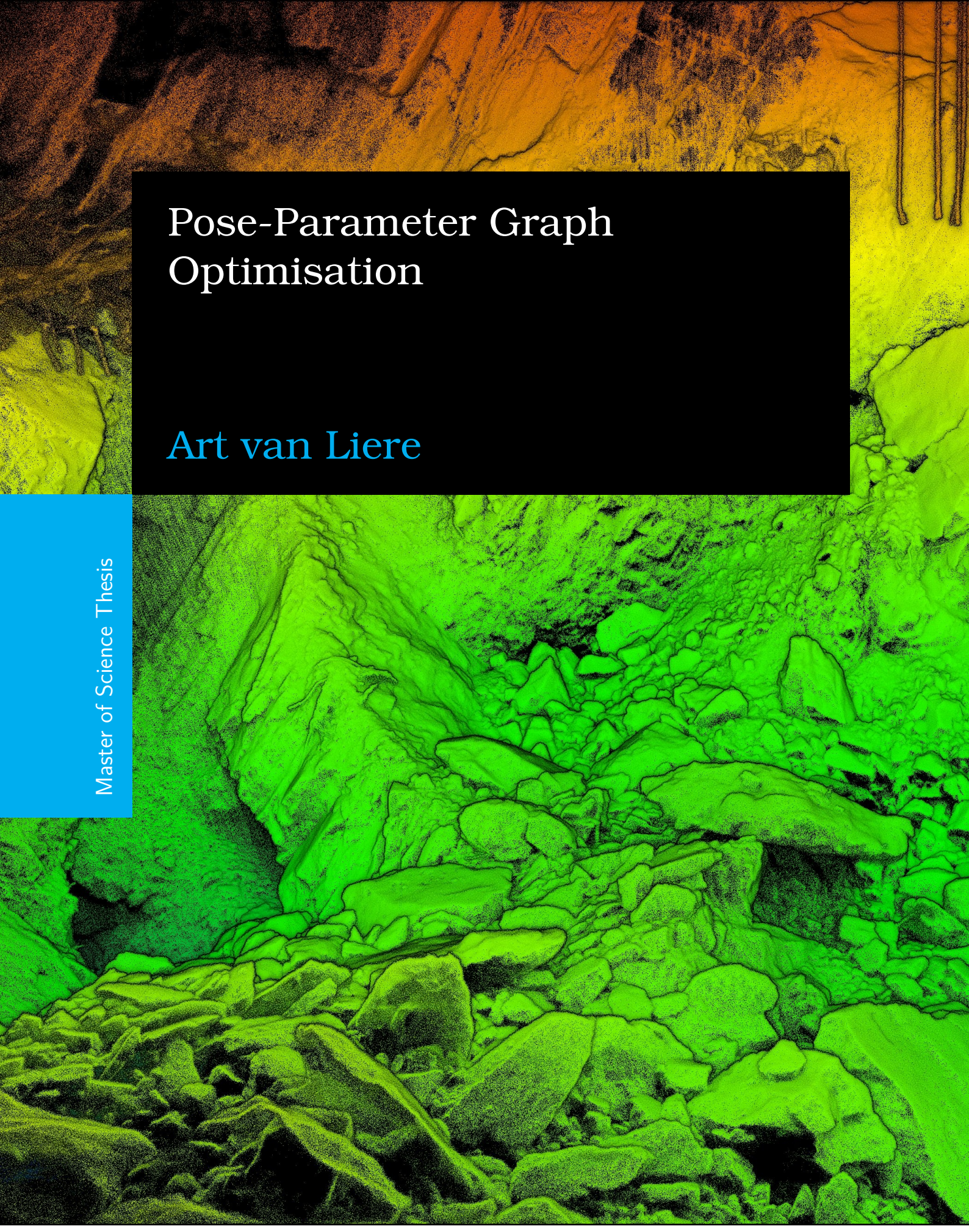


Pose-Parameter Graph Optimisation

Art van Liere

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Pose-Parameter Graph Optimisation

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Art van Liere

5th November 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

Pose Graph Optimisation (PGO) is a technique that is used to solve the Simultaneous Localisation and Mapping (SLAM) problem by relying on least-square minimisation techniques to find the most likely set of robot poses (i.e., location and orientation) given the set of measurements. It formalises the maximum likelihood formulation using a *pose graph*, where poses are represented as nodes, and measurements are represented as edges. Each measurement is paired with a *measurement model* that maps the related poses to an expected measurement value. PGO tries to find the set of poses that minimises the difference between the expected measurement values and the true measurement values. An accurate solution requires accurate measurement models. Models can be dependent on knowledge of specific robot parameters, or may fail to account for unknown systematic measurement deviations. By identifying the dependency of the unknown parameter within the measurement model, the parameter can be added as a graph node, thereby creating a *pose-parameter graph* and the accompanying *Pose-Parameter Graph Optimisation (PPGO)* problem.

In this thesis, a generalised approach to PPGO is proposed that is based on the implementation of two generally applicable parameters (a *bias* and *scaling factor*). Each parameter implementation defines *modified measurement models* that relate the newly defined parameter-nodes. *Connectivity strategies* are proposed that connect the set of parameter-nodes within the pose-parameter graph based on the nature of the parameter fluctuation.

For this thesis, a framework was developed in `python` that uses `g2o` to solve the PPGO problem. A GUI was designed to intuitively visualise graph components and evaluate performance metrics. The estimation of the bias parameter is found to be reliable over all measurement components, whereas the scaling factor only allows for reliable estimation over measurement components that exhibit consistent non-zero values. The *static* connectivity strategy can be reliably utilised to estimate an unknown constant parameter value. The *sliding window* and *timely batch* strategies are able to reconstruct a sinusoidal parameter with time, with the former offering an accurate instantaneous estimate with a slight delay, and the latter offering higher accuracy when applied as a post-processing step. The *spatial batch* strategy is able to reconstruct a sinusoidal parameter with space by creating a matching parameter heat map.

Contents

List of Figures

List of Tables

Acknowledgements

This thesis marks the end of my seven-year journey at TU Delft. During this amazing time, I have studied at three faculties, participated in three student teams, spent a half year in and around Singapore to study at NTU, and have visited SpaceX to compete in its Hyperloop competition. With this thesis, I conclude my years as a student, and I look forward to what's to come!

First, I would like to thank my supervisor, Tamás Keviczky, for his supervision and guidance in this final stage of the System and Control master's programme.

Second, I would like to thank my daily supervisor, Martijn Krijnen, for always being supportive and critical throughout this thesis. I valued your insights and in-depth feedback during our weekly meetings, which helped me shape my research into what it is now. It was great to be able to converse with you about the complex topics I sometimes got lost in. In addition, I would like to thank Demcon for facilitating this research.

Third, I would like to thank my girlfriend, Emily, for being there when I needed it most. Without your support and affection, this pandemic-struck year would not have been such a joyful period of my life. Also, I would like to thank my friends for keeping me grounded and reminding me to have fun.

Last, but not least, I would like to thank my family for their constant support over the past years. It was a long journey and I am glad you were a part of it.

Delft, University of Technology
5th November 2021

Art van Liere

Chapter 1

Introduction

A problem central to mobile robotics is that of *navigation*. Mobile robots are not fixed to one physical location and have the capability to move around in their environment. Ideally, a mobile robot should have the capacity to autonomously avoid dangerous conditions (such as collisions) and safely reach target destinations. This robust form of navigation requires the robot to have a sense of spatial awareness, and this is what the navigation problem is trying to solve.

Being able to build a map of the environment and to simultaneously localise within this map is an essential skill for mobile robots navigating in unknown environments in absence of external referencing systems. This problem is referred to as the Simultaneous Localisation and Mapping (SLAM) problem [8, 1, 10, 2], which comprises the simultaneous estimation of the state of a robot and the construction of a model (or *map*) of the environment using data from on-board sensors. In simple instances, the robot state is described by its *pose*: a position and orientation (i.e., a translation and rotation).

The graph-based formulation of the *full SLAM problem* [32] aims not only to estimate the current pose (in addition to the map), but rather the entire path. *Pose Graph Optimisation (PGO)*, in particular, aims to solve the SLAM problem by finding the most likely trajectory (represented by a set of poses) given the set of measurements. Due to influences of measurement and/or process noise, the set of (relative) pose transformation measurements available for this purpose are not necessarily mutually congruent. The maximum likelihood formulation is formalised using a *pose graph*, where robot poses are represented by nodes and the measurements are encoded in its edges. By representing the nodes in space according to the corresponding translation value, the pose graph functions as an intuitive visualisation of the robot trajectory, as shown in Figure 1-1.

Each measurement is paired with a *measurement model* that maps the related poses to an expected measurement value, thereby establishing a relationship between the set of poses and the set of measurements. PGO relies on least-square minimisation techniques to find a solution that maximises the likelihood of the set of poses given the set of measurements based on the mismatch between the expected and true measurement value over all measurements.

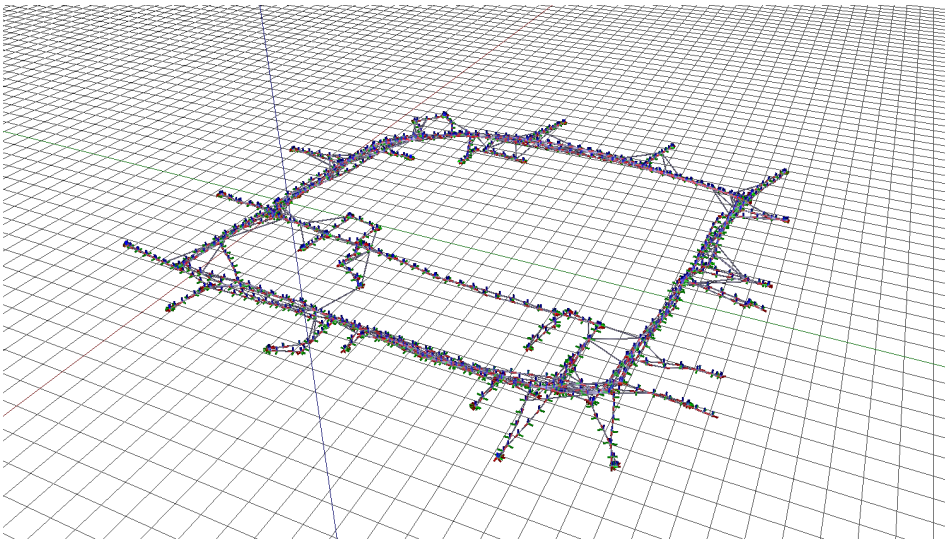


Figure 1-1: Pose graph visualisation of a dataset acquired at Intel Research Lab in Seattle [4].

