development.

3-2 RTAB-Map

RTAB-Map (**R**eal-**T**ime **A**ppearance-**B**ased **Map**ping) [30] is a graph-based SLAM approach based on an incremental appearance-based loop-closure detector. Appearance-based refers to its use of either RGB-D, stereo-imagery, or LiDAR, for its loop-closure detection. It is an open-source library implemented as a ROS package. A memory management approach is used to limit the number of locations used for the loop-closure detection and graph optimisation, such that real-time constraints on large-scale environments are always respected.

RTAB-Map is the framework of choice for this research, because it is versatile, offers support for a large variety of sensors, and, most importantly, is open-source.

3-2-1 Sensors

RTAB-Map supports a variety of sensors. a subset of which can be utilised during operation. The data generated by these sensors are published to specified topics on ROS, allowing RTAB-Map to interpret this data for mapping purposes. The `rtabmap_ros` node subscribes to the following topics:

- *RGB-D images*: colour-depth images combine a colour image (where RGB refers to the colour, encoded by red-green-blue colour model) with a depth image (where D refers to the depth data). The depth data is generally calculated from the distortion of a known infrared dot pattern, as viewed by an offset infrared camera. By default, the colour-depth images contain:
  - A rectified (i.e., projected onto a common image plane) monochromatic or colour image, encoded in the `sensor_msgs/Image` message, published to the `rgb/image` topic;
  - A depth image, encoded in the `sensor_msgs/Image` message, published to the `depth/image` topic;
  - Camera metadata, encoded in the `sensor_msgs/CameraInfo` message, published to the `rgb/camera_info` topic.

  If a single synchronised RGB-D is preferred, this data format contains:
  - A color-depth images, encoded in the `rtabmap_ros/RGBDImage` message, published to the `rgbd_image` topic.

- *Stereo images*: a stereoscopic camera system captures two images from slightly different perspectives to simulate human binocular vision. Stereo photography is used to derive depth information from the differences in these images. The stereoscopic images contain:
  - A left rectified monochromatic or colour image, encoded in the `sensor_msgs/Image` message, published to the `left/image_rect` topic;
  - Left camera metadata, encoded in the `sensor_msgs/CameraInfo` message, published to the `left/camera_info` topic;
  - A right rectified monochromatic or colour image, encoded in the `sensor_msgs/Image` message, published to the `right/image_rect` topic;
  - Right camera metadata, encoded in the `sensor_msgs/CameraInfo` message, published to the `right/camera_info` topic.
- *2D laser scan*: planar laser range-finders are used to generate a 2D depth array. These scans contain:
  - An array of range data for each of the angle increments, encoded in a `sensor_msgs/LaserScan` message, published to the `scan` topic.
- *3D point cloud*: spatial laser range-finders are used to generate a 3D point-cloud. These scans contain:
  - A point-cloud, encoded in a `sensor_msgs/PointCloud2` message, published to the `scan_cloud` topic.
- *External odometry*: this refers to the odometry input that is calculated outside of RTAB-Map. Since it is external, any kind of odometry can be used in harmony with RTAB-Map, depending on what is appropriate for a given application and robot. The odometry data contains:
  - A pose (position and orientation in 3D with respect to specified coordinate frames) and twist (linear and angular velocity in 3D), each with an accompanying covariance matrix, encoded in a `nav_msgs/Odometry` message, published to the `odom` topic.

RTAB-Map requires at least a stream of RGB-D images, or a stream of stereo images with external odometry. This is because RTAB-Map performs loop-closures based on RGB images.

3-2-2 Odometry

RTAB-Map offers implementations for the estimation of odometry from visual sources (i.e., RGB-D and stereo imagery) and LiDAR. These approaches are independent of the mapping process, and can therefore be replaced by any other odometry approaches, such as e.g. wheel odometry, or any other visual or LiDAR-based odometry package (like A-LOAM [31]).

- *Visual odometry*: this can be used with either RGB-D images or stereo images, processed by the `rgbd_odometry` and `stereo_odometry` nodes, respectively. The process works as follows [30]:
  1. Feature detection: when a new frame is captured, features are extracted (via e.g. GoodFeaturesToTrack).
  2. Motion prediction: a motion model predicts where the features should be in the current frame, based on the previous motion transformations. This limits the search window for feature matching to provide better matches.

3. Feature matching: features of the current frame are matched with previous observations (using e.g. optical flow).
4. Motion estimation: feature correspondences are used to compute the transformation of the current frame according to previous observations.
5. Local bundle adjustment: the resulting transformation is refined using bundle adjustment.
6. Pose update: the output odometry is updated and the corresponding estimation covariance is computed.
7. Key-frame and feature-map update: the set of past observations is updated, based on the number of inliers computed during motion estimation.

RTAB-Map implements two standard odometry approaches:

- Frame-To-Frame (F2F), which registers the new frame against the last key-frame;
- Frame-To-Map (F2M), which registers the new frame against a local map of features created from past key-frames.

- *LiDAR-based odometry*: both planar (2D) and spatial (3D) LiDAR scans can be used to estimate odometry. The process works as follows:
  1. Point-cloud filtering: the newly captured point-cloud is down-sampled.
  2. Motion prediction: either a motion model is used to predict the current transform, or an external odometry source is consulted.
  3. ICP registration: iterative-closest-point (ICP) is used to register the new point-cloud with respect to previous observations.
  4. Pose update: the output odometry is updated and the corresponding estimation covariance is computed.
  5. Key-frame and point-cloud map update: the set of past observations is updated, based on the number of correspondences.

  Similarly to the two implement odometry approaches for visual odometry, RTAB-Map implements two approaches for LiDAR-based odometry:

  - Scan-To-Scan (S2S), which registers the new point-cloud against the last point-cloud;
  - Scan-To-Map (S2M), which registers the new point-cloud against a local map of features created from the past point-clouds.

Even though RTAB-Map only directly supports one source of odometry, multiple sources (including RTAB-Map's own visual and LiDAR odometry) can be fused together prior to being fed into RTAB-Map (using e.g. `robot_localization` [32]).

3-2-3 Graph construction

A graph consists of nodes and edges. The nodes represent reference locations in space and time (typically robot poses), and the edges represent the geometric relations between nodes in the form of a soft constraint derived from observations. Nodes are created at a fixed rate (`Rtabmap/DetectionRate` in milliseconds) according to how much the data from successive nodes should overlap. The relations are rigid transformations that are of three kinds:

- *Neighbour* edges are created between consecutive nodes with odometry transformation.

- *Loop-closure* edges are added between non-consecutive nodes through loop-closure detections.
- *Proximity* edges are added between nearby subsequent nodes (e.g., between nodes that are at least 4 neighbouring edges apart) through proximity detections.

When a new loop-closure or proximity edge is added to the graph, graph optimisation propagates the computed error to the graph, thereby increasing the online odometry drift.

3-2-4 Memory management

RTAB-Map's memory is divided into a Working Memory (WM) and a Long-Term Memory (LTM). When a node is transferred to LTM, it is not available anymore for modules inside the WM. When the update time exceeds a fixed time threshold (`Rtabmap/TimeThr` in milliseconds) or the memory size exceeds a memory threshold (`Rtabmap/MemoryThr`), some nodes in the WM are transferred to LTM to limit the size of the WM and decrease the update time.

3-2-5 Loop-closure and proximity detection

Loop-closure detection is done using the bag-of-words approach [33]. This creates an image signature that is represented by a set of visual words. This signature is contained in a visual vocabulary that is incrementally constructed online (as opposed to a pre-trained vocabulary, which requires an extra training step).

When using RTAB-Map's visual odometry, features extracted for odometry can be re-used for loop-closure detection. This eliminates the double extraction of features and reduces computational load.

3-2-6 Global map assembly

The global map can be assembled by RTAB-Map in the following formats:

- *Octomap*, which is encoded in the `octomap_msgs/Octomap` message. It models the surroundings in a binary format in an octree, which is a tree data structure in which each internal node has exactly eight children. Colour information can be added to the occupied voxels to better reflect reality.
- *Occupancy grid*, which is encoded in the `nav_msgs/OccupancyGrid` message. With an occupancy grid, the surroundings are only modelled in two dimensions. A grid represents the a horizontal plane of the surroundings, and each cell represents the probability of occupancy.
- *Point-cloud*, which is encoded in the `sensor_messages/PointCloud2` message. It derives a representation of the surroundings by overlaying the point-clouds generated by the LiDAR scanners.

3-3 Back-end

The back-end contains the optimiser that solves the non-linear least-squares optimisation problem expressed by the graph. The back-end relies heavily on the topological correctness of the graph structure and is not robust against misplaced constraint edges. Therefore, edges that represent false positive loop-closures could lead to divergence of non-robust solvers.

The optimisation algorithm that most commonly used in graph optimisation are discussed below. Each of these algorithm locally linearises the cost function and performs inference based on the direction of the gradient or by exploiting the sparsity of the linearised matrix form.

- *Gradient methods*:
  - Olson's algorithm [34]
  - TORO (**T**ree-based netw**OR**k **O**ptimiser) [35]
- *Sparse matrix factorisation*:
  - $\sqrt{\text{SAM}}$ (**square-root** **S**moothing **A**nd **M**apping) [18]
  - iSAM (**i**ncremental **S**moothing **A**nd **M**apping) [19]
  - iSAM2 (**i**ncremental **S**moothing **A**nd **M**apping **2**) [20]
  - g2o (**g**eneral (hyper)**g**raph **o**ptimisation) [36]

Chapter 4

SLAM problem formulation

The goal of SLAM is to infer an unknown map, while simultaneously localising a robot with respect to said evolving map, using information coming from its sensors. A *smoothing* approach to SLAM [18, 19, 20] involves estimating not just the most current robot location, but the entire robot trajectory up to the current time. Smoothing approaches have been shown to be very fast alternatives to the filtering approaches based on an interpretation of factorisation in terms of the graphical model associated with the SLAM problem [18].

In this chapter, the SLAM problem is derived as a non-linear least-squares optimisation problem. Although only pose SLAM is considered in this work, the notions derived in this chapter are also valid for feature-based SLAM, once a valid error function is defined.

4-1 Preliminaries

Before the SLAM problem description can be formally derived, some preliminaries need to be established.

4-1-1 Reference frames

SLAM is a geometrical optimisation problem. To formulate such a problem, a set of reference frames is defined, relative to which these geometrical constraints can be defined. The Special Euclidean group $\mathcal{SE}(n)$ is used to formulate relative rigid body transformations. Such a transformation consist of the instantaneous orientation $R \in \mathcal{SO}(n)$ and position $\mathbf{t} \in \mathbb{R}^n$ of one reference frame in terms of another, which is highlighted by the definition of $\mathcal{SE}(n)$ of Equation (2-8).

The reference frames that are necessary for the formulation of the SLAM problem are the following [37]:

1. The *fixed global* (or *inertial*) *reference frame* O (where 'O' refers to the 'global origin') that is fixed at the robot starting position.

2. The *robot-fixed reference frame* R that is fixed to the robot's *odometric centre* (i.e., at the centre of rotation, along the axis of forward motion). Due to the movement of the robot, the position and orientation of the robot-fixed frame changes over time, with the respect to the inertial frame. Accordingly, the transformation of the frame R with respect to O, denoted by $T^{(\mathsf{O})}_{\mathsf{R}_t} := T^{(\mathsf{O})}_{\mathsf{R}}(t)$ is time-dependent. A subscript t can be added to the frame symbol to the highlight this time-dependence and indicate which version of the robot-fixed frame is referred to.

3. The *sensor-fixed reference frame* S^{id} that is fixed at the centre of measurement of sensor with identifier 'id'. The sensor attachment is assumed to be rigid and, therefore, the relative sensor frame $\ell_{\mathrm{id}} := T^{(\mathsf{R})}_{\mathsf{S}^{\mathrm{id}}}$ is considered a constant. On the other hand, the transformation $T^{(\mathsf{O})}_{\mathsf{S}^{\mathrm{id}}_t} := T^{(\mathsf{O})}_{\mathsf{S}_{\mathrm{id}}}(t)$ is a function of time.

4-1-2 Sets of interest

A robot performing SLAM uses its sensory data to estimate its own position within a self-constructed map of the environment. The relevant data and parameters can be classified as follows:

- *Robot poses*: the robot moves through the environment via a sequence of poses. At time instant T, the robot has moved through the sequence $\mathcal{X}_{0:T} = \{x_0, \ldots, x_T\} = \mathcal{X}$, where $\mathcal{X}$ denotes the set of all poses. Each pose $x_t = T^{(\mathsf{O})}_{\mathsf{R}_t} = (\mathbf{t}_t, R_t) \in \mathcal{SE}(n)$ denotes the robot's pose as seen in the global reference frame O at time t.