This solution is defined in terms of the node configuration of the pose graph.

PGO relies on the measurement models that are encoded in the pose graph to be accurate. Oftentimes, these measurement models are dependent on knowledge of specific robot parameters that are to be set a priori, such as e.g. wheel radii or sensor coordinate frame transformation. However, such parameters could be hard to measure, require the use of lengthy ad-hoc calibration procedures, or be simply subject to change, promptly rendering any initial estimate useless [31, 6, 5, 21]. Other times, the observation of a systematic measurement deviation requires the model to be modified to include an unknown parameter based on an assumed dependency relation — take, for instance, an unanticipated sensor bias or unaccounted for kinematic parameter of the odometry system.

For the purpose of estimating these parameters without modifying the physical robot configuration, calibration techniques exist that rely only on data gathered by the robot [17, 22]. These techniques include the unknown parameters in the set of nodes, thereby creating a graph that contains both pose-nodes and parameter-nodes. In this research, such a graph will be referred to as a *pose-parameter graph*. Accordingly, the measurement models are extended to include a dependency on the set of parameter-nodes. Finding the maximum likelihood solution will be referred to as the *Pose-Parameter Graph Optimisation (PPGO)* problem.

1-1 Motivation

Although PPGO is easily described by its extended set of nodes (consisting of pose-nodes and parameter-nodes), the implementation of the extra node-type is not trivial. With a generalised approach not readily available, and most approaches relying on custom implementations [6, 5], the need arises for *a generalised approach to PPGO that is applicable to a variety of commonly occurring use-cases.*

Whereas PGO is able to provide an estimate for the robot pose for every time instance, the

extension to PPGO does not enable the same for the parameters. Because the parameter values are not directly measured, but rather deduced from inconsistencies between the measurement models and actual measurement values, the addition of a unique parameter-node for each time instance would result in an under-constrained optimisation problem. Therefore, a different connectivity strategy is required that is dependent on the definition of the measurement model, the nature of the underlying parameter value, and the performance that is sought to be achieved.

This thesis aims to serve as a starting guide for implementing PPGO for a variety of commonly occurring use-cases. For this purpose, a set of generally applicable basis measurement models is defined and its behaviour investigated in a variety of scenarios. These basis parameters comprise an additive *bias* parameter and a multiplicative *scaling factor* parameter.

1-2 Research objective

The objective of this research is formulated using the following set of research questions:

- *Which types of parameters can be estimated with the use of Pose-Parameter Graph Optimisation?*

  Although the modification to the measurement models to include a dependence on the basis parameters is rather straightforward, whether such a PPGO problem is able to find a *correct* cost-optimal solution is not so. The primary goal of the graph optimisation techniques employed to solve the SLAM problem is to approximate the ground truth, irrespective of the cost value associated with the cost-optimal solution. As such, the cost-optimal solution is only deemed correct if approximates the ground truth.

- *How does the performance of Pose-Parameter Graph Optimisation compare with Pose Graph Optimisation under appropriate conditions?*

  It is important to compare multiple viable implementations with the PGO counterpart for every use-case because the implementation of PPGO depends on the connectivity strategy employed. The connectivity strategy in turn depends on the fluctuation of the underlying parameter and the performance sought.

- *Under what conditions is Pose-Parameter Graph Optimisation able to capture the dynamics of a parameter?*

  The nature of the dynamics of the true underlying parameter also influences the implementation. These dynamics can be assigned to either be subjected to a *temporal correlation* or a *spatial correlation*, each of which stipulates its corresponding connectivity strategy. As such, the following sub-questions are formulated:

  - *What insights can be gained by modelling a parameter's temporal correlation using the sliding window and timely batch connectivity strategies?*

    For the purpose of modelling a parameter subjected to a temporal correlation, two connectivity strategies are proposed: the *timely batch* strategy and the *sliding window* strategy.

  - *What insights can be gained by modelling a parameter's spatial correlation using the spatial batch strategy?*

For the purpose of modelling a parameter subjected to a spatial correlation, the *spatial batch* connectivity strategy is proposed.

1-3 Contributions

- *A first-principles derivation of the Pose-Parameter Graph Optimisation problem formulation, which includes the implementation of the additive bias parameter and the multiplicative scaling factor parameter.*
- *A novel formulation of a dynamic sliding window approach for the modelling of parameters with a temporal correlation.*
- *A novel formulation of a spatial batch approach for the modelling of parameters with a spatial correlation.*
- *A simulation framework that provides an intuitive and easily extendable Application Programming Interface (API) to generate and optimise pose-parameter graphs.*
- *A pose-parameter graph analysis framework with a Graphical User Interface (GUI) that intuitively visualises all graph components and provides a means of evaluating performance criteria.*
- *A modification to the* `g2o` *framework [11], which allows for PPGO to be used as of now in any robot with a SLAM framework that makes use of* `g2o`.

1-4 Outline

This thesis consists of the following six chapters:

- Chapter 1 *'Introduction'* gives an overview of this research by introducing its motivation, formulating the research objective, and listing the contributions. It introduces Pose-Parameter Graph Optimisation, which is an extension of Pose Graph Optimisation, and establishes the need for a generalised approach that is applicable to a variety of commonly occurring use-cases.
- Chapter 2 *'Pose graph optimisation'* derives the Pose Graph Optimisation problem formulation while introducing notation and performance metrics. The notation is necessary for the derivation of Pose-Parameter Graph Optimisation in Chapter 3. Furthermore, the performance metrics are used in the performance assessment in Chapter 5.
- Chapter 3 *'Parameter calibration'* derives the Pose-Parameter Graph Optimisation problem formulation, defines the measurement models for the basis parameters, and introduces the connectivity strategies. The combination of measurement model and connectivity strategy constitutes the parameter implementation.
- Chapter 4 *'Simulation framework'* introduces the simulation framework and trajectories that are used to generate the result of Chapter 5.
- Chapter 5 *'Results'* presents the results of the considered parameter implementations under the influence of constant parameters and dynamic parameters with both a temporal correlation and a spatial correlation.

- Chapter 6 *'Conclusion'* concludes the thesis by stacking up the results with the research objective and suggesting the work that is left for future research.

The outline of this thesis is visualised in Figure 1-2.

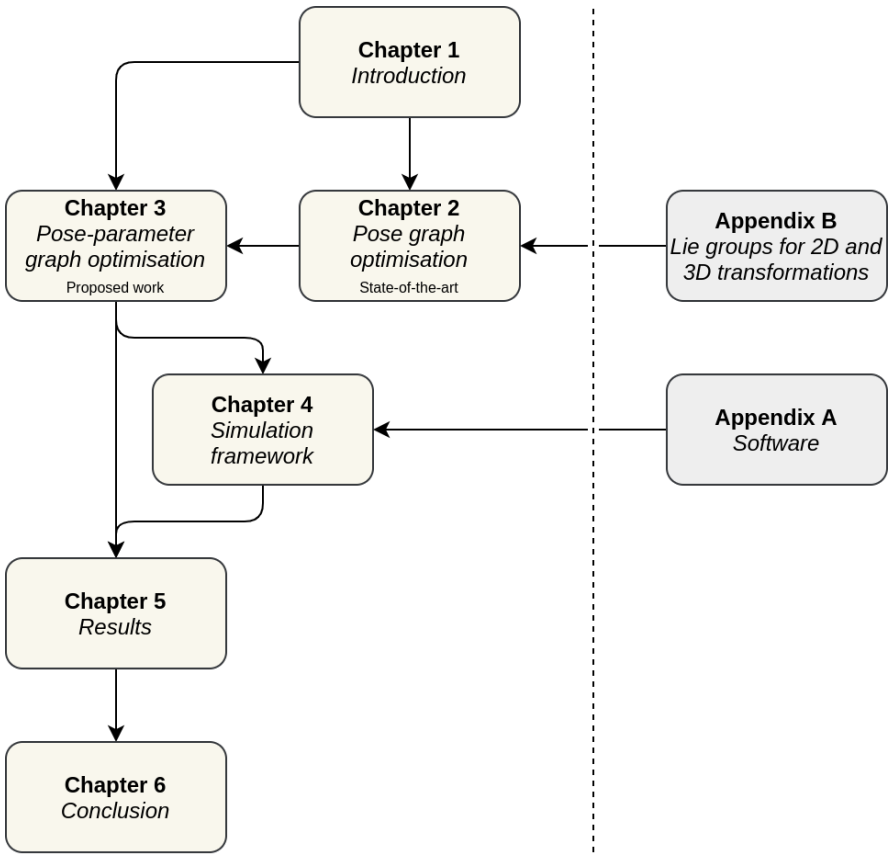


Figure 1-2: Thesis outline

Chapter 2

Pose graph optimisation

Pose Graph Optimisation (PGO) is a technique that addresses the full SLAM problem by finding the most likely trajectory (represented by a set of poses) given the set of measurements [10, 3, 2]. PGO forms the basis from which Pose-Parameter Graph Optimisation (PPGO) is derived, as will be discussed in Chapter 3. This chapter provides an extensive derivation of the PGO problem and introduces the notation necessary for the extension to PPGO.

PGO utilises a *pose graph* to formalise maximum likelihood formulation of a set of robot poses given the set of measurements. In a pose graph, poses are represented by nodes and measurements are encoded in edges. Due to influences of measurement and/or process noise, the set of measurements is likely to be incongruent. As such, PGO relies on least-square minimisation techniques to find the solution that maximises the likelihood over all measurements.

This chapter covers the following:

- Section 2-1 *'Notation'* introduces the necessary notation used to derive the PGO problem formulation. Furthermore, it introduces the mathematical objects used to express poses and rotations, known as *Lie groups*, alongside the corresponding Lie theory.
- Section 2-2 *'Problem formulation'* derives the PGO problem formulation as a least-square minimisation problem. It introduces the formalisation of the maximum likelihood formulation of the set of robot poses given the set of measurements over a special factor graph called a pose graph.
- Section 2-3 *'Constraints'* introduces the measurement models that map the connecting poses to the expected measurement value, which are used to establish a relationship between the set of poses and the set of measurements in the form of graph constraints.
- Section 2-4 *'Congruence'* introduces the concept of graph congruence, which describes an incomplete graph due to the lack of conflicting measurements.
- Section 2-5 *'Linearisation'* linearises the PGO problem and introduces the quadratic form.
- Section 2-6 *'Error metrics'* introduces the error metrics that use the ground truth solution to assess the performance of the estimate solution.
- Section 2-7 *'Summary'* provides a summary of the introduced material.

2-1 Notation

This section introduces the notation that is used to derive the PGO problem formulation, which comprises the definition of relevant reference frames, the definition of poses and measurements, and the sets of interest.

2-1-1 Reference frames

Reference frames are used to express relative transformations, translations, and rotations. Each is defined as a right-handed Cartesian coordinate frame and is denoted by in Sans-serif typeface. The expression of one (reference) frame A in another frame B is defined by its rigid body transformation $T_{\mathsf{A}}^{(\mathsf{B})}$, which comprises the relative translation of the origins $\mathbf{t}_{\mathsf{A}}^{(\mathsf{B})}$ and relative rotation of the axes $R_{\mathsf{A}}^{(\mathsf{B})}$. The specifics of these mathematical objects are discussed in Section 2-1-2. The following reference frames are defined:

1. The *fixed global* (or *inertial*) *reference frame* O (where 'O' refers to the 'global origin') is typically fixed at the robot's starting position.
2. The *robot-fixed reference frame* R is fixed to the robot's *odometric centre* (i.e., at the centre of rotation, with the x-axis pointing along the axis of forward motion and the z-axis pointing up). Due to the movement of the robot, the relative transformation of R with respect to O changes with time. To highlight this time-dependence and indicate which version of the robot-fixed frame is referred to, the relative transformation is denoted by $T_{\mathsf{R}_t}^{(\mathsf{O})} := T_{\mathsf{R}}^{(\mathsf{O})}(t)$.
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 transformation $\ell_{\text{id}} := T_{\mathsf{S}^{\text{id}}}^{(\mathsf{R})}$ is considered a constant. On the other hand, the transformation $T_{\mathsf{S}_t^{\text{id}}}^{(\mathsf{O})} := T_{\mathsf{S}^{\text{id}}}^{(\mathsf{O})}(t)$ is a function of time.

2-1-2 Poses

As discussed in Section 2-1-1, the transformation $T_{\mathsf{R}_t}^{(\mathsf{O})}$ of body-fixed reference frame with respect to the inertial frame comprises a *translation* and *rotation*. This rigid body transformation is what is implied by the robot *pose*. Robot poses are denoted by x_i, where i denotes the node index and/or time instance. Furthermore, $\mathcal{X} = \{x\}$ denotes the set of all robot poses.

At first sight, one would expect such a transformation to be defined in vector space, given by $T_{\mathsf{R}_t}^{(\mathsf{O})} = (x, y, \theta) \in \mathbb{R}^3$, where $x, y, \theta \in \mathbb{R}$ are the x-translation, y-translation and planar rotation components, respectively. However, the expression of a planar rotation using a scalar angle $\theta \in \mathbb{R}$ does not respect the circle topology; that is, the wrap-around property of angles that stipulates that $\theta \in [0, 2\pi)$ or $\theta \in [-\pi, \pi)$. This can be solved by using rotation matrices to represent rotations, where the compounding of multiple rotation matrices by matrix multiplication results in another rotation matrix. This has the advantage that no additional steps are required to make rotation matrices respect the required circle topology. The special structure of rotation matrices is described by a rotation manifold, which can be expressed using Lie theory.

Lie theory

Rotations and transformations are conveniently described by *matrix Lie groups*, which are thoroughly examined in Appendix B. *Lie theory* (and the accompanying *Lie groups*) can be summarised with the following bullet points:

- A Lie group $\mathcal{G}$ is a smooth manifold whose elements satisfy the group axioms [28].
- The associated Lie algebra $\mathfrak{g}$ is a vector space that can be considered a linearisation of the Lie group. It has the same dimension $n_{\mathcal{G}}$ as the number of degrees of freedom of $\mathcal{G}$; any element can therefore be identified by vector $\boldsymbol{\tau} \in \mathbb{R}^{n_{\mathcal{G}}}$.
- There exists a mapping (called the exponential map) that provides the 'de-linearisation' from $\mathfrak{g}$ to $\mathcal{G}$. As a result, the vector $\boldsymbol{\tau} \in \mathbb{R}^{n_{\mathcal{G}}}$ that identifies the Lie algebra, efficiently parameterises the Lie group element $g \in \mathcal{G}$. This is indicated by $\boldsymbol{\tau} \cong g \in \mathcal{G}$, with $\boldsymbol{\tau} \in \mathbb{R}^{n_{\mathcal{G}}} \cong \mathcal{G}$.

To aid in the manipulation of Lie group elements (e.g. for defining measurement models in Section 2-1-3), the following operators are introduced:

- The *plus* operator $\oplus$ defines a homomorphism that is used to find the updated group element $z \in \mathcal{G}$ from $x \in \mathcal{G}$, obtained by composition with a group element $y \in \mathcal{G}$ identified by $\mathbf{y} \in \mathbb{R}^{n_{\mathcal{G}}}$. The homomorphism is defined by
$$\oplus : \mathcal{G} \times \mathbb{R}^{n_{\mathcal{G}}} \to \mathcal{G} : \quad x \times \mathbf{y} \mapsto z \in \mathcal{G},$$
with $x\, y = z$. This basically allows for incrementing a Lie group element by a vector and returning another group element, which is useful in e.g. iterative optimisation.
- Similarly the *minus* operator $\ominus$ can be thought of as the inverse of $\oplus$ (i.e., 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 and expresses it as a vector:
$$\ominus : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^{n_{\mathcal{G}}} : \quad z \times x \mapsto \mathbf{y} \in \mathbb{R}^{n_{\mathcal{G}}},$$
with $x^{-1}\, z = y$.

Rotations

The orientation of a body is described by the relative rotation 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 relative to A.

For the 2-dimensional case, it is obtained by stacking the unit coordinate vectors of the principal axes $\mathbf{x}_{\mathsf{B}}^{(\mathsf{A})}, \mathbf{y}_{\mathsf{B}}^{(\mathsf{A})} \in \mathbb{R}^2$ of B as seen from A; that is,
$$R_{\mathsf{B}}^{(\mathsf{A})} = \begin{bmatrix} \mathbf{x}_{\mathsf{B}}^{(\mathsf{A})} & \mathbf{y}_{\mathsf{B}}^{(\mathsf{A})} \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \in \mathcal{SO}(2),$$
where rotations matrices are represented by the *Special Orthogonal* group $\mathcal{SO}(n)$. *'Special'* refers to unit determinant ($\det(R) = 1$), and *'Orthogonal'* refers to the orthogonality constraint ($R^\top R = I_n$). The rotation group is defined as
$$\mathcal{SO}(n) = \{R \in \mathbb{R}^{n \times n} : R^\top R = I_n, \det(R) = 1\}.$$

The planar rotation angle θ in radians parameterises the rotation group element as

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \in \mathcal{SO}(2),$$

which is indicated by $\theta \cong R(\theta)$, with $\theta \in \mathbb{R} \cong \mathcal{SO}(2)$. This definition can be inverted to give $\theta = \tan^{-1}(R_{21}/R_{11})$.

Transformations

The instantaneous orientation and position of a body coordinate frame B relative to an inertial frame A is described using a rigid body transformation $T_{\mathsf{B}}^{(\mathsf{A})}$. For the 2-dimensional case, let $\mathbf{t}_{\mathsf{B}}^{(\mathsf{A})} \in \mathbb{R}^2$ be the translation vector of the origin of frame B relative to frame A, and $R_{\mathsf{B}}^{(\mathsf{A})} \in \mathcal{SO}(2)$ the orientation of frame B relative to A. The transformation $T_{\mathsf{B}}^{(\mathsf{A})}$ then comprises the pair $(\mathbf{t}_{\mathsf{B}}^{(\mathsf{A})}, R_{\mathsf{B}}^{(\mathsf{A})})$ defined over the *Special Euclidean Group* $\mathcal{SE}(2)$:

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

Alternatively, the Special Euclidean group can be defined as a homogeneous matrix representation [9]:

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

Let $\mathbf{q}$ be a point in 2-dimensional space that is expressed in the B frame as $\mathbf{q}^{(\mathsf{B})} \in \mathbb{R}^2$. To obtain the expression of $\mathbf{q}$ in the A frame, a coordinate transformation $T_{\mathsf{B}}^{(\mathsf{A})}$ is required. This coordinate change can be expressed as the affine transformation

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

where $\bar{T}_{\mathsf{B}}^{(\mathsf{A})} \in \mathbb{R}^{3\times3}$ is the *homogeneous matrix representation* of $T_{\mathsf{B}}^{(\mathsf{A})} \in \mathcal{SE}(2)$, and $\bar{\mathbf{q}}^{(\mathsf{A})}, \bar{\mathbf{q}}^{(\mathsf{B})} \in \mathbb{R}^3$ are the *homogeneous coordinates* of $\mathbf{q}^{(\mathsf{A})}, \mathbf{q}^{(\mathsf{B})} \in \mathbb{R}^2$.

The parameterisation of $\mathcal{SE}(n)$ that corresponds to Lie algebra $\mathfrak{se}(n)$ is given in Appendix B. For the 2-dimensional case, however, it is more intuitive to use the identifying vector $\boldsymbol{\xi} = (x, y, \theta) \in \mathbb{R}^3$, which is simply the concatenation of translation $\mathbf{t} = (x, y)$ and the planar rotation angle θ. As such, any transformation group element is given by

$$T\big((x, y), R(\theta)\big) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix} \in \mathcal{SE}(n).$$

In summary, robot poses in two dimensions are represented by elements of the 2-dimensional transformation group $\mathcal{SE}(2)$, which are efficiently and intuitively parameterised by a vector $\boldsymbol{\xi} \cong T(\boldsymbol{\xi})$, with $\boldsymbol{\xi} = (x, y, \theta) \in \mathbb{R}^3 \cong \mathcal{SE}(2)$. That is, $x_i \in \mathcal{SE}(2)$ and $\mathcal{X} \subset \mathcal{SE}(2)$, where x_i comprises the rotation and translation components, $R_i \in \mathcal{SO}(2)$ and $\mathbf{t}_i \in \mathbb{R}^2$, respectively.

2-1-3 Measurements

In PGO, a measurement refers to a relative pose transformation, translation, or rotation, derived from sensor data. Measurements are denoted by $z_{\mathcal{I}}$, where $\mathcal{I}$ denotes the set of node indices $\mathcal{I} = \{i\}$. Furthermore, $\mathcal{Z} = \{z\}$ denotes the set of all measurements.