- *Measurements*: the robot perceives its environment through various sensor systems. The front-end of the SLAM system processes the raw sensor data and constructs a graph of constraints. These constraints consist of a *constraint value* and *constraint covariance*. The constraint value is determined from a sensor model as the geometric relation that perfectly matches the raw observation. The constraint covariance encodes the confidence that the front-end has in its estimation of the constraint value. Both parameters fully define a graph constraint.

  Each of the constraints generated by the front-end defines a relative transformation $z_{i,j}$ between poses x_i and x_j such that $x_i z_{i,j} = x_j$, with $x_i, x_j, z_{i,j} \in \mathcal{SE}(n)$. This transformation acts as the constraint value. The constraint covariance is generally set a priori and specifically for each sensor system.

  The set of all measurements is denoted by $\mathcal{Z}$.

4-2 Solution

In order to solve the full SLAM problem [22]), the knowledge about the poses $\mathcal{X}$ is characterised with respect to the measurements $\mathcal{Z}$. This quantity describes the belief over $\mathcal{X}$ given the measurements $\mathcal{Z}$ and is equivalent to the posterior probability $p(\mathcal{X} \mid \mathcal{Z})$.

Maximum a posteriori (MAP) estimation simply maximises the posterior probability

$$p(\mathcal{X} \mid \mathcal{Z}) = \frac{p(\mathcal{X}, \mathcal{Z})}{p(\mathcal{Z})} = \frac{p(\mathcal{Z} \mid \mathcal{X})}{p(\mathcal{Z})} p(\mathcal{X}) \propto \mathcal{L}(\mathcal{X}; \mathcal{Z}) p(\mathcal{X}),$$

where $\mathcal{L}(\mathcal{X}; \mathcal{Z})$ denotes the likelihood of $\mathcal{X}$ given $\mathcal{Z}$ (which is defined as any function proportional to $p(\mathcal{Z} \mid \mathcal{X})$) and $p(\mathcal{X})$ is the prior probability over $\mathcal{X}$ (which includes any prior knowledge about $\mathcal{X}$). The normalising constant $1/p(\mathcal{Z})$ is dropped, since it only ensures the posterior probability integrates to one, and therefore does not change the maximum point.

The solution $\mathcal{X}^*$ to the full SLAM problem is the set $\mathcal{X}$ that maximises the posterior probability:

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} \mathcal{L}(\mathcal{X}; \mathcal{Z}) p(\mathcal{X}).$$

When no prior knowledge is available for $\mathcal{X}$, $p(\mathcal{X})$ becomes constant (i.e., a uniform distribution) that is inconsequential for optimisation, and can thus be dropped. Then,

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} \mathcal{L}(\mathcal{X}; \mathcal{Z}). \tag{4-1}$$

4-3 Constraint models

Measurements are subject to uncertainties. These uncertainties can be modelled via a probabilistic description based on the Bayesian interpretation of probability. In many cases, it is both justified and convenient to model measurements as corrupted by additive zero-mean Gaussian noise.

The uncertain measurement $z_{i,j}$ between poses x_i and x_j is modelled as

$$z_{i,j} \oplus \epsilon = f_{i,j}(\mathcal{X}),$$

where $f_{i,j}$ is the unbiased estimator of the measurement $z_{i,j}$ and $\epsilon \sim \mathcal{N}(0, \Sigma_{i,j})$ models the (assumed) zero-mean Gaussian process noise with covariance $\Sigma_{i,j}$ for the constraint between pose x_i and x_j.

The measurement $z_{i,j} := T_{\mathsf{R}_j}^{(\mathsf{R}_i)}$ is the transformation of frame R_i to frame R_j. The identity $x_i z_{i,j} = x_j$ states that the transformation is 'added' to the pose $x_i = T_{\mathsf{R}_i}^{(\mathsf{O})}$ by right-multiplication to yield the combined pose $x_j = T_{\mathsf{R}_j}^{(\mathsf{O})}$ [11]. That is,

$$x_j = T_{\mathsf{R}_j}^{(\mathsf{O})} = \begin{cases} T_{\mathsf{R}_i}^{(\mathsf{O})} T_{\mathsf{R}_j}^{(\mathsf{R}_i)} = x_i T_{\mathsf{R}_j}^{(\mathsf{R}_i)} \\ T_{\mathsf{R}_i}^{(\mathsf{O})} \left(T_{\mathsf{R}_i}^{(\mathsf{O})}\right)^{-1} T_{\mathsf{R}_j}^{(\mathsf{O})} = x_i \left(T_{\mathsf{R}_i}^{(\mathsf{O})}\right)^{-1} T_{\mathsf{R}_j}^{(\mathsf{O})} \end{cases} \quad \Longrightarrow \quad T_{\mathsf{R}_j}^{(\mathsf{R}_i)} = \left(T_{\mathsf{R}_i}^{(\mathsf{O})}\right)^{-1} T_{\mathsf{R}_j}^{(\mathsf{O})}.$$

With the definitions

$$T^{-1} = \begin{bmatrix} R^\top & -R^\top \mathbf{t} \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad T_1 T_2 = \begin{bmatrix} R_1 R_2 & R_1 \mathbf{t}_2 + \mathbf{t}_1 \\ 0 & 1 \end{bmatrix},$$

the closed form of the relative transformation from x_i to x_j is derived as

$$f_{i,j}(\mathcal{X}) = \begin{bmatrix} R_i^\top R_j & R_i^\top \mathbf{t}_j - R_i^\top \mathbf{t}_i \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_i^\top R_j & R_i^\top (\mathbf{t}_j - \mathbf{t}_i) \\ 0 & 1 \end{bmatrix} \in \mathcal{SE}(n).$$

This measurement model suffices for all constraint present in a pose SLAM problem (i.e., odometry constraint, proximity constraints, and loop-closure constraints).

Its conditional density that describes the probability of measurement $z_{i,j}$ given the poses x_i and x_j is given by

$$p(z_{i,j} \mid x_i, x_j) = \frac{1}{\sqrt{(2\pi)^{n_{\mathcal{SE}(n)}} |\Sigma_{i,j}|}} \exp\Big(-\frac{1}{2} \| f_{i,j}(\mathcal{X}) \ominus z_{i,j} \|^2_{\Sigma_{i,j}} \Big), \tag{4-2}$$

where $\| f_{i,j}(\mathcal{X}) \ominus z_{i,j} \|^2_{\Sigma_{i,j}} := (f_{i,j}(\mathcal{X} \ominus z_{i,j})^\top \Sigma_{i,j}^{-1} (f_{i,j}(\mathcal{X} \ominus z_{i,j})$ denotes the squared Mahalanobis distance.

Due to the use of the minus operator, which is defined in Equation (2-4), the argument of the Mahalanobis distance operator defines a vector error function that computes the difference between the expected observation $f_{i,j}(\mathcal{X})$ and the real observation $z_{i,j}$:

$$\mathbf{e}_{i,j}(\mathcal{X}) = f_{i,j}(\mathcal{X}) \ominus z_{i,j}. \tag{4-3}$$

This error function plays a key role in the formulation of the SLAM problem, as elaborated upon in the subsequent sections.

4-4 Graphical models

SLAM is solved by exploiting probabilistic graphical models, which provide a mechanism for compactly describing complex probability densities. All of the graphical models discussed in this section can be converted from one form into another and differ only in their description of the problem. Some problem formulations match up better with the structure of one model compared to another.

4-4-1 Bayesian network

A Bayesian network (or Bayes net) is a probabilistic graphical model that represents a set of random variables $\mathcal{U}$ and their conditional dependencies via a directed acyclic graph [11]. That is, a Bayes net $F = (\mathcal{U}, \mathcal{E})$ defines the joint probability density $p(\mathcal{U})$ over the set of nodes $\mathcal{U} = \{u_1, \ldots, u_n\}$ as the product of conditional densities associated with each of the nodes:

$$p(\mathcal{U}) = \prod_i p(u_i \mid \text{parent}(u_i)),$$

where $\text{parent}(u_i) \subset \mathcal{U}$ denotes the set of parent nodes to u_i. As such, the factorisation of the joint density is dictated by its graph structure (i.e., the node-parent relationships).

When modelling a SLAM problem, the poses $\mathcal{X}$ and measurements $\mathcal{Z}$ are all represented by nodes in the Bayes net. Edges are directed from poses to the relevant measurements (derived from odometry, loop-closures, and proximity matches).

Example: Consider the Bayes net example shown in Figure 4-1, with robot poses $\mathcal{X}_{0:4}$ and corresponding observations. The joint density $p(\mathcal{X}, \mathcal{Z})$ is obtained as the product of conditional densities:

$$\begin{aligned} p(\mathcal{X}, \mathcal{Z}) = p(z_{0,1} \mid x_0, x_1)p(z_{1,2} \mid x_1, x_2)p(z_{2,3} \mid x_2, x_3)p(z_{3,4} \mid x_3, x_4) & \quad \text{Odometry measurements} \\ \times p(z_{1,3} \mid x_1, x_3) & \quad \text{Proximity measurements} \\ \times p(z_{0,4} \mid x_0, x_4)p(z_{1,4} \mid x_1, x_4) & \quad \text{Loop-closure measurements} \end{aligned}$$

where the prior factors on $\mathcal{X}$ are dropped, as these become constant without prior knowledge.

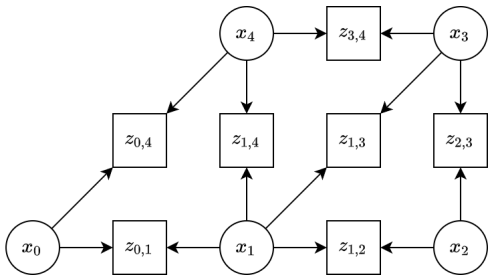

Figure 4-1: Bayesian network example. Not only are the robot poses $x_0, \ldots, x_4$ represented by graph nodes, so are the odometry measurements $z_{0,1}, z_{1,2}, z_{2,3}, z_{3,4}$, proximity measurement $z_{1,3}$ and loop-closure measurements $z_{0,4}, z_{1,4}$. The edges represent the parent-child relationship of measurements to poses.

Although a Bayes net is an ideal generative modelling framework (simulation is simply done by ancestral sampling), its modelling flow is mismatched with the actual MAP estimation problem of SLAM. This is because of the following:

- Bayesian networks make no distinction between poses and measurements, as both are represented as graph nodes. In pose SLAM, poses are considered unknown variables, whereas measurements are considered known constants. As such, measurements can be considered as parameters of the joint probability factors over the *actual* unknowns.
- Likelihood functions are non-Gaussian, and therefore not proper probability densities. This is due to the often lower dimensionality of the measurement $\mathcal{Z}$ than the unknown variables $\mathcal{V}$ it depends on, which results in the accordance of the same likelihood to an infinite subset of the domain of unknowns (i.e., *perceptual aliasing*).

The *factor graph* is better suited for inference, as it addresses both issues.

4-4-2 Factor graph

A factor graph is a bipartite graph $G = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ with two types of nodes:

- Factors $\phi_i(\mathcal{V}_i) \in \mathcal{U}$, where $\mathcal{V}_i \subseteq \mathcal{V}$;
- Variables $v_j \in \mathcal{V}$.

Like Bayes nets, factor graphs allow for the specification of a joint density over $\mathcal{V}$ as a product of factors. In addition, they allow for any factored function f over $\mathcal{V}$; not just probability densities.

Undirected edges only exist between factor nodes and variable nodes, where each factor node is connected only to those variables it is a function of. Formally, the undirected edges $e_{ij} \in \mathcal{E}$ exists between factor node ϕ_i and variable vertex v_j if and only if $v_j \in \mathcal{V}_i \subseteq \mathcal{V}$, where $\mathcal{V}_i$ denotes the corresponding set of dependencies.

A factor graph G defines the factorisation of a global function $f(\mathcal{V})$ as

$$f(\mathcal{V}) = \prod_i \phi_i(\mathcal{V}_i). \tag{4-4}$$

That is, the independence relationships are encoded by the edges.