Measurements typically originate from two types of sensors:

- *Proprioceptive* sensor systems, which measure the robot's interaction with the environment, such as:
  - Rotary encoders on wheels to measure wheel rotation and derive a relative pose transformation estimates between consecutive poses $T^{(\mathsf{R}_t)}_{\mathsf{R}_{t+1}} \in \mathcal{SE}(2)$;
  - GPS sensors to measure a relative pose translations $\mathbf{t}^{(\mathsf{O})}_{\mathsf{R}_t}$;
  - Accelerometers, gyroscopes and magnetometers to derive estimates for relative pose transformations $T^{(\mathsf{R}_t)}_{\mathsf{R}_{t+1}} \in \mathcal{SE}(2)$, relative pose translations $\mathbf{t}^{(\mathsf{R}_t)}_{\mathsf{R}_{t+1}} \in \mathbb{R}^2$, and relative pose rotations $R^{(\mathsf{R}_t)}_{\mathsf{R}_{t+1}} \in \mathcal{SO}(2)$.
- *Exteroceptive* sensor systems, which directly measure the environment, such as:
  - (Depth) cameras and optical flow or scan matching algorithms to derive relative pose transformation estimates $T^{(\mathsf{R}_i)}_{\mathsf{R}_j} \in \mathcal{SE}(2)$;
  - Active ranging systems with LiDAR, sonar and radar and scan matching algorithms to derive relative pose transformation estimates $T^{(\mathsf{R}_i)}_{\mathsf{R}_j} \in \mathcal{SE}(2)$.

The measurements used in PGO are elements in $\mathcal{SE}(2)$, or its sub-components: translations in $\mathbb{R}^2$ and rotations in $\mathcal{SO}(2)$. That is, $z_{\mathcal{I}} \in \mathcal{SE}(2) \cup \mathcal{SO}(2) \cup \mathbb{R}^2 = \mathcal{M}$.

As an abstraction for the different measurement types, the generalisation addition and subtraction operators are introduced, which are denoted by '$\boxplus$' and '$\boxminus$', respectively. Both are defined as follows:

$$\boxplus : \mathcal{M} \times \mathbb{R}^m \to \mathcal{M}$$
$$\boxminus : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^m,$$

where m is the dimension of the identifying vector of the measurement. For the different measurement types, the generalised addition and subtraction operators are defined as:

- For transformation and rotation group elements, the generalised addition and subtraction parameters are equal to the addition plus operator '$\oplus$' and minus operator '$\ominus$', respectively.
- For vectors, the generalised addition and subtraction parameters are equal to the vector addition operator '$+$' and vector subtraction operator '$-$', respectively.

2-2 Problem formulation

This section introduces the PGO problem formulation as a least-square minimisation problem. It does so by first deriving the maximum likelihood formulation, then introducing the factor graph formalisation, and the Gaussian measurement model assumption. Finally, the least-square minimisation formulation is derived.

2-2-1 Probabilistic formulation

The SLAM problem is often formulated as a *Maximum a Posteriori (MAP)* problem [20, 2]. The MAP estimate is an estimate of an unknown quantity on the basis of empirical data. It maximises the posterior probability, which describes the probability of an unknown quantity, conditional on the evidence. Intuitively, the MAP estimate is the unknown quantity that is most probable, given the empirical data or evidence.

When applied to SLAM, the posterior probability describes the probability of the set of robot poses $\mathcal{X}$ given the set of measurements $\mathcal{Z}$. It is given by

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

and contains the following elements:

- $\mathcal{L}(\mathcal{X}; \mathcal{Z})$ denotes the likelihood of $\mathcal{X}$ given $\mathcal{Z}$, and is defined as any function proportional to $p(\mathcal{Z} \mid \mathcal{X})$;
- $p(\mathcal{X})$ denotes the prior belief of $\mathcal{X}$.

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 of poses $\mathcal{X}$ 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. As a result, the solution to the full SLAM problem is given by

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} \mathcal{L}(\mathcal{X}; \mathcal{Z}). \tag{2-1}$$

Assuming the measurements $\mathcal{Z}$ are independent (i.e., the corresponding measurement noise is uncorrelated), the maximum likelihood formulation of (2-1) formalises into

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} \prod_{\mathcal{I}} p(z_{\mathcal{I}} \mid \mathcal{X}_{\mathcal{I}}). \tag{2-2}$$

2-2-2 Gaussian noise assumption

Measurements are subject to uncertainties. In many cases, it is both convenient and justified to model measurements as corrupted by additive zero-mean Gaussian noise [29]. The measurement $z_{\mathcal{I}}$ is then modelled as

$$z_{\mathcal{I}} \boxplus \boldsymbol{\epsilon} = f_{\mathcal{I}}(\mathcal{X}),$$

where

- $f_{\mathcal{I}}(\mathcal{X}) := f_{\mathcal{I}}(\mathcal{X}_{\mathcal{I}}) : \mathcal{X} \to \mathcal{M}$ denotes the *measurement model function*, which maps the related poses $\mathcal{X}_{\mathcal{I}}$ to an *expected* measurement value in $\mathcal{M}$. It is assumed to be an unbiased estimator.

- $\boldsymbol{\epsilon} \in \mathbb{R}^{m_\mathcal{I}}$ denotes an $m_\mathcal{I}$-dimensional measurement noise vector, where $\boldsymbol{\epsilon} \sim \mathtt{N}(\mathbf{0}, \Sigma_\mathcal{I})$ is drawn from a zero-mean Gaussian distribution with *measurement model covariance* $\Sigma_\mathcal{I} \in \mathbb{R}^{m_\mathcal{I} \times m_\mathcal{I}}$.

Rewriting yields the measurement error vector function

$$\mathbf{e}_\mathcal{I}(\mathcal{X}) = f_\mathcal{I}(\mathcal{X}) \boxminus z_\mathcal{I} \in \mathbb{R}^{m_\mathcal{I}}.$$

The conditional probability density of the measurement $z_\mathcal{I}$ given the poses $\mathcal{X}_\mathcal{I}$ is given by the multivariate Gaussian distribution as

$$p(z_\mathcal{I} \mid \mathcal{X}_\mathcal{I}) = \frac{1}{\sqrt{(2\pi)^m |\Sigma_\mathcal{I}|}} \exp\Big(-\frac{1}{2} \|\mathbf{e}_\mathcal{I}(\mathcal{X})\|^2_{\Sigma_\mathcal{I}} \Big), \tag{2-3}$$

where

$$\|\mathbf{e}_\mathcal{I}(\mathcal{X})\|^2_{\Sigma_\mathcal{I}} := \mathbf{e}_\mathcal{I}^\top(\mathcal{X})\, \Sigma_\mathcal{I}^{-1}\, \mathbf{e}_\mathcal{I}(\mathcal{X}) \in \mathbb{R}$$

denotes the squared Mahalanobis distance.

2-2-3 Factor graph

Problem (2-2) can be interpreted as inference over a factor graph [7, 16]. This model provides a mechanism for compactly describing the complex probability density function, such as given in (2-2).

A factor graph is a bipartite graph representing the factorisation of a function. The graph is denoted $g = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ and contains elements of the following types:

- Variable nodes $v_i \in \mathcal{V}$, which represent the function variables;
- Factor nodes $u_i(\mathcal{V}_i) \in \mathcal{U}$, which represent factorisation of the connecting variable nodes;
- Edges $e_{i,j} \in \mathcal{E}$, which connects factor nodes i to the the dependent variable nodes j.

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

$$f(\mathcal{V}) = \prod_i u_i(\mathcal{V}_i).$$

The factor graph and its factorisation can utilised to factorise the likelihood of robot poses $\mathcal{X}$ given measurements $\mathcal{Z}$ by taking:

- The set of robot poses $\mathcal{X}$ to be the variable nodes;
- The set of conditional measurement probability densities $\{p(z_\mathcal{I} \mid \mathcal{X}_\mathcal{I})\}$ to be equal to the factor nodes.

By only considering robot poses as variables nodes, the factor graph is classified a *pose graph* [3]. The pose graph encodes the dependence relation of the measurements with the corresponding robot poses: each factor node encodes the posterior probability of the connecting robot poses given the observed measurement. Not only does the factor graph offer an insightful visualisation of the robot path, but its connectivity also defines the sparsity of the resulting SLAM problem.

Section 2-4 introduces an example of a pose graph with Figure 2-1.

2-2-4 Least-square error minimisation

By combining maximum likelihood formulation derived in (2-2) (which is identical to what is formalised by the factor graph factorisation) and the conditional measurement probability density under the Gaussian noise assumption described in (2-3), the MAP estimate is written as

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} \prod_{\mathcal{I}} \left[\frac{1}{\sqrt{(2\pi)^m |\Sigma_{\mathcal{I}}|}} \exp\Big(-\frac{1}{2} \|\mathbf{e}_{\mathcal{I}}(\mathcal{X})\|^2_{\Sigma_{\mathcal{I}}} \Big) \right].$$

By taking the negative log-likelihood of this expression, the product maximisation problem turns into a summation minimisation problem, and is written as

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_{\mathcal{I}} \left[\underbrace{\ln \sqrt{(2\pi)^m |\Sigma_{\mathcal{I}}|}}_{\text{const.}} + \frac{1}{2} \|\mathbf{e}_{\mathcal{I}}(\mathcal{X})\|^2_{\Sigma_{\mathcal{I}}} \right].$$

The measurement covariance is independent of the solution $\mathcal{X}$. Therefore, the first term for every conditional measurement density can be dropped [26]. By also dropping the factor $1/2$, the pose graph optimisation problem is written as a least-square minimisation problem

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_{\mathcal{I}} \|\mathbf{e}_{\mathcal{I}}(\mathcal{X})\|^2_{\Sigma_{\mathcal{I}}}.$$

Each measurement contributes a scalar quadratic cost term

$$\|\mathbf{e}_{\mathcal{I}}(\mathcal{X})\|^2_{\Sigma_{\mathcal{I}}} = \left(f_{\mathcal{I}}(\mathcal{X}) \boxminus z_{\mathcal{I}}\right)^\top \Sigma_{\mathcal{I}}^{-1} \left(f_{\mathcal{I}}(\mathcal{X}) \boxminus z_{\mathcal{I}}\right) \in \mathbb{R},$$

the structure of which has a few interesting consequences:

- The cost term is minimally zero when $f_{\mathcal{I}}(\mathcal{X}) = z_{\mathcal{I}}$; i.e. when the outcome of the measurement model perfectly matches the observed measurement value. More specifically, this occurs when the components of the related poses are configured such that the measurement model yields a value that is identical to what is observed by the sensors.
- The cost term that corresponds to each measurement scales quadratically with the error function $\mathbf{e}_{\mathcal{I}}(\mathcal{X}) = f_{\mathcal{I}}(\mathcal{X}) \boxminus z_{\mathcal{I}}$ that compares the expected measurement value with the true measurement value. This function imposes a soft *constraint* on the connected nodes, which penalises potential solutions for not aligning with the measurement (or *constraint value*) and rewards poses that are placed close to what is dictated by the relevant measurements.
- Measurements that are deemed less accurate are characterised by a larger measurement model covariance (or *constraint covariance* matrix). Since each cost term is scaled inversely with the constraint covariance, inaccurate measurements result in weaker constraints and vice versa.