Example: Consider the factor graph shown in Figure 4-2, which derived from the Bayes net example shown in Figure 4-1. The joint density $p(\mathcal{X}, \mathcal{Z})$ can now be written as

$$\begin{aligned} p(\mathcal{X} \mid \mathcal{Z}) \propto p(\mathcal{X}, \mathcal{Z}) \propto \phi_{0,1}(x_0, x_1)\phi_{1,2}(x_1, x_2)\phi_{2,3}(x_2, x_3)\phi_{3,4}(x_3, x_4) \\ \times \phi_{1,3}(x_1, x_3) \\ \times \phi_{0,4}(x_0, x_4)\phi_{1,4}(x_1, x_4). \end{aligned}$$

Each factor corresponds to a probability density or likelihood function; e.g.,

$$\phi_{0,1}(x_0, x_1) = \mathcal{L}(x_0, x_1; z_{0,1}) \propto \exp\Big(-\frac{1}{2} \| f_{0,1}(\mathcal{X}) \ominus z_{0,1} \|^2_{\Sigma_{0,1}} \Big).$$

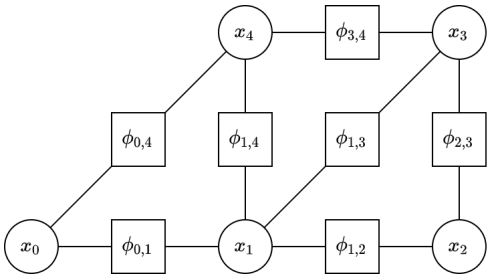

Figure 4-2: Factor graph example. The robot poses $x_0, \ldots, x_4$ are represented by graph variables, whereas the odometry constraints $\phi_{0,1}, \phi_{1,2}, \phi_{2,3}, \phi_{3,4}$, proximity constraints $\phi_{1,3}$ and loop-closure constraints $\phi_{0,4}, \phi_{1,4}$ are represented by graph factors. The edges encodes the dependence between the constraints and the corresponding robot poses.

4-4-3 Markov random field

A third way to express the probabilistic relationships in terms of graphical models is via Markov random fields (MRF) $G(\mathcal{V}, \mathcal{E})$ — basically a factor graph in which the factor nodes themselves are eliminated. The graph of an MRF is undirected and does not have factor nodes: its adjacency structure indicates which variables $\mathcal{X}$ are linked by a common factor (i.e., a measurement $\mathcal{Z}$). When considering pairwise cliques, at this level of abstraction, the expression for a pairwise MRF corresponds exactly to Equation (4-4).

Factor graphs are still better suited for the modelling of SLAM problems, as they allow for the expression of a finer-grained factorisation. Ternary (or higher arity) factors in a factor

graph are expressed in the equivalent MRF by a connection of all associated nodes in an undirected clique (i.e., a fully connected sub-graph). The information regarding the original factorisation is then lost.

Example: The MRF corresponding to the Bayes net example of Figure 4-1 is shown in Figure 4-3.

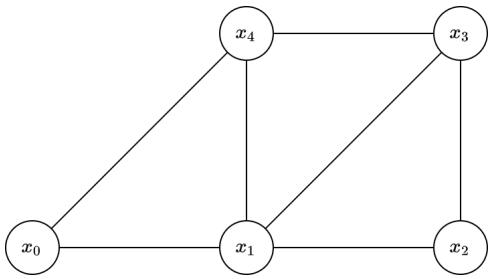

Figure 4-3: Markov random field example. Only the robot poses $x_0, \ldots, x_4$ are represented in the graph by graph variables. The edges represent the probabilistic relations between poses (i.e., constraints) implicitly.

4-5 The SLAM problem

Finally, after having introduced the necessary notation, data formats, measurement models and graphical models, the full SLAM optimisation problem is summarised, along with all the relevant notation.

In Section 4-2, the solution to the SLAM problem is found to be the set of poses $\mathcal{X}$ that maximises its likelihood $\mathcal{L}(\mathcal{X}; \mathcal{Z})$, given the set of measurements $\mathcal{Z}$. In Section 4-4 the joint probability distribution $p(\mathcal{X}, \mathcal{Z})$ (which is proportional to the $\mathcal{L}(\mathcal{X}; \mathcal{Z})$ with uniform measurement prior $p(\mathcal{Z})$) is expressed in terms of inference over a factor graph. The factor graph encodes the dependence between the measurements and the corresponding robot poses: its nodes correspond to the robot poses, and its factors correspond to the likelihood terms, which encode probabilistic constraints over a subset of nodes. Not only does the factor graph offer an insightful visualisation of the SLAM problem, its connectivity defines the sparsity of the resulting SLAM problem.

Following the description of the factor graph, the MAP estimate of Equation (4-1) factorises into

$$\mathcal{X}^* = \prod_{(i,j)\in\mathcal{I}} p(z_{i,j} \mid x_i, x_j),$$

where $\mathcal{I}$ defines the set of pairs of pose indices for which a constraint is defined. Since maximising the likelihood is identical to minimising the negative log-likelihood, the MAP estimate becomes

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} -\sum_{(i,j)\in\mathcal{I}} \ln\big(p(z_{i,j} \mid x_i, x_j)\big),$$

where $p(z_{i,j} \mid x_i, x_j)$ is defined in Equation (4-2). The log-likelihood for one measurement i is described as

$$-\ln\left(p(z_{i,j} \mid x_i, x_j)\right) = \underbrace{\ln\left(\sqrt{(2\pi)^{n_{\mathcal{SE}(n)}}|\Sigma_{i,j}|}\right)}_{\text{const.}} + \frac{1}{2}\|\mathbf{e}_{i,j}(\mathcal{X})\|^2_{\Sigma_{i,j}}, \tag{4-5}$$

with the first term being constant due to an assumed constant constraint covariance $\Sigma_{i,j}$. Finally, by neglecting the first constant term of each measurement log-likelihood, the pose SLAM problem can be formulated as a non-linear least squares problem:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_{(i,j)\in\mathcal{I}} \|\mathbf{e}_{i,j}(\mathcal{X})\|^2_{\Sigma_{i,j}}. \tag{4-6}$$

The error function, defined by Equation (4-3), defines a cost function term that corresponds the underlying soft constraint: the cost increases proportional to the deviation of the proposed solution from the constraint value, inversely scaled with the constraint covariance. This induces a solution that most closely matches the measurements that the SLAM system deems accurate (i.e., low constraint covariance). Constraints with higher covariance are considered more lenient, and will amount a lower cost per deviation. As such, the MAP solution is the set of poses that is considered the 'optimal compromise' over all measurements.

Note that this is only one definition of the full SLAM problem; one which corresponds to the definitions of Section 4-3. The SLAM problem format can be generalised by realising that the parameters $\mathcal{V}$ can be abstracted as *optimisation variables* that do not need to be the absolute poses (and/or features) of the robot; they are arbitrary variables which can be mapped to poses in real-world coordinates [38].

Chapter 5

Optimisation techniques

This chapters gives an overview of the popular optimisation techniques for graph optimisation. First, the linearisation of the optimisation problem is discussed in Section 5-1. Gradient methods are discussed in Section 5-2. Smoothing approaches are discussed in Section 5-3.

5-1 Linearisation

Non-linear optimisation methods converge to global minima by solving a succession of linear approximation of the cost function. Hence, in order to solve the least-squares problem using these methods, a linearised version of the problem needs to be constructed.

Linearising the least-squares problem give the following insights into the optimisation problem:

1. First, the error functions are linearised using a Taylor series expansion around a given linearisation point $\breve{\mathcal{X}}$. The function value at $\mathcal{X} = \breve{\mathcal{X}} + \Delta\mathcal{X}$ is then approximated as

$$\begin{aligned}\mathbf{e}_{i,j}(\mathcal{X}) &= \mathbf{e}_{i,j}(\breve{\mathcal{X}} + \Delta\mathcal{X}) \\ &\approx \mathbf{e}_{i,j}(\breve{\mathcal{X}}) + J_{\mathbf{e}_{i,j}}(\breve{\mathcal{X}})\Delta\mathcal{X},\end{aligned}$$

   where $J_{\mathbf{e}_{i,j}}(\breve{\mathcal{X}})$ is the Jacobian of $\mathbf{e}_{i,j}(\mathcal{X})$ computed at $\breve{\mathcal{X}}$:

$$J_{\mathbf{e}_{i,j}}(\breve{\mathcal{X}}) = \begin{bmatrix}\dfrac{\partial \mathbf{e}_{i,j}(\breve{\mathcal{X}})}{\partial x_1} & \cdots & \dfrac{\partial \mathbf{e}_{i,j}(\breve{\mathcal{X}})}{\partial x_N}\end{bmatrix}.$$

   With the error functions linearised, the *linear* least-squares cost function term is written as

$$\begin{aligned}F_{i,j}(\Delta\mathcal{X}) &\approx \|\mathbf{e}_{i,j}(\breve{\mathcal{X}}) + J_{\mathbf{e}_{i,j}}(\breve{\mathcal{X}})\Delta\mathcal{X}\|^2_{\Sigma_{i,j}} \\ &= \|\mathbf{b}_{i,j} + A_{i,j}\Delta\mathcal{X}\|^2_2\end{aligned} \qquad \text{with} \qquad \begin{cases} A_{i,j} = \Sigma_{i,j}^{-1/2} J_{\mathbf{e}_{i,j}}(\breve{\mathcal{X}}), \\ \mathbf{b}_{i,j} = \Sigma_{i,j}^{-1/2} \mathbf{e}_{i,j}(\breve{\mathcal{X}}), \end{cases}$$

   where $A_{i,j}$ and $\mathbf{b}_{i,j}$ represents the linear transformation that transforms the error function to have identity covariance (this process is called *whitening*). As such, the Mahalanobis norm is equivalent to the Euclidean norm (or 2-norm).

The structure of the linearised error term is of block-sparse nature, with only non-zero block-entries at the location of parameters connected to the constraint:

$$A_{i,j} = \begin{bmatrix} \cdots & \underbrace{\Sigma_{i,j}^{-1/2} \frac{\partial \mathbf{e}_{i,j}(\breve{\mathcal{X}})}{\partial x_i}}_{A_{i,j}^i} & \cdots & \underbrace{\Sigma_{i,j}^{-1/2} \frac{\partial \mathbf{e}_{i,j}(\breve{\mathcal{X}})}{\partial x_j}}_{A_{i,j}^j} & \cdots \end{bmatrix}$$

2. Second, all linearised error functions are stacked into a single matrix, yielding the linearised SLAM cost function

$$F(\Delta\mathcal{X}) = \|A\Delta\mathcal{X} + \mathbf{b}\|_2^2 = \left\| \begin{bmatrix} \vdots \\ A_{i,j} \\ \vdots \end{bmatrix} \Delta\mathcal{X} + \begin{bmatrix} \vdots \\ \mathbf{b}_{i,j} \\ \vdots \end{bmatrix} \right\|_2^2, \tag{5-1}$$

where $A = \Sigma^{-1/2} J_{\mathbf{e}}(\breve{\mathcal{X}})$ is the whitened Jacobian of the stacked error function $\mathbf{e}(\mathcal{X})$ at $\breve{\mathcal{X}}$ (with Σ the diagonally stacked covariance matrix), and $\mathbf{b} = \Sigma^{-1/2}\mathbf{e}(\breve{\mathcal{X}})$ is the stacked whitened local error term.

The whitened Jacobian matrix A is a sparse matrix with a block-structure that mirrors the structure of the underlying factor graph. Therefore, the Jacobian matrix is associated with the factor graph representation, because *the block-structure of A corresponds exactly to the adjacency matrix of the factor graph.*

Example: The linear least-squares problem corresponding to the factor graph in Figure 5-1 is written as

$$F(\Delta\mathcal{X}) = \|A\Delta\mathcal{X} + \mathbf{b}\|_2^2 = \left\| \begin{bmatrix} A_{0,1}^0 & A_{0,1}^1 & & & \\ & A_{1,2}^1 & A_{1,2}^2 & & \\ & & A_{2,3}^2 & A_{2,3}^3 & \\ & & & A_{3,4}^3 & A_{3,4}^4 \\ & A_{1,3}^1 & & A_{1,3}^3 & \\ A_{0,4}^0 & & & & A_{0,4}^4 \\ & A_{1,4}^2 & & & A_{1,4}^4 \end{bmatrix} \underbrace{\begin{bmatrix} \Delta x_0 \\ \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \Delta x_4 \end{bmatrix}}_{\Delta\mathcal{X}} + \begin{bmatrix} \mathbf{b}_{0,1} \\ \mathbf{b}_{1,2} \\ \mathbf{b}_{2,3} \\ \mathbf{b}_{3,4} \\ \mathbf{b}_{1,3} \\ \mathbf{b}_{0,4} \\ \mathbf{b}_{1,4} \end{bmatrix} \right\|_2^2.$$

The sparsity pattern clearly indicates which variables are connected to which error function.

3. The Hessian of the non-linear least-squares problem is approximated by the *information matrix* $\Lambda := A^\top A = J_{\mathbf{e}}(\breve{\mathcal{X}})^\top \Sigma^{-1} J_{\mathbf{e}}(\breve{\mathcal{X}})$. For the linearised problem, both are equivalent. In order to show the significance of the information matrix, the least-squares cost function is expanded as

$$\|A\Delta\mathcal{X} - \mathbf{b}\|_2^2 = \Delta\mathcal{X}^\top A^\top A \Delta\mathcal{X} + 2\Delta\mathcal{X}^\top A^\top \mathbf{b} + \mathbf{b}^\top \mathbf{b}. \tag{5-2}$$

The sparsity pattern of the information matrix is exactly the adjacency matrix of the corresponding MRF, which indicates why the information matrix is commonly associated with the MRF.

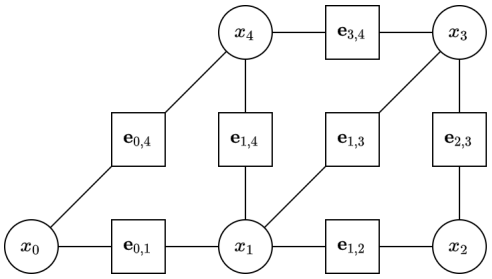

Figure 5-1: Generalised factor graph example. The robot poses $x_0, \ldots, x_4$ are represented by graph variables, whereas the odometry constraints $\phi_{0,1}, \phi_{1,2}, \phi_{2,3}, \phi_{3,4}$, proximity constraints $\phi_{1,3}$ and loop-closure constraints $\phi_{0,4}, \phi_{1,4}$ are represented by graph factors. The edges encodes the dependence between the constraints and the corresponding robot poses.

Example: The information matrix corresponding to the MRF in Figure 4-3 is written as

$$\Lambda = \begin{bmatrix} \Lambda_{11} & \Lambda_{12} & & & \Lambda_{15} \\ \Lambda_{21} & \Lambda_{22} & \Lambda_{23} & \Lambda_{24} & \Lambda_{25} \\ & \Lambda_{32} & \Lambda_{33} & \Lambda_{34} & \\ & \Lambda_{42} & \Lambda_{43} & \Lambda_{44} & \Lambda_{45} \\ \Lambda_{51} & \Lambda_{52} & & \Lambda_{54} & \Lambda_{55} \end{bmatrix}.$$

With the SLAM problem linearised as a linear least-squares problem, the optimal increment that minimises the cost is defined as

$$\Delta\mathcal{X}^* = \arg\min_{\Delta\mathcal{X}} \|A\Delta\mathcal{X} + \mathbf{b}\|_2^2.$$

The solution to the linearised SLAM problem, $\mathcal{X}^*$, is then obtained by adding the optimal increment $\Delta\mathcal{X}^*$ to the initial guess $\breve{\mathcal{X}}$:

$$\mathcal{X}^* = \breve{\mathcal{X}} + \Delta\mathcal{X}^*.$$

5-2 Gradient methods

A large family of iterative algorithms consider the curvature of the cost surface around the current estimate, ignoring the curvature farther away. However, when the state estimate is corrupted by significant noise, the local gradient may point toward a local minimum, not the global minimum. Unless gradient methods happen to stumble out of the local basin, they will typically fail.

A family of algorithm [34, 38, 39, 35] based on stochastic gradient descent (SGD) was initiated by the work of Olson et al. [34]. The basic idea of SGD is to consider a single edge and the gradient with respect to just that edge. The state is then moved in the direction of greatest descent. It is "stochastic" in the sense that the constraint is usually selected randomly. The distance that SGD travels for each edge is slowly decreased over time in order prevent oscillation with the result that it becomes increasing likely that SGD will get stuck in the most popular minimum (which is likely the global minimum).

It is often the case that the increment can be reasonably estimated much faster than it can be computed exactly. This is acceptable, since even an "exact" increment is only the solution to

a linear approximation of a non-linear problem. This motivates the consideration of iterative optimisation algorithms.

The iteration step follows from the linear least-squares solution of Equation (5-2) (while considering *only* constraint (i, j)) obtained by differentiating with respect to $\Delta\mathcal{X}$ and setting to zero [34]:

$$\Delta\mathcal{X}_{i,j}^{*} = -\underbrace{J_{\mathbf{e}}(\breve{\mathcal{X}})^{\top}\Sigma^{-1}J_{\mathbf{e}}(\breve{\mathcal{X}})}_{\Lambda^{-1}}\underbrace{J_{\mathbf{e}_{i,j}}(\breve{\mathcal{X}})\Sigma_{i,j}^{-1}\mathbf{e}_{i,j}(\breve{\mathcal{X}})}_{A_{i,j}^{\top}\mathbf{b}_{i,j}}$$

Note that Λ^{-1} represents the *full* Hessian, not just the Hessian with respect to constraint (i, j). Intuitively, the sequential procedure used in SGD can be understood by reading the update term from right to left:

1. $\mathbf{e}_{i,j}(\breve{\mathcal{X}})$ is the error vector at the linearisation point. Changing the parameter set in the direction of the negative error vector (i.e., the residual) decreases the error.
2. $\Sigma_{i,j}^{-1}$ is the information matrix of the measurement, which scales the residual components according to the information encoded in the constraint.
3. $J_{\mathbf{e}_{i,j}}(\breve{\mathcal{X}})$ is the Jacobian of the constraint, which maps the residual terms into a set of variations in the parameter space.
4. Λ is the approximated Hessian and represents the curvature of the error function. It scales the variations resulting from the Jacobian depending on the curvature of the error surface (Jacobi Preconditioning). The computationally expensive inverting of the Hessian is mitigated by approximating it as

$$\Lambda^{-1} \approx (\text{diag}(\Lambda))^{-1}.$$

The iterative update step is defined as

$$\mathcal{X}_{k+1} = \mathcal{X}_k - \underbrace{\lambda\Lambda^{-1}A_{i,j}^{\top}\mathbf{b}_{i,j}}_{\Delta\mathcal{X}_{i,j}^{*}}$$

where x_k denotes the k-th iteration (and linearisation) point, and where λ is the learning rate.

In an effort to simplify the problem, a state representation was proposed that captures the cumulative nature of the robot's motion and has a convenient structure for efficient computation; namely incremental poses:

$$\mathcal{X} = \left(\mathbf{0}, \begin{bmatrix}\mathbf{t}_1\\ \boldsymbol{\theta}_1\end{bmatrix}, \ldots, \begin{bmatrix}\mathbf{t}_{t+1} - \mathbf{t}_t\\ \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\end{bmatrix}, \ldots\right).$$

This parameterisation significantly simplifies the Jacobian with respect to a single constraint to

$$J_{i,j}(\breve{\mathcal{X}}) = \sum_{k=i+1}^{j} I_k,$$

where I_k is a zero matrix with a size that matches the Jacobian, but with a $I \in \mathbb{R}^{3\times3}$ at the k-th 3×3 block. Due to the simplicity of the Jacobian, it never needs to be explicitly constructed or stored.

5-3 Square-root SAM

As can be seen from the least-squares problem shown in Equation (5-1), performing MAP inference in non-linear SLAM requires repeatedly solving large and sparse linear systems. Sparsity in this case is dictated by the topology of the underlying factor graph (i.e., the factor graph is far from fully connected). The sparsity of these systems can be exploited to form specific and efficient algorithms.

The full SLAM problem can be concisely stated in terms of sparse linear algebra. The efficient factorisation of either the information matrix Λ (using sparse Cholesky factorisation) or the measurement Jacobian A (using sparse QR factorisation) into square root form can be used immediately to obtain the optimal robot trajectory. This family of approaches is referred to as *square-root SAM* [18]. In the case of linear measurement functions and additive normally distributed noise, the elimination algorithm is equivalent to sparse matrix factorisation.

5-3-1 The elimination algorithm

In the linear case, sparse matrix factorisation is equivalent to the *elimination algorithm*. The elimination algorithm converts a factor graph back to a Bayes net that contains only *unknown* variables of interest $\mathcal{X}$. This then allows for easy MAP inference.

Specifically, the elimination algorithm factorises the factor graph of the form $f(\mathcal{X})$ into a factored Bayes net probability density of the form

$$p(\mathcal{X}) = \prod_i p(x_i \mid \mathcal{X}_{\text{sep}(i)}),$$

where $\mathcal{X}_{\text{sep}(i)} \subseteq \mathcal{X}$ denotes the *separator* associated with x_i. The separator is the set of variables on which x_i is conditioned after elimination.

Elimination of nodes is done under a chosen variable ordering $x_1, \ldots, x_N$. The partially eliminated factor graph is denoted by $G_{i:N}$, where $x_i, \ldots, x_N$ are still to be eliminated. Starting with the complete factor graph $G_{1:N}$, the algorithm eliminates one variable x_i at a time by associating it with a conditional density, replacing its adjacent factor nodes with directed edges and adding an additional factor, yielding the reduced graph $G_{i+1:N}$.

More specifically, elimination of x_i is done in the following steps:

1. The set of all factors $f_j(\mathcal{X}_j) \in \mathcal{U}$ adjacent to x_i is removed from $G_{i:N}$.
2. All adjacent factors $f_j(\mathcal{X}_j)$ are multiplied into a product $\psi(x_i, \mathcal{X}_{\text{sep}(i)})$ that is factorised into a conditional distribution $p(x_i \mid \mathcal{X}_{\text{sep}(i)})$ and a new factor $f_i(\mathcal{X}_{\text{sep}(i)})$:

$$p(x_i \mid \mathcal{X}_{\text{sep}(i)})\ f_i(\mathcal{X}_{\text{sep}(i)}) = \psi(x_i, \mathcal{X}_{\text{sep}(i)}) = \prod_j f_j(\mathcal{X}_j).$$

   Here $\mathcal{X}_{\text{sep}(i)} = \{x \in \mathcal{X}_j\} \setminus x_i$.
3. The conditional density $p(x_i \mid \mathcal{X}_{\text{sep}(i)})$ is associated with node x_i and the new factor f_i is added to the graph. The modified graph is denoted by $G_{i+1:N}$.

The algorithm terminates when x_N is eliminated and $\mathcal{X}_{\text{sep}(i)} = \emptyset$, yielding only a prior $p(x_N)$ on x_N.

Example: The factor graph example shown in Figure 4-2 can be converted into the Bayes net shown in Figure 5-2 using the ordering x_0, x_4, x_1, x_2, x_3 corresponding to the factorisation

$$p(x_0, x_1, x_2, m_1, m_2) = p(x_0 \mid x_1, x_4)\ p(x_4 \mid x_1, x_3)\ p(x_1 \mid x_2, x_3)\ p(x_2 \mid x_3)\ p(x_3).$$

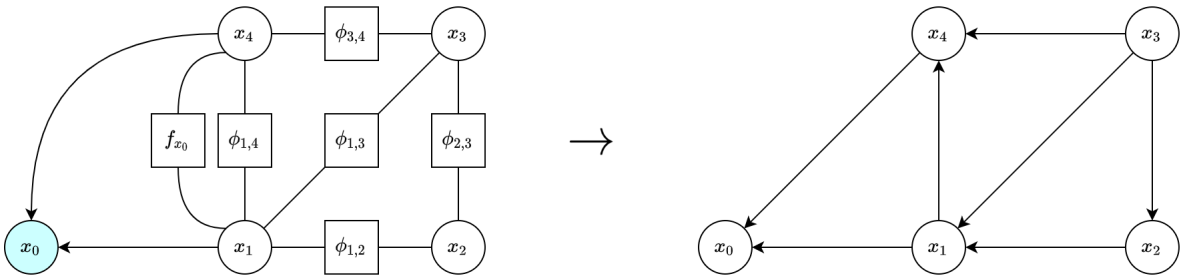

Figure 5-2: Factor graph example converted to Bayes net via variable elimination. First configuration shows the graph after the elimination of x_0, where an additional factor is added between the two connected poses x_1 and x_4; second configuration shows fully converted Bayes net.