Each measurement contributes a cost term that imposes a constraint on the related poses. Accordingly, the cost-optimal solution to PGO is the set of poses that is a perfect match to all measurements, as this would yield the minimal zero cost function value. However, eventually, measurements are bound to conflict, even if it is by the tiniest margins due to measurement noise, so a cost-optimal compromise has to be found with respect to the pose configuration.

In summary, PGO seeks to find the most likely set of poses given the set of measurements. It relies on least-square minimisation techniques to find the cost-optimal MAP estimate that

minimises the weighted mismatch between the expected and true measurement values over all constraints. The PGO problem formulation is given by

$$\begin{aligned} \mathbf{e}_{\mathcal{I}}(\mathcal{X}) &= f_{\mathcal{I}}(\mathcal{X}) \boxminus z_{\mathcal{I}}, \\ F_{\mathcal{I}}(\mathcal{X}) &= \|\mathbf{e}_{\mathcal{I}}(\mathcal{X})\|^2_{\Sigma_{\mathcal{I}}}, \\ F_{\text{pgo}}(\mathcal{X}) &= \sum_{\mathcal{I}} F_{\mathcal{I}}(\mathcal{X}), \\ \mathcal{X}^* &= \arg\min_{\mathcal{X}} F_{\text{pgo}}(\mathcal{X}), \end{aligned} \tag{2-4}$$

where

- $\mathcal{I} = \{i\}$ denotes the set of pose identifiers that are related by a constraint;
- $\mathcal{X} = \{x\} \subset \mathcal{SE}(2)$ denotes the set of robot poses. Correspondingly, $\mathcal{X}_{\mathcal{I}} \subseteq \mathcal{X}$ denotes the subset of poses given by indices $\mathcal{I}$;
- $z_{\mathcal{I}} \in \mathcal{Z} \subset \mathcal{M}$ denotes the true measurement value that relates the poses identified by $\mathcal{I}$;
- $f_{\mathcal{I}}(\mathcal{X}) := f_{\mathcal{I}}(\mathcal{X}_{\mathcal{I}}) \in \mathcal{M}$ denotes the measurement model function, which maps the set of related poses $\mathcal{X}_{\mathcal{I}}$ to the expected measurement value;
- $\mathbf{e}_{\mathcal{I}}(\mathcal{X}) := \mathbf{e}_{\mathcal{I}}(\mathcal{X}_{\mathcal{I}}) \in \mathbb{R}^{m_{\mathcal{I}}}$ denotes the vector error function that describes the mismatch between the expected measurement value (as defined by the measurement model function) and the true measurement value. A perfect match yields $\mathbf{e}_i(\mathcal{X}) = \mathbf{0}$;
- $\Sigma_{\mathcal{I}} \in \mathbb{R}^{m_{\mathcal{I}} \times m_{\mathcal{I}}}$ denotes the $m_{\mathcal{I}}$-dimensional measurement model covariance (or constraint covariance) matrix of the constraint identified by $\mathcal{I}$;
- $F_{\mathcal{I}}(\mathcal{X}) := F_{\mathcal{I}}(\mathcal{X}_{\mathcal{I}}) \in \mathbb{R}$ denotes the scalar constraint cost term function, which is identical to the exponent of the measurement likelihood distribution and is given by the Mahalanobis distance of the vector error function and measurement model covariance.
- $F_{\text{pgo}}(\mathcal{X}) \in \mathbb{R}$ denotes the scalar PGO cost function that is the summation of the cost terms $\|\mathbf{e}_{\mathcal{I}}(\mathcal{X})\|^2_{\Sigma_{\mathcal{I}}}$ contributed by all constraints formalised in the pose graph, as a function of the set of nodes $\mathcal{X}$.

Note that simply scaling a cost function will not move its extrema. As such, only the *relative* values of the constraint covariances are of importance. In essence, these values act as weights that indicate the varying degree of importance (or accuracy) of each measurement. Therefore, the information that each measurement provides will be taken into consideration for optimisation based on the relative value of the accompanying constraint covariance value relative to all other constraints.

2-3 Constraints

As discussed in Section 2-2-4, each measurement contributes a cost term that imposes a *constraint* on the related poses, which forces the poses to (more or less) comply with the measurement. To achieve a low PGO cost function value, a set of poses is sought that minimises the weighted mismatch between the expected and true measurement value.

In this section, the constraint classifications are discussed and the measurement models are defined for some of the most common constraints used in PGO.

2-3-1 Classifications

A pose graph typically consists of two classifications of constraints:

1. *Odometry constraints*, which connect poses sequentially. Technically, odometry is the use of data from the movement of actuators to estimate the change in position over time; such as e.g. the use of rotary encoders to measure wheel rotation. However, these sensors and/or actuators might not always be available. Therefore, the definition of odometry is extended to include equivalent information. That is, any measurement of displacement between consecutive robot poses is considered an odometry measurement.

   Odometry constraints are typically the first to be established after instantiating a new pose. Odometry constraints can be introduced by both proprioceptive and exteroceptive sensor systems.

2. *Loop closure constraints*, which occur after a robot has returned to a previously visited location. By establishing this relationship, two distant poses are connected by a constraint and a portion of the pose graph is made cyclic, thereby *closing* a *loop*. In order to register whether this has happened, exteroceptive sensors are required.

2-3-2 Measurement models

For each of the measurements considered in PGO, a constraint is introduces that contributes an cost term based on the measurement value and the *measurement model*. The measurement model defines a *measurement model function* $f_{\mathcal{I}}$ and a *measurement model covariance* $\Sigma_{\mathcal{I}}$. The measurement model maps the set of related poses to an expected measurement value in $\mathcal{M}$.

The measurement models for some of the most common measurements are defined below:

- *Pose transformation* measurements $z_{i,j}^{\text{transf.}} \in \mathcal{SE}(2)$ relate two robot poses $x_i, x_j \in \mathcal{SE}(2)$, and approximate the relative transformation $T_{\mathsf{R}_j}^{(\mathsf{R}_i)} \in \mathcal{SE}(2)$ of the robot-fixed frame R_j in terms of R_i. Accordingly, the measurement model defines

$$f_{i,j}^{\text{transf.}}(\mathcal{X}) = T_{\mathsf{R}_j}^{(\mathsf{R}_i)}(x_i, x_j) \in \mathcal{SE}(2). \tag{2-5}$$

  As stated in Section 2-1-2, matrix Lie group elements are compounded using matrix multiplication. As such, a pose transformation $T_{\mathsf{R}_j}^{(\mathsf{R}_i)}$ can be 'added' onto a pose $x_i = T_{\mathsf{R}_i}^{(\mathsf{O})}$ by right-multiplication to yield pose $x_j = T_{\mathsf{R}_j}^{(\mathsf{O})}$ [7]:

$$x_i \, T_{\mathsf{R}_j}^{(\mathsf{R}_i)} = x_j \quad \Longrightarrow \quad T_{\mathsf{R}_j}^{(\mathsf{R}_i)} = x_i^{-1} \, x_j = f_{i,j}^{\text{transf.}}(\mathcal{X}). \tag{2-6}$$

  The closed homogeneous form of the relative pose transformation derived using the fact that rotation matrices are ortho-normal, which implies that their inverse is equal to their transpose:

$$\bar{T}_{\mathsf{R}_j}^{(\mathsf{R}_i)} = \bar{x}_i^{-1} \, \bar{x}_j = \begin{bmatrix} R_i^\top & -R_i^\top \mathbf{t}_i \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_j & \mathbf{t}_j \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_i^\top R_j & R_i^\top (\mathbf{t}_j - \mathbf{t}_i) \\ 0 & 1 \end{bmatrix} \in \bar{\mathcal{SE}}(2).$$

Finally, the error function $\mathbf{e}_{i,j}^{\text{transf.}\cdot}(\mathcal{X})$ of a relative pose transformation is defined as

$$\mathbf{e}_{i,j}^{\text{transf.}}(\mathcal{X}) = f_{i,j}^{\text{transf.}}(\mathcal{X}) \ominus z_{i,j}^{\text{transf.}} = x_i^{-1}\, x_j \ominus z_{i,j}^{\text{transf.}} \in \mathbb{R}^3. \tag{2-7}$$

- *Pose rotation* measurements $z_{i,j}^{\text{rot.}\cdot} \in \mathcal{SO}(2)$ relate two robot poses $x_i, x_j \in \mathcal{SE}(2)$, and approximate the relative transformation $R_{\mathsf{R}_j}^{(\mathsf{R}_i)} \in \mathcal{SO}(2)$ of the robot-fixed frame R_j in terms of R_i. Accordingly, the measurement model defines

  $$f_{i,j}^{\text{rot.}\cdot}(\mathcal{X}) = R_{\mathsf{R}_j}^{(\mathsf{R}_i)}(x_i, x_j) \in \mathcal{SO}(2).$$

  In similar fashion the case of pose transformations, the measurement model function and error function are derived as

  $$\begin{aligned} R_i\, R_{\mathsf{R}_j}^{(\mathsf{R}_i)} = R_j \quad &\Longrightarrow \quad R_{\mathsf{R}_j}^{(\mathsf{R}_i)} = R_i^\top\, R_j = f_{i,j}^{\text{rot.}\cdot}(\mathcal{X}), \\ \mathbf{e}_{i,j}^{\text{rot.}\cdot}(\mathcal{X}) = f_{i,j}^{\text{rot.}\cdot}(\mathcal{X}) &\ominus z_{i,j}^{\text{rot.}\cdot} = R_i^\top\, R_j \ominus z_{i,j}^{\text{rot.}\cdot} \in \mathcal{SO}(2). \end{aligned}$$