5-3-2 Sparse matrix factorisation

For every variable x_i that is to be removed, the factor product $\psi(x_i, \mathcal{X}_{\text{sep}(i)})$ is written as

$$\psi(x_i, \mathcal{X}_{\text{sep}(i)}) = \prod_j f_j(\mathcal{X}_j) = \exp\Big(-\frac{1}{2}\sum_j \|A_j\Delta_{\mathcal{X}_j} - \mathbf{b}_j\|_2^2\Big) = \exp\Big(-\frac{1}{2}\|A_i(\Delta_{x_i}; \Delta_{\mathcal{X}_{\text{sep}(i)}}) - \mathbf{b}_i\|_2^2\Big).$$

The factor product can be factorised using the QR factorisation

$$\left[\ A_i \mid \mathbf{b}_i\ \right] = Q\begin{bmatrix} R_i & T_i & \mathbf{d}_i \\ & A_{f_i} & \mathbf{b}_{f_i} \end{bmatrix},$$

where R_i is upper-triangular and Q is orthogonal, resulting in the factorisation

$$\begin{aligned}\psi(x_i, \mathcal{X}_{\text{sep}(i)}) &= \exp\Big(-\frac{1}{2}\|A_i(\Delta_{x_i}; \Delta_{\mathcal{X}_{\text{sep}(i)}}\Big) - \mathbf{b}_i\|_2^2 \\ &= \underbrace{\exp\Big(-\frac{1}{2}\|R_i\Delta_{x_i} + T_i\Delta_{\mathcal{X}_{\text{sep}(i)}} - \mathbf{d}_i\|_2^2\Big)}_{p(x_i|\mathcal{X}_{\text{sep}(i)})} \underbrace{\exp\Big(-\frac{1}{2}\|A_{f_i}\Delta_{\mathcal{X}_{\text{sep}(i)}} - \mathbf{b}_{f_i}\|_2^2\Big)}_{f_i(\mathcal{X}_{\text{sep}(i)}}.\end{aligned} \tag{5-3}$$

Variable elimination transforms the Jacobian matrix (corresponding to a factor graph) to an upper-triangular matrix R (corresponding to a Bayes net). An equivalent matrix can be obtained via the Cholesky factorisation of the information matrix (corr responding to an MRF)

$$\Lambda = A^\top A = R^\top R.$$

Example: For the elimination of x_0 in the example shown in Figure 4-2, the factor product $\psi(x_0, x_1, x_4)$ is written as

$$\psi(x_0, x_1, x_4) = \exp\left(-\frac{1}{2} \left\| \begin{bmatrix} A^0_{0,1} & A^1_{0,1} & \\ A^4_{0,4} & & A^4_{0,4} \\ & A^0_{1,4} & A^4_{1,4} \end{bmatrix} \begin{bmatrix} \Delta_{x_0} \\ \Delta_{x_1} \\ \Delta_{x_4} \end{bmatrix} - \begin{bmatrix} b_{0,1} \\ b_{0,4} \\ b_{1,4} \end{bmatrix} \right\|^2_2 \right).$$

The first and final configuration of Figure 5-2 are described by the matrix

$$\left[\begin{array}{c|ccc} R_1 & T^1_1 & & T^2_1 \\ \hline & A^1_3 & A^2_3 & \\ & & & A_4 \\ & & & & A_5 \\ & \widetilde{A}^1_8 & & \widetilde{A}^2_8 \\ & & A^1_9 & & A^2_9 \end{array}\right] \quad \rightarrow \quad R = \begin{bmatrix} R_1 & T^1_1 & & T^2_1 & \\ & R_2 & T^1_2 & T^2_2 & \\ & & R_3 & & T_3 \\ & & & R_4 & \\ & & & & R_5 \end{bmatrix},$$

where $\widetilde{A}^1_8$ and $\widetilde{A}^2_8$ denote the modified matrices corresponding to the introduction of factor f_{x_0}.

5-3-3 Back-substitution

The exponent of the conditional linear-Gaussian densities in the factorisation of Equation (5-3) on the eliminated variables can be written as

$$\|R_i \Delta_{x_i} + T_i \mathcal{X}_{\text{sep}(i)} - \mathbf{d}_i\|^2_2 = (x_i - \mu_i)^\top R_i^\top R_i (x_i - \mu_i) = \|x_i - \mu_i\|^2_{\Sigma_i},$$

where the mean $\mu_i = R_i^{-1}(\mathbf{d}_i - T_i \mathcal{X}_{\text{sep}(i)})$ depends linearly on the separator $\mathcal{X}_{\text{sep}(i)}$ and the covariance matrix is given by $\Sigma_i = (R_i^\top R_i)^{-1}$.

Back-substitution allows for the MAP estimate to be obtained for each variable by proceeding in reverse elimination order. That is, for each variable in the reversed order, the MAP estimate is given by conditional mean.

5-3-4 Variable ordering

Different elimination orderings influence complexity by giving rise to different separator sizes throughout the elimination process. Larger separator sizes are the result of *fill-in*: the creation of dependencies in the sparse graphs between variables that were previously independent of each other.

An elimination ordering with minimum fill-in minimises the cost of the elimination/factorisation algorithm. Such an ordering can be approximating using ordering heuristics:

- *Minimum degree orderings* eliminate the least constrained variables of graph G first.
- *Nested dissection* algorithms try to exploit the fact that node elimination only introduces new constraints for a set of (spatially) direct neighbours by recursively partitioning the graph using a divide-and-conquer approach.

Chapter 6

Robustness

During the pose graph construction process of the SLAM front-end, sequential constraints are introduced by an odometry system that estimates the incremental change in the robot pose. Non-sequential constraints are introduced by a place recognition system, which represent loop closures or proximity matches.

There is a fundamental difference between edges generated by odometry and those generated by place recognition: odometry constraints, by definition, are topologically correct (even though they might be metrically inaccurate), while perceptual aliasing may cause the place recognition system to incorrectly associate two topologically unrelated places. Since least-squares optimisation methods are in general not robust against outliers, the back-end has to rely on the front-end to construct a topologically correct pose graph. Optimising an ill-defined represented is likely to produce defective solutions.

A number of methods have been proposed to solve the robust SLAM problem.

6-1 Switchable constraints

Switchable constraints rely on the premise that if constraint edges representing outliers and data association errors could be identified and removed during the optimisation process, the graph topology would be corrected and the optimisation could converge towards a correct solution [40]. The augmented optimisation problem then not only seeks the optimal configuration $\mathcal{X}^*$ of robot poses, but at the same time we also seek the optimal topology of the constraint graph.

Removing an edge from the factor graph corresponds to disabling the constraint associated with that edge, meaning that the disabled constraint should not have any influence on the optimisation process. This can be implemented as follows:

1. Introduction of a binary weight factor $\omega_{i,j} \in \{0, 1\}$ for every loop closure constraint would allow for loop constraints to be switched off whenever $\omega = 0$. Such discrete variables are not suited for least-squares optimisation methods, however, so the the binary weight

factor is replaced by a *switch variable* $s_{i,j} \in \mathbb{R}$ and a switch function that maps $s_{i,j} \mapsto \omega_{i,j} \in \{0, 1\}$. A suitable function is the sigmoid

$$\omega_{i,j} = \omega(s_{i,j}) : \mathbb{R} \to (0, 1) = \frac{1}{1 + \mathrm{e}^{-s_{i,j}}},$$

which is continuously differentiable and asymptotically converges towards 0 and 1.

2. In order to prevent the optimisation procedure from driving the weighting of all loop closure constraints to zero, a *prior constraint* can be introduced for each of the switch variables, which should loosely anchor the switch variable $s_{i,j}$ at their initial value $\gamma_{i,j}$. Since the loop closure constraints are proposed and requested by the front-end, it is reasonable to initially accept all loop-closure constraint, thereby initialising with $\omega_{i,j} = 1$.

The robust pose-graph optimisation problem is written as

$$\begin{aligned}\mathcal{X}^*, \mathcal{S}^* &= \arg\min_{\mathcal{X},\mathcal{S}} \Bigg(\overbrace{\sum_i \|f_{i,i+1}(\mathcal{X}) \ominus z_{i,i+1}\|^2_{\Sigma_{i,i+1}}}^{\text{odometry constraints}} + \overbrace{\sum_{(i,j)} \|\sigma(s_{i,j})[f_{i,j}(\mathcal{X}) \ominus z_{i,j})]\|^2_{\Sigma_{i,j}}}^{\text{switched loop closure constraints}} + \overbrace{\sum_{(i,j)} \|\gamma_{i,j} - s_{i,j}\|^2_{\Xi_{i,j}}}^{\text{switch prior constraints}} \Bigg) \\ &= \arg\min_{\mathcal{X},\mathcal{S}} \Bigg(\sum_i \|\mathbf{e}_{i,i+1}(\mathcal{X}\|^2_{\Sigma_{i,i+1}} + \sum_{(i,j)} \sigma^2(s_{i,j}) \|\mathbf{e}_{i,j}(\mathcal{X})\|^2_{\Sigma_{i,j}} + \sum_{(i,j)} \Xi^{-1}(1 - \sigma(s_{i,j}))^2 \Bigg),\end{aligned} \tag{6-1}$$

where $\mathcal{X}$ is the (familiar) set of robot poses and $\mathcal{S}$ is the set of switch variables. Ξ is a tuning parameter that acts as the covariance matrix used in the switch prior constraints: it adjusts the trade-off between robustness against outliers and the tendency to weight valid measurments down. The proposed robust problem formulation is an optimisation problem that consists of three types of constraints:

- Odometry constraints, which are unmodified with respect to the standard problem formulation;
- *Switched* loop-closure constraints, which allows for the deactivation of associated loop-closure constraints by driving the switch variable to low values;
- *Switch prior constraints*, which penalise the deactivation of loop-closure constraints.

The effect of switch variables can be thought of as acting on the information matrix $\Sigma^{-1}_{i,j}$ for loop closure constraint $z_{i,j}$: a negative switch variable drives the information matrix to zero, which indicates a weak constraint that can be ignored.

6-2 Dynamic covariance scaling

Dynamic covariance scaling extends the idea of switch constraints by interpreting the switch variables as scaling factors for the information matrix associated with the constraint [41]. A technique is introduced that provides an analytical solution to the scaling factors, which simplifies the optimisation problem by greatly reducing the number of variables.

The error terms due to a loop-closure constraint between poses x_i and x_j is follows from Equation (6-1) as

$$\omega_{i,j} \|\mathbf{e}_{i,j}(\mathcal{X})\|^2_{\Sigma_{i,j}} + (1 - \Xi^{-1}\omega_{i,j})^2.$$

The closed-form solution for computing the scaling factor $\omega_{i,j}$ is derived to be [41]

$$\omega_{i,j} = \min\left(1, \frac{2\Phi}{\Phi + \|\mathbf{e}_{i,j}(\mathcal{X})\|^2_{\Sigma_{i,j}}}\right) \quad \text{with} \quad \Phi = \Xi^{-1},$$

which dynamically scales the information matrix of each non-incremental edge (i, j) by $\omega^2_{i,j}$.

6-3 Dynamic covariance estimation

In standard SLAM, the transformation from a maximum likelihood estimator to a non-linear least-squares problem is done by taking the logarithm of the MAP estimate factorisation. For each measurement, the cost term is derived to be as in Equation (4-5), with a constant first term because of the constant constraint covariance. However, if the constraint covariance is added to the estimation process, this first term becomes variable and has to be included in the cost function. That is, the cost function becomes:

$$\begin{aligned}\mathcal{X}^*, \sigma^* &= \arg\max_{\mathcal{X},\sigma} \prod_{(i,j)\in\mathcal{I}} p(z_{i,j} \mid x_i, x_j, \sigma_{i,j}) \\ \text{with} \quad -\ln\left(p(z_{i,j} \mid x_i, x_j)\right) &= \underbrace{\ln\left(\sqrt{(2\pi)^{n_{\mathcal{SE}(n)}}|\Sigma_{i,j}|}\right)}_{\text{variable}} + \frac{1}{2}\|\mathbf{e}_{i,j}(\mathcal{X})\|^2_{\Sigma_{i,j}},\end{aligned}$$

where $\sigma^2_{i,j} = \Sigma_{i,j}$ is the standard deviation corresponding to the covariance matrix.

To keep the proposed equation valid for least-squares optimisation, it has to be guaranteed that $-\ln\left(p(z_{i,j} \mid x_i, x_j)\right) \geq 0$, which is not possible with arbitrary $\sigma_{i,j}$. Therefore, a lower bound $\sigma_{\min}$ is set, corresponding to which a regularisation term shifts the error term of each measurement [42]. Accordingly, the error function is derived to be

$$\begin{aligned}-\ln\left(p(z_{i,j} \mid x_i, x_j)\right) &\propto \ln\left(\sqrt{(2\pi)^{n_{\mathcal{SE}(n)}}|\Sigma_{i,j}|}\right) - \ln\left(\sqrt{(2\pi)^{n_{\mathcal{SE}(n)}}|\Sigma_{\min}|}\right) + \frac{1}{2}\|\mathbf{e}_{i,j}(\mathcal{X})\|^2_{\Sigma_{i,j}} \\ &\propto \ln\left(\frac{\sigma_{i,j}}{\sigma_{\min}}\right) + \frac{1}{2}\|\mathbf{e}_{i,j}(\mathcal{X})\|^2_{\Sigma_{i,j}} \\ &\propto \frac{1}{2}\ln\|\sigma_{i,j}\|^2_{\Sigma_{\min}} + \frac{1}{2}\|\mathbf{e}_{i,j}(\mathcal{X})\|^2_{\Sigma_{i,j}}, \end{aligned} \tag{6-2}$$

where $\Sigma_{\min} = \sigma^2_{\min}$ is the pre-defined covariance of the physical sensor under normal conditions.

Chapter 7

Calibration

The calibration parameters of a mobile robot play a substantial role in navigation tasks. Often these parameters are subject to variations that depend either on environmental changes or on the wear of the devices. These parameters typically include the position of the sensor on the platform or the parameters of the kinematic model that translates encoder ticks into a relative movement of the mobile base. The influence of the parameters on the accuracy of state estimation processes can be substantial. For instance, an accurate calibration of the odometry can seriously improve the expected accuracy of the motion prediction by reducing the search space of the algorithms that provide the motion estimates.

To obtain these parameters it is common to either rely on the specifications of the platform, to manually measure them, or to run ad-hoc calibration procedures before a mission (which are not able to estimate non-stationary parameters and need to be repeated whenever there is a potential change in the robot configuration). Calibration procedures, however, only rely on the data gathered by the robot and do not require any preparation or additional information. This approach allows a mobile robot, for example, to estimate a different set of odometry parameters for different regions of the environment and to better model the motion of the robot in these areas.

7-1 Sensor models

Kummerle et al. [36] proposed an approach to SLAM that combines the pose estimation of traditional SLAM with the simultaneous estimation of calibration parameters. In contrast to traditional SLAM, where the front-end handles the raw sensor data to formulate nicely formatted geometric relations, the estimation of calibration parameters requires explicit low-level modelling of relevant measurement equations. That is, the dependency relations of the front-end, which abstract sensor data into constraints that are manageable for estimation, need to be pulled into the back-end to be included in the optimisation problem.

Instead, the front-end only processes the raw sensor data and converts it to comply with a standardised format (e.g. raw accelerometer data is processed and presented to the back-end

as a vector of linear accelerations, along each of the principal axes, measured in m/s). As such, the specific hardware implementations are abstracted in the front-end, and all sensor types are expected to provide the back-end with measurements in identical format. This processing might be dependent on certain calibration parameters (such as e.g. the steering system geometry when processing the wheel encoder data to yield a linear speed and turn rate). With these processed measurements, constraints can be generated in terms of the robot poses and sensor model parameters.

For each of the following sensors, the estimated outputs are defined [37]:

- **Visual/LiDAR odometry**

  The exterioceptive sensors used in visual and LiDAR odometry track the environment to derive estimates of the relative distance between the consecutive measurement points S_t and S_{t+1}. The SLAM framework feeds the raw sensor data into the odometry algorithm, which then outputs the processed relative transformation estimate $z^{\text{scan}}_{t,t+1} := T^{(\mathsf{S}_t)}_{\mathsf{S}_{t+1}}$, which is the sensor frame S_{t+1} as seen from S_t. The next pose is described as

  $$x_{t+1} = T^{(\mathsf{O})}_{\mathsf{R}_{t+1}} = \big((T^{(\mathsf{O})}_{\mathsf{R}_t} \circ T^{(\mathsf{R}_t)}_{\mathsf{S}_t}) \circ T^{\mathsf{S}_t}_{\mathsf{S}_{t+1}}\big) \circ T^{\mathsf{S}_{t+1}}_{\mathsf{R}_{t+1}} = \big((x_t \circ \ell) \circ z^{\text{scan}}_{t,t+1}\big) \circ \ell^{-1}.$$

  From this, the constraint can be derived as

  $$\begin{aligned} \mathbf{e}^{\text{scan}}_{t,t+1}(\mathcal{X},\mathcal{P}) &= \big[\big((x_t \circ \ell) \circ z^{\text{scan}}_{t,t+1}\big) \circ \ell^{-1}\big] \ominus x_t \sim \mathcal{N}(\mathbf{0}, \Sigma^{\text{scan}}_{t,t+1}), \\ \text{or} \quad f^{\text{scan}}_{t,t+1}(\mathcal{X},\mathcal{P}) &= ((\ell^{-1} \circ x_t^{-1}) \circ x_{t+1}) \circ \ell \sim \mathcal{N}(z_{t,t+1}, \Sigma^{\text{scan}}_{t,t+1}), \end{aligned}$$

  where $\mathcal{P}$ denotes the set of all calibration parameters, and $\ell \in \mathcal{P}$ denotes the relative sensor frame.

- **Odometer**

  The odometer model gives a measurement $z^{\text{odo}}_t := (\mathbf{v}^{(\mathsf{S}_t)}_{\mathsf{S}_t}, \boldsymbol{\omega}^{(\mathsf{S}_t)}_{\mathsf{S}_t}) \in \mathbb{R}^{n\mathcal{SE}(n)}$ of the linear speed $\mathbf{v}^{(\mathsf{S}_t)}_{\mathsf{S}_t} \in \mathbb{R}^n$ and turn rate $\boldsymbol{\omega}^{(\mathsf{S}_t)}_{\mathsf{S}_t} \in \mathbb{R}^{n\mathcal{SO}(n)}$ of the sensor frame S_t as measured in the same frame at time t. In this sensor model, the gyroscope is assumed to be at the centre of rotation of the robot; therefore $\ell = T(\mathbf{0})$ and $x_t = T^{(\mathsf{O})}_{\mathsf{R}} = T^{(\mathsf{O})}_{\mathsf{S}}$. The measurements are transformed to the inertial frame O via the rigid body transformation per Equation (2-11) as

  $$\begin{bmatrix} \mathbf{v}^{(\mathsf{O})}_{\mathsf{S}} \\ \boldsymbol{\omega}^{(\mathsf{O})}_{\mathsf{S}} \end{bmatrix} = \begin{bmatrix} R^{(\mathsf{O})}_{\mathsf{S}} & (\mathbf{t}^\wedge)^{(\mathsf{O})}_{\mathsf{S}} R^{(\mathsf{O})}_{\mathsf{S}} \\ 0 & R^{(\mathsf{O})}_{\mathsf{S}} \end{bmatrix} \begin{bmatrix} \mathbf{v}^{(\mathsf{S})}_{\mathsf{S}} \\ \boldsymbol{\omega}^{(\mathsf{S})}_{\mathsf{S}} \end{bmatrix} \quad \Longrightarrow \quad \begin{aligned} \mathbf{v}^{(\mathsf{O})}_{\mathsf{S}} &= \mathbf{t}^{(\mathsf{O})}_{\mathsf{S}} \times (R^{(\mathsf{O})}_{\mathsf{S}} \boldsymbol{\omega}^{(\mathsf{O})}_{\mathsf{S}}) + R^{(\mathsf{O})}_{\mathsf{S}} \mathbf{v}^{(\mathsf{S})}_{\mathsf{S}}, \\ \boldsymbol{\omega}^{(\mathsf{O})}_{\mathsf{S}} &= R^{(\mathsf{O})}_{\mathsf{S}} \boldsymbol{\omega}^{(\mathsf{S})}_{\mathsf{S}}, \end{aligned}$$

  where $\mathbf{t}^{(\mathsf{O})}_{\mathsf{S}} \in \mathbb{R}^n$ and $R^{(\mathsf{O})}_{\mathsf{S}} \in SO(n)$ are the translation and rotation of the sensor frame S relative to the inertial frame O. Therefore, $\mathbf{t}^{(\mathsf{O})}_{\mathsf{S}}$ and $R^{(\mathsf{O})}_{\mathsf{S}}$ fully define the rigid body transformation $T^{(\mathsf{O})}_{\mathsf{S}}$. By assuming a constant velocity for $\tau \in [\tau_t, \tau_t + \Delta\tau]$ for a sufficiently small time-step $\Delta\tau$, the motion follows from the differential equation that describes the Lie algebra $\mathfrak{se}(n)$ as

  $$x_{t+1} = \exp\Bigg(\underbrace{\begin{bmatrix} (\boldsymbol{\omega}^\wedge)^{(\mathsf{O})}_{\mathsf{S}} & \mathbf{v}^{(\mathsf{O})}_{\mathsf{S}} \\ 0 & 0 \end{bmatrix}}_{\xi^\wedge \in \mathfrak{se}(n)} \Delta\tau\Bigg) x_t.$$

  Therefore, the constraint is derived to be

  $$\mathbf{e}^{\text{odo}}_{t,t+1}(\mathcal{X},\mathcal{P}) = x_{t+1} \ominus \exp(\xi^\wedge \Delta\tau) x_t \sim \mathcal{N}(\mathbf{0}, \Sigma^{\text{odo}}_{t,t+1}).$$

- **Gyroscope**

  The gyroscope gives a measurement of the turn rate $z_t^{\text{gyro}} := \boldsymbol{\omega}_{\mathsf{S}_t}^{(\mathsf{S}_t)} \in \mathbb{R}^{n_{\mathcal{SO}(n)}}$ of the sensor frame S_t measured in the same frame at time t. In this sensor model, the gyroscope is assumed to be at the centre of rotation of the robot; therefore $\ell = T(\mathbf{0})$ and $x_t = T_{\mathsf{R}}^{(\mathsf{O})} = T_{\mathsf{S}}^{(\mathsf{O})}$. This quantity is transformed to the inertial frame O per Equation (2-11) as

  $$\boldsymbol{\omega}_{\mathsf{S}}^{(\mathsf{O})} = R_{\mathsf{S}}^{(\mathsf{O})} \boldsymbol{\omega}_{\mathsf{S}}^{(\mathsf{S})},$$

  where $R_{\mathsf{S}}^{(\mathsf{O})} = R_{\mathsf{R}}^{(\mathsf{O})} \in SO(3)$ is the rotation of S relative to O. By assuming a constant velocity for $\tau \in [\tau_t, \tau_t + \Delta\tau]$ for a sufficiently small time-step $\Delta\tau$, the motion follows from the differential equation that describes the Lie algebra $\mathfrak{so}(n)$ as

  $$R_{t+1} = \exp\left((\boldsymbol{\omega}_{\mathsf{S}}^{(\mathsf{O})})^{\wedge} \Delta\tau\right) R_t.$$

  Therefore, the constraint is derived to be

  $$\mathbf{e}_{t,t+1}^{\text{gyro}}(\mathcal{X}, \mathcal{P}) = R_{t+1} \ominus \exp\left((\omega^{\wedge})_{\mathsf{S}}^{(\mathsf{O})} \Delta\tau\right) R_t \sim \mathcal{N}(\mathbf{0}, \Sigma_{t,t+1}^{\text{gyro}}).$$

- **Accelerometer**

  The accelerometer sensor gives an estimate for the linear acceleration along n axes, including gravity, as $z_t^{\text{acc}} := \mathbf{a}_{\mathsf{S}}^{(\mathsf{S})} \in \mathbb{R}^n$. In this sensor model, the gyroscope is assumed to be at the centre of rotation of the robot; therefore $\ell = T(\mathbf{0})$ and $x_t = T_{\mathsf{R}}^{(\mathsf{O})} = T_{\mathsf{S}}^{(\mathsf{O})}$. Assuming a constant acceleration between $\tau \in [\tau_t, \tau_t + \Delta\tau]$ for a sufficiently small time-step $\Delta\tau$, the average velocity can be estimated to be

  $$\mathbf{v}_{\mathsf{S}}^{(\mathsf{S})} = \frac{x_t \ominus x_{t-1}}{\Delta\tau} + \mathbf{a}_{\mathsf{S}}^{(\mathsf{S})} \Delta\tau.$$

  The estimates are transformed to the inertial frame O per Equation (2-11) as

  $$\mathbf{v}_{\mathsf{S}}^{(\mathsf{O})} = R_{\mathsf{S}}^{(\mathsf{O})} \mathbf{v}_{\mathsf{S}}^{(\mathsf{S})},$$

  where $R_{\mathsf{S}}^{(\mathsf{O})} = R_{\mathsf{R}}^{(\mathsf{O})} \in SO(3)$ is the translation of S relative to O. The motion follows from the differential equation that describes the Lie algebra $\mathfrak{se}(n)$ as

  $$x_{t+1} = \exp\Bigg(\underbrace{\begin{bmatrix} 0 & \mathbf{v}_{\mathsf{S}}^{(\mathsf{O})} \\ 0 & 0 \end{bmatrix}}_{\xi^{\wedge} \in \mathfrak{se}(n)} \Delta\tau\Bigg) x_t.$$

  Therefore, the constraint is derived to be

  $$\mathbf{e}_{t,t+1}^{\text{acc}}(\mathcal{X}, \mathcal{P}) = \mathbf{t}_{t+1} - \exp\left((\omega^{\wedge})_{\mathsf{S}}^{(\mathsf{O})} \Delta\tau\right) \mathbf{t}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_{t,t+1}^{\text{acc}}).$$

- **Absolute position**

  An absolute position measurement $z_t^{\text{abs}} := \mathbf{t}_{\mathsf{S}}^{(\mathsf{O})}$ expresses the sensor frame S relative to the inertial frame O at time t. This measurement is related to the robot location as

  $$\mathbf{t}_t = \mathbf{t}_{\mathsf{R}_t}^{(\mathsf{O})} = \mathbf{t}_{\mathsf{S}_t}^{(\mathsf{O})} + \mathbf{t}_{\mathsf{R}_t}^{(\mathsf{S}_t)} = z_t^{\text{abs}} + \mathbf{t}_{\ell},$$

  where $\mathbf{t}_{\ell}$ denotes the position component of the relative sensor frame. The constraint can quite straightforwardly be derived as

  $$\begin{aligned} \mathbf{e}_t^{\text{abs}}(\mathcal{X}, \mathcal{P}) &= (z_t^{\text{abs}} + \mathbf{t}_{\ell}) - \mathbf{t}_t \sim \mathcal{N}(\mathbf{t}_t, \Sigma_t^{\text{abs}}) \\ \text{or} \quad f_t^{\text{abs}}(\mathcal{X}, \mathcal{P}) &= \mathbf{t}_t - \mathbf{t}_{\ell}. \end{aligned}$$