- *Pose translation* measurements $z_{i,j}^{\text{transl.}} \in \mathbb{R}^2$ relate two robot poses $x_i, x_j \in \mathcal{SE}(2)$, and approximate the relative translation $\mathbf{t}_{\mathsf{R}_j}^{(\mathsf{R}_i)} \in \mathbb{R}^2$ of the robot-fixed reference frame R_j in terms of R_i. Accordingly, the measurement model defines

  $$f_{i,j}^{\text{transl.}}(\mathcal{X}) = \mathbf{t}_{\mathsf{R}_j}^{(\mathsf{R}_i)}(x_i, x_j) \in \mathbb{R}^2.$$

  Composition of Cartesian vectors is done by simple addition; as such, the pose translation follows from

  $$\mathbf{t}_i + \mathbf{t}_{\mathsf{R}_j}^{(\mathsf{R}_i)} = \mathbf{t}_j \quad \Longrightarrow \quad \mathbf{t}_{\mathsf{R}_j}^{(\mathsf{R}_i)} = \mathbf{t}_j - \mathbf{t}_i = f_{i,j}^{\text{transl.}}(\mathcal{X}).$$

  Consequently, the error function $\mathbf{e}_{i,j}^{\text{transl.}}(\mathcal{X})$ of a pose translation is defined as

  $$\mathbf{e}_{i,j}^{\text{transl.}}(\mathcal{X}) = f_{i,j}^{\text{transl.}}(\mathcal{X}) - z_{i,j}^{\text{transl.}} = \mathbf{t}_j - \mathbf{t}_i - z_{i,j}^{\text{transl.}} \in \mathbb{R}^2.$$

- *Orientation prior* measurements $z_{i,j}^{\text{or.}\cdot} \in \mathcal{SO}2$ relate a single robot poses $x_i \in \mathcal{SE}(2)$, and approximate the relative translation $R_{\mathsf{R}_j}^{(\mathsf{O})} \in \mathbb{R}^2$ of the robot-fixed frame R_i in terms of the inertial frame O. Accordingly, the measurement model defines

  $$f_i^{\text{or.}\cdot}(\mathcal{X}) = R_{\mathsf{R}_j}^{(\mathsf{O})}(x_i) \in \mathcal{SO}(2).$$

  This translation is a direct component of the robot pose x_i, which brings about the measurement function $f_i^{\text{or.}\cdot}(\mathcal{X}) = R_i$. If it holds that $\mathsf{O} = \mathsf{R}_0$, the measurement model function $f_i^{\text{or.}} = f_{0,i}^{\text{rot.}}$.

  The orientation prior error function $\mathbf{e}_i^{\text{or.}\cdot}(\mathcal{X})$ is defined as

  $$\mathbf{e}_i^{\text{or.}\cdot}(\mathcal{X}) = f_{i,j}^{\text{or.}\cdot}(\mathcal{X}) - z_i^{\text{or.}\cdot} = R_i - z_i^{\text{or.}\cdot} \in \mathcal{SO}(2).$$

- *Location prior* measurements $z_{i,j}^{\text{loc.}\cdot} \in \mathbb{R}^2$ relate a single robot poses $x_i \in \mathcal{SE}(2)$, and approximate the relative translation $\mathbf{t}_{\mathsf{R}_j}^{(\mathsf{O})} \in \mathbb{R}^2$ of the robot-fixed frame R_i in terms of the inertial frame O. Similar to the case of orientation prior, the measurement model function and error function are derived as

  $$\begin{aligned} f_i^{\text{loc.}\cdot}(\mathcal{X}) &= \mathbf{t}_i \\ \mathbf{e}_i^{\text{loc.}\cdot}(\mathcal{X}) &= f_i^{\text{loc.}\cdot}(\mathcal{X}) - z_i^{\text{loc.}\cdot} = \mathbf{t}_i - z_i^{\text{loc.}\cdot} \in \mathbb{R}^2. \end{aligned}$$

Pose transformation:	$f_{i,j}^{\text{transf.}}(\mathcal{X}) = x_i^{-1}\, x_j$
Pose rotation:	$f_{i,j}^{\text{rot.}}(\mathcal{X}) = R_i^{\top}\, R_j$
Pose translation:	$f_{i,j}^{\text{transl.}}(\mathcal{X}) = \mathbf{t}_j - \mathbf{t}_i$
Orientation prior:	$f_i^{\text{or.}}(\mathcal{X}) = R_i$
Location prior:	$f_i^{\text{loc.}}(\mathcal{X}) = \mathbf{t}_i$

Table 2-1: Measurement model functions for some of the most common measurements.

The measurement models are summarised in Table 2-1.

With all measurements being elements in $\mathcal{SE}(2)$ or its sub-components, the measurement model functions exhibit the same relationship. That is, all models can also be defined in terms of the pose transformation measurement model function $f_{i,j}^{\text{transf.}}$:

$$\begin{aligned} f_{i,j}^{\text{rot}}(\mathcal{X}) &= R[f_{i,j}^{\text{transf.}}(\mathcal{X})], \\ f_{i,j}^{\text{transl}}(\mathcal{X}) &= \mathbf{t}[f_{i,j}^{\text{transf.}}(\mathcal{X})], \\ f_i^{\text{or.}}(\mathcal{X}) &= R[f_{0,i}^{\text{transf.}}(\mathcal{X})], \\ f_i^{\text{loc.}}(\mathcal{X}) &= \mathbf{t}[f_{0,i}^{\text{transf.}}(\mathcal{X})], \end{aligned}$$

where $R[f_{\mathcal{I}}^{\text{transf.}}(\mathcal{X})] \in \mathcal{SO}(2)$ and $\mathbf{t}[f_{\mathcal{I}}^{\text{transf.}}(\mathcal{X})] \in \mathbb{R}^2$ denote the rotation and translation elements of $f_{\mathcal{I}}^{\text{transf.}}(\mathcal{X}) \in \mathcal{SE}(2)$, respectively.

2-4 Congruence

This section introduces the notion of *congruence* of a pose graph. A pose graph is said to be *congruent* if all *accumulated constraints* over the set of adjacent constraints can be satisfied. The corresponding solution yields zero-cost over all constraints, which indicates a graph to be under-constrained. Due to the presence of measurement and process noise, measurements are likely to conflict. The solution that takes into account many conflicting measurements tends to neutralise the noise influence and therefore yield an accurate solution. A congruent graph, on the other hand, takes no conflicting measurements into account and produces a solution that perfectly matches all noise influences.

Consider the toy example shown in Figure 2-1, where a pose graph is shown with nodes $\mathcal{X} = \{x_i\}_{i=1}^4$. The pose graph is in the process of adding two scan-matching loop-closure constraints, $z_{2,4}^{\text{scan}}, z_{1,4}^{\text{scan}} \in \mathcal{SE}(2)$. Initially, in Figure 2-1a, all nodes are only connected sequentially by odometry constraints $\mathcal{Z} = \{z_{1,2}^{\text{odo}}, z_{2,3}^{\text{odo}}, z_{3,4}^{\text{odo}}\} \subset \mathcal{SE}(2)$. A constraint is satisfied if the poses are configured such that the constraint cost function is zero. For instance, the constraint that corresponds to $z_{1,2}^{\text{odo}}$ is satisfied if

$$\mathbf{e}_{1,2}^{\text{odo}}(x_1, x_2) = \left\| f_{1,2}^{\text{odo}}(x_1, x_2) \ominus z_{1,2}^{\text{odo}} \right\|_{\Sigma_{1,2}}^2 = \mathbf{0} \quad \Longrightarrow \quad x_1^{-1} x_2 = z_{1,2}^{\text{odo}},$$

where $f_{\mathcal{I}}^{\text{odo}} = f_{\mathcal{I}}^{\text{transf.}}$. An accumulated constraint is constructed by linking together multiple constraints by substitution. For instance, the accumulated constraint of $\{z_{1,2}^{\text{odo}}, z_{2,3}^{\text{odo}}, z_{3,4}^{\text{odo}}\}$

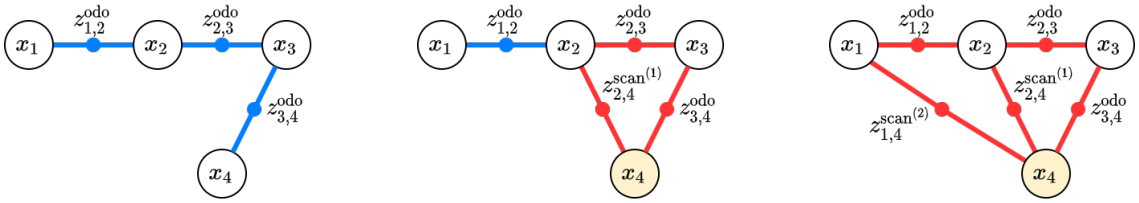
relates the poses x_1 and x_4, and is satisfied if

$$x_1^{-1} x_4 = z_{1,2}^{\text{odo}}\, z_{2,3}^{\text{odo}}\, z_{3,4}^{\text{odo}}.$$

The graph shown in Figure 2-1a is congruent, because all accumulated constraints can be satisfied. More specifically, the graph configuration that satisfies

$$\begin{aligned} x_1^{-1}\, x_2 &= z_{1,2}^{\text{odo}}, \\ x_2^{-1}\, x_3 &= z_{2,3}^{\text{odo}}, \\ x_3^{-1}\, x_4 &= z_{3,4}^{\text{odo}}, \end{aligned}$$

also satisfies all accumulated constraints. Therefore, it can be concluded that for the pose graph shown in Figure 2-1a, the *congruent solution* exists. The constraints that are satisfied by the congruent solutions are indicated by a **blue** colour.



(a) Graph(4-3) **(b)** Graph(4-4) **(c)** Graph(4-5)

Figure 2-1: A toy example of a pose graph.

Such an acyclic (sub)graph, where all nodes are connected sequentially, is referred to as a *'tail'* (or *'graph dead end'*). Its composition inherently allows for all accumulated constraints to be satisfied, because none are conflicting. In contrast, when a graph is made cyclic, such as shown in Figure 2-1b, the set of all accumulated constraints is not likely to be satisfied. For instance, the accumulated constraints of $\{z_{2,3}^{\text{odo}}, z_{3,4}^{\text{odo}}\}$ and of $\{z_{2,4}^{\text{scan}}\}$ both relate the poses x_2 and x_4. Due to influence of measurement and process noise, the constraint value is likely to be different; i.e.,

$$z_{2,3}^{\text{odo}}\, z_{3,4}^{\text{odo}} \neq z_{2,4}^{\text{scan}}.$$

In this case, the graph configuration that satisfies the constraint of $\{z_{2,3}^{\text{odo}}, z_{3,4}^{\text{odo}}\}$ does not satisfy the constraint of $\{z_{2,4}^{\text{scan}}\}$. That is, the measurements $z_{2,3}^{\text{odo}}$ and $z_{3,4}^{\text{odo}}$ are said to be *in conflict* with the measurement $z_{2,4}^{\text{scan}}$. Constraints that are part of the incongruent solution are indicated by a **red** colour. At this point, the subgraph with $\{x_1, x_2, z_{1,2}^{\text{odo}}\}$ is still congruent due to it being a tail.

An acyclic subgraph inherently has no conflicting accumulated constraints, because between any two nodes only one 'path' of constraints exists. This can be included in the definition for congruence: *a subgraph is congruent if a single path of measurement constraints exists between any set of nodes.* This is also the reason that graph optimisation is only performed after the establishment of a loop closure constraint: because the tail is made cyclic, the congruent solution is no longer valid, and optimisation is required.

2-5 Linearisation

In an effort to solve the non-linear least-square minimisation problem defined in (2-1), methods such as Gauss-Newton or Levenberg-Marquardt approximate the error function terms $\mathbf{e}_\mathcal{I}$ by its first-order Taylor series expansion around a given linearisation point $\breve{\mathcal{X}}_\mathcal{I}$. By considering a small vector parameterisation $\Delta\mathbf{x}_\mathcal{I} \cong \Delta\mathcal{X}_\mathcal{I}$ with $\Delta\mathbf{x}_\mathcal{I} \in \mathbb{R}^{m_\mathcal{I}}$, the approximation at $\mathcal{X}_\mathcal{I} = \breve{\mathcal{X}}_\mathcal{I} \boxplus \Delta\mathbf{x}_\mathcal{I}$ is given by

$$\mathbf{e}_\mathcal{I}(\mathcal{X}_\mathcal{I}) = \mathbf{e}_\mathcal{I}(\breve{\mathcal{X}}_\mathcal{I} \boxplus \Delta\mathbf{x}_\mathcal{I}) \approx \mathbf{e}_\mathcal{I}(\breve{\mathcal{X}}_\mathcal{I}) + \breve{J}_\mathcal{I}\Delta\mathbf{x}_\mathcal{I},$$

where $\breve{J}_\mathcal{I} := J_\mathcal{I}(\breve{\mathcal{X}})$ is the Jacobian matrix of partial derivatives of $\mathbf{e}_\mathcal{I}$ at $\breve{\mathcal{X}}$ with respect to $\mathcal{X}_\mathcal{I} \subset \mathcal{X}$. Note that the generalised addition operator $\boxplus$ is used instead of $\oplus$ to not invalidate this notation by the introduction of additional parameter-nodes in Chapter 3 that are not necessarily of the type $\mathcal{SE}(n)$.

By substituting the linear approximation into the error function, a quadratic approximation is obtained:

$$\begin{aligned} F_\mathcal{I}(\breve{\mathcal{X}} \boxplus \Delta\mathbf{x}) &= \mathbf{e}_\mathcal{I}(\breve{\mathcal{X}} \boxplus \Delta\mathbf{x})^\top \, \Sigma_\mathcal{I}^{-1} \, \mathbf{e}_\mathcal{I}(\breve{\mathcal{X}} \boxplus \Delta\mathbf{x}) \\ &\approx \left[\mathbf{e}_\mathcal{I}(\breve{\mathcal{X}}) + \breve{J}_\mathcal{I}\Delta\mathbf{x}\right]^\top \, \Sigma_\mathcal{I}^{-1} \, \left[\mathbf{e}_\mathcal{I}(\breve{\mathcal{X}}) + \breve{J}_\mathcal{I}\Delta\mathbf{x}\right] \\ &= \underbrace{\left[\mathbf{e}_\mathcal{I}(\breve{\mathcal{X}})^\top \Sigma_\mathcal{I}^{-1} \mathbf{e}_\mathcal{I}(\breve{\mathcal{X}})\right]}_{c_\mathcal{I}} + 2\underbrace{\left[\mathbf{e}_\mathcal{I}(\breve{\mathcal{X}})^\top \Sigma_\mathcal{I}^{-1} \breve{J}_\mathcal{I}\right]}_{\mathbf{b}_\mathcal{I}^\top} \Delta\mathbf{x} + \Delta\mathbf{x}^\top \underbrace{\left[\breve{J}_\mathcal{I}^\top \Sigma_\mathcal{I}^{-1} \breve{J}_\mathcal{I}\right]}_{H_\mathcal{I}} \Delta\mathbf{x}. \end{aligned}$$

With this local approximation, the total cost function is obtained as through summation over all constraints, which can then be combined into a single matrix equation:

$$\begin{aligned} F_{\text{pgo}}(\breve{\mathcal{X}} \oplus \Delta\mathbf{x}) &\approx \sum_\mathcal{I} \left(c_\mathcal{I} + 2\mathbf{b}_\mathcal{I}^\top \Delta\mathbf{x} + \Delta\mathbf{x}^\top H_\mathcal{I} \Delta\mathbf{x}\right) \\ &= c + 2\mathbf{b}^\top \Delta\mathbf{x} + \Delta\mathbf{x}^\top H \Delta\mathbf{x}. \end{aligned}$$

In order to take a closer look into this linearisation process, a single constraint is considered in detail. The constraint with identifier $\mathcal{I}$ has m-dimensional measurement $z_\mathcal{I}$ and thus an m-dimensional vector error function $\mathbf{e}_\mathcal{I}(\mathcal{X})$. It is connected to q nodes $\mathcal{X}_\mathcal{I}$.

1. First, the Jacobian matrices of the error function $\mathbf{e}_\mathcal{I}$ are calculated with to respect to each of the connected nodes. That is, for each node $x_i \in \mathcal{X}_\mathcal{I}$ with parameterisation $\mathbf{x}_i \in \mathbb{R}^{m_i}$, the Jacobian sub-matrix is defined as

$$\breve{J}_{\mathcal{I}_i} = \begin{bmatrix} \left.\frac{\partial \mathbf{e}_\mathcal{I}^{[1]}(\mathcal{X}_\mathcal{I})}{\partial \mathbf{x}_i^{[1]}}\right|_{\mathcal{X}_\mathcal{I}=\breve{\mathcal{X}}_\mathcal{I}} & \cdots & \left.\frac{\partial \mathbf{e}_\mathcal{I}^{[1]}(\mathcal{X}_\mathcal{I})}{\partial \mathbf{x}_i^{[m_i]}}\right|_{\mathcal{X}_\mathcal{I}=\breve{\mathcal{X}}_\mathcal{I}} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial \mathbf{e}_\mathcal{I}^{[m]}(\mathcal{X}_\mathcal{I})}{\partial \mathbf{x}_i^{[1]}}\right|_{\mathcal{X}_\mathcal{I}=\breve{\mathcal{X}}_\mathcal{I}} & \cdots & \left.\frac{\partial \mathbf{e}_\mathcal{I}^{[m]}(\mathcal{X}_\mathcal{I})}{\partial \mathbf{x}_i^{[m_i]}}\right|_{\mathcal{X}_\mathcal{I}=\breve{\mathcal{X}}_\mathcal{I}} \end{bmatrix} \in \mathbb{R}^{m \times m_i},$$

   where $\mathbf{x}_i^{[j]}$ and $\mathbf{e}_\mathcal{I}^{[j]}(\mathcal{X}_\mathcal{I})$ denote jth elements of $\mathbf{x}_i$ and $\mathbf{e}_\mathcal{I}(\mathcal{X}_\mathcal{I})$, respectively. Each of the columns of the Jacobian sub-matrix describes the gradient of the error function $\mathbf{e}_\mathcal{I}(\mathcal{X}_\mathcal{I})$

with respect to a degree of freedom of node $x_i \in \mathcal{X}_\mathcal{I}$. The jth column, denoted by $\breve{J}^{[j]}_{\mathcal{I}_i}$, can be numerically approximated with a central difference scheme as [11]

$$\breve{J}^{[j]}_{\mathcal{I}_i} \approx \frac{1}{2\delta}\Big(\mathbf{e}_\mathcal{I}(x_i \boxplus \delta \mathbf{1}^{[j]}_{m_i}) - \mathbf{e}_\mathcal{I}(x_i \boxplus -\delta \mathbf{1}^{[j]}_{m_i})\Big) \in \mathbb{R}^{m_i},$$

where $\delta > 0$ is a small constant (e.g. 1×10^{-9}) and $\mathbf{1}^{[j]}_{m_i} \in \mathbb{R}^{m_i}$ denotes the unit vector of size m_i along dimension j.

The Jacobian sub-matrix $\breve{J}_{\mathcal{I}_i}$ of $\mathbf{e}_\mathcal{I}$ with respect to node x_i can then be constructed by concatenating all columns as

$$\breve{J}_{\mathcal{I}_i} = \begin{bmatrix} \breve{J}^{[1]}_{\mathcal{I}_i} & \cdots & \breve{J}^{[m_i]}_{\mathcal{I}_i} \end{bmatrix} \in \mathbb{R}^{m \times m_i}.$$

Finally, the full constraint Jacobian can be constructed by concatenating all sub-matrices that correspond to the connecting nodes as

$$\breve{J}_\mathcal{I} = \begin{bmatrix} \breve{J}_{\mathcal{I}_1} & \cdots & \breve{J}_{\mathcal{I}_q} \end{bmatrix} \in \mathbb{R}^{m \times \left(\Sigma_i m_i\right)}.$$

2. The linearisation constants, $\mathbf{b}_\mathcal{I}$ and $H_\mathcal{I}$, comprise of matrix-blocks for each of the connected nodes, with

$$\mathbf{b}_\mathcal{I} := \breve{J}_\mathcal{I}^\top \Sigma_\mathcal{I}^{-1} \mathbf{e}_\mathcal{I}(\breve{\mathcal{X}}) = \begin{bmatrix} \breve{J}_{\mathcal{I}_1} & \cdots & \breve{J}_{\mathcal{I}_q} \end{bmatrix}^\top \Sigma_\mathcal{I}^{-1} \mathbf{e}_\mathcal{I}(\breve{\mathcal{X}})$$

$$H_\mathcal{I} := \breve{J}_\mathcal{I}^\top \Sigma_\mathcal{I}^{-1} \breve{J}_\mathcal{I} = \begin{bmatrix} \breve{J}_{\mathcal{I}_1}^\top \Sigma_\mathcal{I}^{-1} \breve{J}_{\mathcal{I}_1} & \cdots & \breve{J}_{\mathcal{I}_1}^\top \Sigma_\mathcal{I}^{-1} \breve{J}_{\mathcal{I}_q} \\ \vdots & \ddots & \vdots \\ \breve{J}_{\mathcal{I}_q}^\top \Sigma_\mathcal{I}^{-1} \breve{J}_{\mathcal{I}_1} & \cdots & \breve{J}_{\mathcal{I}_q}^\top \Sigma_\mathcal{I}^{-1} \breve{J}_{\mathcal{I}_q} \end{bmatrix}$$

2-6 Performance metrics

The performance of a PGO solution is evaluated using performance metrics that compare the ground truth solution to the estimated solution. This is in contrast to using the solution cost as a performance metric, which does not necessarily imply an *accurate* solution, but rather an *cost-optimal* solution [24]. In the map optimisation domain, the error surface can be complex due to the nature of the underlying observations. As such, it is possible for a map to change dramatically with little effect on the error value. Furthermore, if a graph is constructed from only odometric constraints that relate poses sequentially, the graph is labelled congruent (see Figure 2-1a), and a zero-cost solution exists that places each pose perfectly at the measured location relative to the previous pose. Even if the measurements are completely off and the resulting graph looks nothing like the ground truth, the cost will be zero.