The estimates that the sensor models provide, along with the constraints that can be derived from these estimates in combination with other parameters, are summarised in Table 7-1

Sensor model	Estimate	Constraint
Visual/LiDAR odometry	Relative transformation $T_{\mathsf{S}_{t+1}}^{(\mathsf{S}_t)} \in \mathcal{SE}(n)$	$\mathbf{e}_{t,t+1}^{\text{scan}}(x_t, \ell, T_{\mathsf{S}_{t+1}}^{(\mathsf{S}_t)}) = \mathbf{0}$
Odometer	Linear velocity $\mathbf{v}_{\mathsf{S}}^{(\mathsf{S})} \in \mathbb{R}^n$ Turn rate $\boldsymbol{\omega}_{\mathsf{S}}^{(\mathsf{S})} \in \mathbb{R}^{n_{\mathcal{SO}(n)}}$	$\mathbf{e}_{t,t+1}^{\text{odo}}(x_t, \mathbf{v}_{\mathsf{S}}^{(\mathsf{S})}, \boldsymbol{\omega}_{\mathsf{S}}^{(\mathsf{S})}) = \mathbf{0}$
Gyroscope	Turn rate $\boldsymbol{\omega}_{\mathsf{S}}^{(\mathsf{S})} \in \mathbb{R}^3$	$\mathbf{e}_{t,t+1}^{\text{gyro}}(x_t, \boldsymbol{\omega}_{\mathsf{S}}^{(\mathsf{S})}) = \mathbf{0}$
Accelerometer	Linear acceleration $\mathbf{a}_{\mathsf{S}}^{(\mathsf{S})} \in \mathbb{R}^n$	$\mathbf{e}_{t,t+1}^{\text{acc}}(x_{t-1}, x_t, x_{t+1}, \mathbf{a}_{\mathsf{S}}^{(\mathsf{S})}) = \mathbf{0}$
Absolute position	Absolute position $\mathbf{t}_{\mathsf{S}}^{(\mathsf{O})} \in \mathbb{R}^n$	$\mathbf{e}_{t,t+1}^{\text{abs}}(x_t, \ell, \mathbf{t}_{\mathsf{S}}^{(\mathsf{O})}) = \mathbf{0}$

Table 7-1: Summary of estimates and constrained obtained via sensor models.

7-2 Calibration parameters

Each of the sensor models work with estimates derived from the raw sensor data and processed to comply with a standardised format. With these estimates, constraints are generated that relate the set of robot poses $\mathcal{X}$ and the set of calibration parameters $\mathcal{P}$. These calibration parameters relate to the constraints as follows:

- *Physical quantities*, which are involved in the model that maps the raw sensor data to the processed estimate. For instance, a wheeled robot that uses a set of wheel encoders to form its odometry estimates, the geometry of the steering system needs to accounted for. Furthermore, the wheel radius might be variable depending on its location or time instant. Since the standardised estimates are then functions of the raw data and the (physical quantity) calibration parameters, their values can be optimised by minimising the sum over all error functions.
- *Scaling* and *bias* (c_1 and c_2, respectively), which follow the assumption that a linear function of the measurement estimate is a more accurate depiction of the true value. That is, the 'corrected' measurement is described by

  $$z = c_1 \hat{z} + c_2,$$

  where z and $\hat{z}$ denote the corrected and uncorrected measurement estimate, respectively.
- *Sensor location*, which describes the position of sensor relative to the coordinate frame of the robot. This offset can be a significant error source and be cumbersome to measure accurately. This parameter is essentially the (constant) relative transform $\ell = T_{\mathsf{S}}^{(\mathsf{R})}$ between the sensor frame S and the robot frame R at each time t.

The varying nature of calibration parameters can be accounted for by modelling the time-evolution by a random-walk model [43]. This requires a different calibration value for each time-step, combined with the constraint

$$c_{t+1} = c_t + \epsilon \quad \text{with} \quad \epsilon \sim \mathcal{N}(0, \sigma_c^2),$$

where σ_c^2 is a tuning parameter that models the variance of the random-walk process.

7-3 Self-calibrating SLAM formulation

The least-squares formulation of the SLAM problem of Equation (4-6) can be extended to include the calibration parameters. For this purpose, the constraints that relate the set of robot poses $\mathcal{X}$ and the set of calibration parameters $\mathcal{P}$ are added to the cost function:

$$\mathcal{X}^*, \mathcal{P}^* = \arg\min_{\mathcal{X},\mathcal{P}} \sum_{s\in\mathcal{S}} \left[\sum_{i\in\mathcal{I}^s} \|\mathbf{e}_i^s(\mathcal{X},\mathcal{P})\|^2_{\Sigma_i^s} \right], \tag{7-1}$$

where $\mathcal{S}$ denotes the set of constraint-type identifiers (which includes identifiers for all sensor models along with the null identifier that refers to the traditional error terms that are independent of calibration parameters), and $\mathcal{I}^s$ is the set of (unary or binary) indices, for which constraints of type $s \in \mathcal{S}$ are defined.

7-4 Extended self-calibrating SLAM formulation

The self-calibrating SLAM problem can be extended to also optimise over the constraint covariance. The approach for implementing this feature is derived from the notions of dynamic covariance estimation, which are discussed in Section 6-3.

The negative log-likelihood formulation that neatly describes the pose SLAM problem as a least-squares problem neglects the constant terms that scale the probability density function of each constraint. When considering a variable constraint covariance, however, this terms becomes variable and cannot be neglected. This adds another term to the cost function that conveniently acts as a regularisation term and prevents the covariance from moving to infinity (which is the optimal solution without the regularisation term).

Following the derivation of Equation (6-2), the extended self-calibrating SLAM formulation is given as

$$\mathcal{X}^*, \mathcal{P}^*, \mathcal{C}^* = \arg\min_{\mathcal{X},\mathcal{P},\mathcal{C}} \sum_{s\in\mathcal{S}} \left[\sum_{i\in\mathcal{I}^s} \left(\|\mathbf{e}_i^s(\mathcal{X},\mathcal{P})\|^2_{(\sigma_i^s)^2} + \ln \|\sigma_i^s\|^2_{\Sigma^s_{\min}} \right) \right], \tag{7-2}$$

where $(\sigma_i^s)^2 = \Sigma_i^s \in \mathcal{C}$ is the (variable) constraint covariance and the tuning parameter $\Sigma^s_{\min}$ is its lower bound. The set of all constraint covariances is denoted by $\mathcal{C}$.

The extended self-calibrating SLAM problem finds set of poses, the set of calibration parameters, and the set of constraint covariances, that minimises the cost over all constraint. Like with traditional SLAM, the cost can be interpreted as the deviation of the solution from the constraint values, inversely scaled by the constraint covariance. This time, however, the constraint covariance is also an optimisation variable.

Chapter 8

Conclusion

8-1 Summary, motivation and contribution

The goal of this research is to assist Demcon's Robotics Competence Group at gaining expertise in the field of robotic navigation. With this expertise, Demcon can aid customers in building bespoke navigation and mapping systems for a variety of applications.

To supplement this development, Demcon has built a test platform based on the BigBot wheeled robot. BigBot is loaded with sensors popularly used in robotic navigation (i.e., LiDAR, RGB-D cameras, IMU, differential drive unit). This allows for the testing of popular off-the-shelf frameworks and even the development, implementation and validation of new approaches.

The problem of navigation is formalised as the Simultaneous Localisation and Mapping (or SLAM) problem. An approach to SLAM incrementally builds a consistent map of an unknown environment, while simultaneously finding its own location within that map. Modern solutions to SLAM are based on the pose-graph formulation, which discretises the robot trajectory as a set of reference locations in space and time, and finds the optimal compromise of robot poses over all observations. These observations are used to derive relations between poses in the form of *constraints*: poses are constrained relative to each other at the *constraint value* with a strength given by the inverse *constraint covariance* (which measures constraint accuracy). The optimal compromise is found by taking into account all constraints, and taking the minimal weighted deviations from those constraints as an optimum.

A typical SLAM approach has a front-end, which abstracts sensor data into models that are amenable for estimation, and a back-end, which performs inference on the abstracted data. Basically, the front-end constructs the graph from raw sensor data, and the back-end optimises the graph to find the SLAM solution. The models that generate the constraints from observations are dependent on a variety of parameters that model real-life quantities. These parameters need to be estimated (or measured) accurately to have the best possible constraint accuracy. Live-estimating these parameters would significantly simplify the initialisation process, since only an initial estimate is required, with the algorithm converging

to a more accurate value over time. Furthermore, live-estimation could also account for non-constant parameters.

Since the pose SLAM problem is fully defined in terms of the constraints in its pose-graph, the constraint covariance also plays a significant role. This tuning parameter is traditionally set a priori and assumed constant. However, in reality, since this quantity expresses the accuracy of a constraint, time and operation conditions affect its value. As such, having a procedure that includes this quantity in the optimisation process could result in more accurate results. Furthermore, the importance of tuning is reduced, which simplifies the initial configuration process.

By adding both the sensor models (with corresponding calibration parameters) and the covariance to the optimisation problem, a form of self-calibrating SLAM can be developed that requires minimal initial setup. This is what this thesis work aims to contribute to. The expected contributions are as follows:

- A novel formulation is introduced of the sensor models that is based on Lie theory, Lie groups and its corresponding kinematics. Lie groups are generally used in robotics because of their efficient and robust parameterisation of rotations and transformations. While the idea for the inclusion of sensor models in the optimisation problem stems from [37], where sensor models are based on quaternions, a group-based formulation has not been found.
- By successfully implementing the group-based formulation of the sensor models, this thesis can provide a proof of concept for future implementations.
- The constraint covariance is included in the optimisation process with the goal of increasing the accuracy of the SLAM solution, by allowing the covariance to converge to a value that best matches the observations. The method for including the covariance is similar to that of [42]. However, in this case the goal was improving robustness, which could be found to be a side effect present in the self-calibrating SLAM approach of this thesis work.
- The calibration parameters and constraint covariances are saved and analysed to reveal time-dependence or location-dependence. By visualising these parameters over time and topologically on a map, insights into the evolution of these parameters can be made explicit, making for a useful tool to have.

8-2 Problem statement

The self-calibrating SLAM problem is incrementally build up to its final form, starting with:

1. The traditional pose-graph optimisation problem, defined as

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_{i \in \mathcal{I}} \|\mathbf{e}_i(\mathcal{X})\|^2_{\Sigma_i}, \tag{8-1}$$

   where the set of poses $\mathcal{X}$ is sought that minimises the scaled cost over all error functions.

2. The self-calibrating pose-graph optimisation problem, defined as

$$\mathcal{X}^*, \mathcal{P}^* = \arg\min_{\mathcal{X},\mathcal{P}} \sum_{s \in \mathcal{S}} \left[\sum_{i \in \mathcal{I}^s} \|\mathbf{e}^s_i(\mathcal{X}, \mathcal{P})\|^2_{\Sigma_i} \right], \tag{8-2}$$

where the set of poses $\mathcal{X}$ and set of calibration parameters $\mathcal{P}$ is sought that minimises the scaled cost over the error functions of all constraint-types. In this problem formulation, the sensor models are pulled into the optimisation problem, and the calibration parameters are considered as optimisation variables.

3. The covariance-scaling pose-graph optimisation problem, defined as

$$\mathcal{X}^*, \mathcal{C}^* = \arg\min_{\mathcal{X},\mathcal{C}} \sum_{i\in\mathcal{I}} \left(\|\mathbf{e}_i(\mathcal{X})\|^2_{(\sigma_i^s)^2} + \ln \|\sigma_i^s\|^2_{\Sigma^s_{\min}} \right), \tag{8-3}$$

where the set of poses $\mathcal{X}$ and the set of constraint covariances $\mathcal{C}$ is sought that minimises the scaled cost over all error functions. In this problem formulation, the constraint covariances can be independently adjusted for each constraint.

4. The full self-calibrating pose-graph optimisation problem, defined as

$$\mathcal{X}^*, \mathcal{P}^*, \mathcal{C}^* = \arg\min_{\mathcal{X},\mathcal{P},\mathcal{C}} \sum_{s\in\mathcal{S}} \left[\sum_{i\in\mathcal{I}^s} \left(\|\mathbf{e}_i^s(\mathcal{X},\mathcal{P})\|^2_{(\sigma_i^s)^2} + \ln \|\sigma_i^s\|^2_{\Sigma^s_{\min}} \right) \right], \tag{8-4}$$

where the set of poses $\mathcal{X}$, the set of calibration parameters $\mathcal{P}$, and the set of constraint covariances $\mathcal{C}$ is sought that minimises the scaled cost over the error functions of all constraint-types. In this problem formulation, the calibration parameters can be adjusted to minimise the error functions defined for each constraint type, while simultaneously the constraint covariance can be adjusted per constraint.

Due to the complexity of the proposed optimisation problem, the research is conducted in a similar incremental fashion. Firstly, to establish the validity of this problem formulation, the following problem statement is formulated (regarding approaches 2-4):

> *Can each of the extended approaches to SLAM be implemented to converge to an optimal solution?*

Furthermore, to establish the validity of the solution, the following problem statements are formulated:

> *Under what conditions do each of the extended approaches to SLAM converge to an optimal solution?*
>
> *Does each of the extended approaches to SLAM lead to an accurate approximation of the physical quantities that this solution represents?*

To assess the performance of the approaches, the following problem statement is formulated:

> *How is the performance of each of the extended approaches to SLAM related to the others? Does each extension make an observable improvement, with the last approach the most improved?*

Relevant performance metrics are the average trajectory error (ATE) and the relative pose error (RPE), each of which are defined when a ground truth is available (which is the caes in simulation).

To investigate the benefits of mapping the calibration parameters and constraint covariances over time and space, the following problem statement is formulated:

What insights can be gained from the analysis of the history of calibration parameters and constraint covariances? Is a strong correlation discernible between a certain value and environment description?

Finally, to investigate the side effects of implementing the constraint covariance in the optimisation problem, the following problem statement is formulated:

Does the inclusion of constraint covariance improve robustness against erroneous loop-closure constraints?

8-3 Thesis work

The main contribution of this thesis work is the implementation of the different stages of the full self-calibrating SLAM approach. It will be tested in simulation against traditional approaches using test data from the BigBot (and potentially using popular datasets). The thesis work will be structured as follows:

1. Decide no the code structure of the self-calibrating SLAM framework: e.g., the implementation of an optimiser API, the programming language (`python`/`MATLAB`/`C++`), the file structure, implementation in ROS.
2. Start with the implementation of the traditional SLAM approach in the decided code structure to act as a baseline implementation. Develop each of the sensor models and validate accuracy from test data. By incrementally implementing these sensor model in the baseline implementation, simulation-testing should result in an efficient development process.
3. Implement the constraint covariance estimation in a parallel baseline implementation. Validate the approach via simulation-testing.
4. Implement the approaches as ROS packages. Extensively testing on hardware in this phase will ensure that the implementations are valid for use in ROS.
5. Combine the calibration parameter estimation and the constraint covariance estimation in a single implementation of the full self-calibrating SLAM approach.
6. Develop the parameter mapping approach and perform experiments on BigBot.
7. Process the results and document the process.

Bibliography

[1] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," pp. 1–17, 2018.

[2] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–108, 2006.

[3] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[4] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, pp. 31–43, 12 2010.

[5] G. Dissanayake, S. Huang, Z. Wang, and R. Ranasinghe, "A review of recent developments in Simultaneous Localization and Mapping," *2011 6th International Conference on Industrial and Information Systems, ICIIS 2011 - Conference Proceedings*, pp. 477–482, 2011.

[6] S. Huang and G. Dissanayake, "A critique of current developments in simultaneous localization and mapping," *International Journal of Advanced Robotic Systems*, vol. 13, no. 5, pp. 1–13, 2016.

[7] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, and I. Reid, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, pp. 1309–1332, 12 2016.

[8] R. Valencia and J. Andrade-Cetto, *Mapping, Planning and Exploration with Pose SLAM*, vol. 119. 2018.

[9] M. Sualeh and G. W. Kim, "Simultaneous Localization and Mapping in the Epoch of Semantics: A Survey," *International Journal of Control, Automation and Systems*, vol. 17, no. 3, pp. 729–742, 2019.

[10] S. Huang, "A review of optimisation strategies used in simultaneous localisation and mapping," *Journal of Control and Decision*, vol. 6, no. 1, pp. 61–74, 2019.

[11] F. Dellaert and M. Kaess, "Factor Graphs for Robot Perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

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

[13] M. Duarte-Silva, J. Henriques-Calado, and V. Camotim, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem Michael," *Women and Therapy*, vol. 35, no. 3-4, pp. 221–232, 2012.

[14] R. C. Smith and P. Cheeseman, "On the Representation and Estimation of Spatial Uncertainty," *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[15] R. C. Smith, M. Self, and P. Cheeseman, "A stochastic map for uncertain spatial relationships," *Proceedings of the 4th international symposium on Robotics Research*, no. 0262022729, pp. 467–474, 1987.

[16] R. C. Smith, M. Self, and P. Cheeseman, "Estimating Uncertain Spatial Relationships in Robotics," *Autonomous Robot Vehicles*, pp. 167–193, 1990.

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

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

[19] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[20] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *International Journal of Robotics Research*, vol. 31, pp. 216–235, 2 2012.

[21] F. Lu and E. Milios, "Globally Consistent Range Scan Alignment for Environment Mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[22] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005.

[23] V. Ila, J. M. Porta, and J. Andrade-Cetto, "Information-based compact pose SLAM," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 78–93, 2010.

[24] M. Quigley and Others, "ROS: an open-source Robot Operating System," in *IEEE ICRA workshop on open source software*, pp. 1–5, 2009.

[25] R. M. Murray, Z. Li, and S. Shankar Sastry, *A mathematical introduction to robotic manipulation*. 2017.

[26] E. G. Hemingway and O. M. O'Reilly, "Perspectives on Euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments," *Multibody System Dynamics*, vol. 44, no. 1, pp. 31–56, 2018.

[27] E. Eade, "Lie Groups for 2D and 3D Transformations," tech. rep., 2013.

[28] C. Hertzberg, "A Framework for Sparse, Non-Linear Least Squares Problems on Manifolds," *UNIVERSITÄT BREMEN*, 2008.

[29] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the National Conference on Artificial Intelligence*, 2002.

[30] M. Labbe, "RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[31] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Robotics: Science and Systems*, 2014.

[32] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*, Springer, July 2014.

[33] M. Labbé and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 734–745, 2013.

[34] E. Olson, J. J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, pp. 2262–2269, 2006.

[35] "Nonlinear constraint network optimization for efficient map learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 428–439, 2009.

[36] R. Kümmerle, G. Grisetti, and W. Burgard, "Simultaneous calibration, localization, and mapping," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3716–3721, 2011.

[37] D. A. Cucci and M. Matteucci, "Position tracking and sensors self-calibration in autonomous mobile robots by Gauss-Newton optimization," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1269–1275, 2014.

[38] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," *Robotics: Science and Systems*, vol. 3, pp. 65–72, 2008.

[39] G. Grisetti, D. Lodi Rizzini, C. Stachniss, E. Olson, and W. Burgard, "Online constraint network optimization for efficient maximum likelihood map learning," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1880–1885, 2008.

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

[41] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 62–69, 2013.

[42] T. Pfeifer, S. Lange, and P. Protzel, "Dynamic Covariance Estimation - A parameter free approach to robust Sensor Fusion," *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, vol. 2017-Novem, pp. 359–365, 2017.

[43] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Information fusion in navigation systems via factor graph based incremental smoothing," *Robotics and Autonomous Systems*, vol. 61, no. 8, pp. 721–738, 2013.

Glossary

List of Acronyms

SLAM	Simultaneous Localisation and Mapping

List of Symbols

This list describes all symbols that are used beyond a first appearance and description, and outside the section of introduction.

General

$\mathcal{L}(\mathcal{X}; \mathcal{Z})$	Likelihood of $\mathcal{X}$ given $\mathcal{Z}$; $\mathcal{L}(\mathcal{X}; \mathcal{Z}) \propto p(\mathcal{Z} \mid \mathcal{X})$.
n	Dimensionality of the SLAM problem; $n \in \{2, 3\}$ for either 2D or 3D, respectively.

Lie theory

$(\cdot)^\vee$	'Vee' decorator; denotes the identifying vector corresponding to a Lie algebra element.
$(\cdot)^\wedge$	'Hat' decorator; denotes the Lie algebra element corresponding to its identifying vector.
Ad_x	Adjoint $\mathrm{Ad}_x : \mathfrak{g} \to \mathfrak{g}$ of $\mathcal{G}$ at $x \in \mathcal{G}$.
$\circ$	Group composition action.
Exp	Capitalised exponential map; $\mathrm{Exp} : \mathbb{R}^{n_\mathcal{G}} \to \mathcal{G}$ maps the identifying vector of an element of the Lie algebra $\mathfrak{g}$ to an element of the corresponding Lie group $\mathcal{G}$.
exp	Exponential map; $\exp : \mathfrak{g} \to \mathcal{G}$ maps an element of the Lie algebra $\mathfrak{g}$ to an element of the corresponding Lie group $\mathcal{G}$.
Log	Capitalised logarithmic map; $\log : \mathcal{G} \to \mathbb{R}^{n_\mathcal{G}}$ maps an element of the Lie group $\mathcal{G}$ to an element of the identifying vector of the corresponding Lie algebra $\mathfrak{g}$.
log	Logarithmic map; $\log : \mathcal{G} \to \mathfrak{g}$ maps an element of the Lie group $\mathcal{G}$ to an element of the corresponding Lie algebra $\mathfrak{g}$.

$\mathcal{G}$	General Lie group variable.
$\mathfrak{g}$	General Lie algebra variable.
$\ominus$	Minus operator; $\ominus : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^{n_\mathcal{G}}$ finds the difference between two Lie group elements, expressed as the identifying vector of the local tangent plane.
$\oplus$	Plus operator; $\oplus : \mathcal{G} \times \mathbb{R}^{n_\mathcal{G}} \to \mathcal{G}$ finds the Lie group element, obtained by the addition of an increment that is expressed as the identifying vector of local tangent plane.
e	Identity element of a Lie group.
$E^i_\mathcal{G}$	Generator of Lie algebra $\mathfrak{g}$ of $\mathcal{G}$, indexed by $i \in \{1, \ldots, n_\mathcal{G}\}$.
$T_g\mathcal{G}$	Tangent vector space of Lie group $\mathcal{G}$ at group element g.

Lie groups

$\mathcal{SE}(n)$	Special Euclidean group of rigid body transformations with dimension n.
$\mathcal{SO}(n)$	Special Orthogonal group of rotations with dimension n.
$\boldsymbol{\theta}$	Identifying vector of a rotation matrix; $\boldsymbol{\theta} \in \mathbb{R}^3 \cong \mathfrak{so}(3)$ and $\theta \in \mathbb{R} \cong \mathfrak{so}(2)$
$\boldsymbol{\xi}$	Identifying vector of a rigid transformation matrix; $\boldsymbol{\xi} = (\mathbf{s}, \boldsymbol{\theta}) \in \mathbb{R}^6 \cong \mathfrak{se}(3)$ and $\boldsymbol{\xi} = (\mathbf{s}, \theta) \in \mathbb{R}^3 \cong \mathfrak{se}(2)$
$\mathbf{t}_\mathsf{B}^{(\mathsf{A})}$	Translation vector of B relative to A; $\mathbf{t}_\mathsf{B}^{(\mathsf{A})} \in \mathbb{R}^n$.
$R_\mathsf{B}^{(\mathsf{A})}$	Rotation matrix of B relative to A; $R_\mathsf{B}^{(\mathsf{A})} \in \mathcal{SO}(n)$.
$T_\mathsf{B}^{(\mathsf{A})}$	Transformation matrix of B relative to A; $T_\mathsf{B}^{(\mathsf{A})} \in \mathcal{SE}(n)$.

SLAM

ℓ_id	Relative sensor frame of sensor with identifier 'id'; $\ell_\text{id} = T_{\mathsf{S}^\text{id}}^{(\mathsf{R})}$.
O	Fixed global (or inertial) reference frame.
R_t	Robot-fixed reference frame at time t.
S^id	Sensor-fixed reference frame, with sensor identifier 'id'.
$\mathcal{C}$	Set of all constraint covariances.
$\mathcal{I}$	Set of pairs of pose indices for which constraints are defined.
$\mathcal{P}$	Set of all calibration parameters.
$\mathcal{S}$	Set of all constraint-type identifiers.
$\mathcal{X}$	Set of all robot poses.
$\mathcal{Z}$	Set of all measurements.
$\mathbf{t}_t$	Translation component of pose x_t; $\mathbf{t}_t \in \mathbb{R}^n$.
R_t	Rotation component of pose x_t; $R_t \in \mathcal{SO}(n)$.
x_t	Robot pose at time t; $x_t = T_{\mathsf{R}_t}^{(\mathsf{O})} \in \mathcal{SE}(n)$.
$z_{i,j}$	Measurement of relative pose transformation between poses x_i and x_j; $z_{i,j} = T_{\mathsf{R}_j}^{(\mathsf{R}_i)} \in \mathcal{SE}(n)$ such that $x_i z_{i,j} = x_j$.