For a performance metric to be unaffected by these issues, the ground truth has to be taken into account when evaluating the solution quality. Two frequently employed methods are defined below [30, 27]. Note that it is assumed that the estimate graph and ground truth graph contain identical indexing of robot poses that are identically connected by constraints. If this is not the case, values of the metrics can still be computed by taking the intersection of the relevant graph elements between both graphs.

1. *Absolute Trajectory Error (ATE)* evaluates the global consistency of the graph by comparing the absolute distances between estimates and ground truth trajectory. For each robot pose that is present in both graphs, a cost term is contributed that represents the absolute difference in translation. The ATE is defined by taking the root-mean-square error of these terms:
$$e_{\text{ate}}(\mathcal{X}) = \sqrt{\frac{1}{|\mathcal{X}|}\sum_i \left\|\mathbf{t}_i^{\text{truth}} - \mathbf{t}_i\right\|^2}.$$
2. *Relative Position Error (RPE)* evaluates the local accuracy of the trajectory by comparing pairs of robot poses connected by relative pose transformation constraints. For two poses that are connected by such a constraint, the relative transformation is denoted by $\Delta\mathcal{X}_\mathcal{I} = x_i^{-1}\, x_j$, where $\mathcal{I} = \{i, j\}$. The RPE is divided into a translation component $e_{\text{rpe-transl.}}(\mathcal{X})$ and a rotation component $e_{\text{rpe-rot.}}(\mathcal{X})$, which are defined as
$$e_{\text{rpe-transl.}}(\mathcal{X}) = \sqrt{\frac{1}{|\mathcal{Z}|}\sum_\mathcal{I} \left\|\text{transl}(\Delta\mathcal{X}_\mathcal{I}^{\text{truth}} - \Delta\mathcal{X}_\mathcal{I})\right\|^2},$$
$$e_{\text{rpe-rot.}}(\mathcal{X}) = \sqrt{\frac{1}{|\mathcal{Z}|}\sum_\mathcal{I} \left\|\text{rot}(\Delta\mathcal{X}_\mathcal{I}^{\text{truth}} - \Delta\mathcal{X}_\mathcal{I})\right\|^2}.$$

The performance metrics provide a scalar value evaluation of the estimated graph relative to the ground truth. To offer more insight, all metrics can be plotted over time with respect to an increasing graph. This provides insights into the influence of certain constraints by the induced effect on the performance metrics.

2-7 Summary

Pose Graph Optimisation (PGO) formalises the maximum likelihood formulation of the set of poses $\mathcal{X} \subset \mathcal{SE}(2)$ given the set of measurements $\mathcal{Z} \subset \mathcal{M}$ using a pose graph. In a pose graph poses are represented by nodes and measurements are encoded in the edges. Each measurement $z_\mathcal{I} \in \mathcal{Z}$, identified by the set of node indices $\mathcal{I}$, is paired with a measurement model to form a graph constraint. The measurement model comprises:

- Measurement model function $f_\mathcal{I} : \mathcal{X} \to \mathcal{M}$, which maps the set of poses $\mathcal{X}$ to an expected measurement value in $\mathcal{M}$;
- Measurement model covariance (or constraint covariance) $\Sigma_\mathcal{I} \in \mathbb{R}^{m_\mathcal{I} \times m_\mathcal{I}}$, which describes the accuracy of the measurement in terms of a covariance matrix.

Each constraint defines an error function $\mathbf{e}_\mathcal{I}(\mathcal{X}) = f_\mathcal{I}(\mathcal{X}) \boxminus z_\mathcal{I} \in \mathbb{R}^{m_\mathcal{I}}$ and contributes an cost term $F_\mathcal{I}(\mathcal{X}) = \|\mathbf{e}_\mathcal{I}(\mathcal{X})\|^2_{\Sigma_\mathcal{I}}$ that is quadratic in the error function and linear in the constraint covariance. The PGO problem is given by
$$\mathcal{X}^* = \arg\min_\mathcal{X} \sum_\mathcal{I} F_\mathcal{I}(\mathcal{X}).$$
PGO relies on least-square minimisation techniques to find the cost-optimal solution over all constraints in the pose graph.

Congruence describes the presence of conflicting measurements within a subgraph. A graph that is congruent has a zero-cost solution that perfectly matches all constraints. Although this seems favourable at first sight, congruence indicates a graph to be under-constrained.

Chapter 3

Parameter calibration

Pose Graph Optimisation (PGO) relies on the measurement models that are encoded in the pose graph to be accurate. Oftentimes, these measurement models are dependent on knowledge of specific robot parameters that are to be set a priori, or these models simply fail to account for unknown systematic measurement deviations. By identifying the dependency of the unknown parameter within the measurement model, the parameter can be modelled by a set of graph nodes [17], thereby creating a *pose-parameter graph* and the accompanying *Pose-Parameter Graph Optimisation (PPGO)* problem. This is an extension of PGO, which is derived in Chapter 2.

In this chapter a generalised approach to PPGO is proposed that is based on the implementation of two generally applicable parameters (a *bias* and *scaling factor*). These implementations comprise *modified measurement models* and *connectivity strategies* of the set of parameter-nodes within the pose-parameter graph.

This chapter covers the following:

- Section 3-1 *'Motivation'* discusses the motivation for the use of PPGO by introducing a number of interesting use-cases.
- Section 3-2 *'Pose-parameter graph optimisation'* introduces the PPGO problem formulation and discusses the implementation differences with respect to PGO. Extended measurement models are required for the construction of the pose-parameter graph that simultaneously relate the pose-nodes and parameter-nodes. Connectivity strategies dictate the set of parameter-nodes used to model a parameter.
- Section 3-3 *'Parameter-nodes'* introduces the measurement models of the generally applicable parameters that are investigated in this research: the *bias* parameter (and *sensor frame* parameter) and the *scaling factor* parameter.
- Section 3-4 *'Connectivity strategies'* discusses the connectivity strategies that exist for the estimation of a constant parameter, a time-fluctuating parameter, and a space-fluctuating parameter.
- Section 3-5 *'Summary'* summarises the content of this chapter.

3-1 Motivation

Every sensor system is dependent on some set of parameters that is used to convert the raw sensor data to meaningful measurements. Especially in PGO and its derivatives, where all measurements are some form of relative pose transformation, rotation, or translation, measurements are often dependent on robot-specific parameters. The accuracy of these parameters can have a substantial effect on the accuracy of the potential SLAM solution.

Some parameters can be hard to measure and only obtained using lengthy ad-hoc calibration procedures (such as e.g. a sensor coordinate frame transformation), which need to be performed every time the robot configuration changes. Furthermore, if a parameter is subject to change during operation (such as e.g. wheel radii for inflated tyres), any estimate obtained from such a procedure is promptly rendered useless during the mapping process. Self-calibration can also be employed to compensate for a systematic measurement deviation (such as e.g an unanticipated sensor bias). In this case, the deviation is modelled as a parameter and the measurement models are modified based on an assumed dependency relation.

To highlight the advantages of PPGO. some use-cases are sketched that would benefit from the live parameter-calibration capabilities of PPGO:

- Consider a robot that carries a LiDAR sensor. The relative transformation $T_{\mathsf{S}^{\text{lidar}}}^{(\mathsf{R})}$ of the sensor-fixed frame $\mathsf{S}^{\text{lidar}}$ with respect to the body-fixed frame R is required to derive the relative pose transformation from the scan-matching algorithm, which itself produces a relative sensor-frame transformation. Without a good estimate of this parameter, all relevant constraints will be systematically off, and the solution will not be accurate.

  For a rigidly-attached sensor, the body-to-sensor transformation can be considered *constant*. Its value can be considered a *sensor frame parameter* that needs to be added at both ends of the relative sensor transformation.

- Consider a post-delivery robot that goes around an office building carrying packages of varying weights. Packages are constantly being loaded and unloaded as it moves around. Its wheels comprise tyres that deform based on the weight of the load being carried by the robot. If the odometry sensor system relies on rotary encoders on the wheels for its pose transformations, an accurate estimate of the wheel radii is required.

  This influence can be modelled by a *time-dependent* parameter that acts as a *scaling factor* on the translation component of the pose transformation measurements. The weight of A heavy load constitutes are small effective wheel radius, and therefore a scaling factor value below one.

- Consider a robot that uses a GPS sensor to establish location prior constraints in the vicinity of tall buildings. The GPS signal can be obstructed by the buildings or cause deviations due to the signal reflection. The measurement deviation is a function of the environment and fluctuates as a function of space.

  The measurement deviation can be modelled by a *space-dependent* translation *bias* parameter. A parameter value map can be constructed for use in other robots or mapping processes.

3-2 Pose-parameter graph optimisation

Pose-Parameter Graph Optimisation (PPGO) is proposed as an extension to Pose Graph Optimisation (PGO) that is able to estimate a parameter value by including a set of parameter-nodes in the pose-graph, thereby forming a pose-parameter graph. The pose parameter graph formalises the maximum likelihood formulation of the set of poses-nodes and parameter-nodes, given the set of measurements. The goal of PPGO is to find the most likely path (represented by a set of pose-nodes) and parameter values (represented by a set of parameter-nodes) given the set of measurements.

This section covers the implementation of PPGO in terms of the required modifications with respect to the PGO problem.

3-2-1 Problem description

Similar to PGO, PPGO relies on least-square minimisation techniques to find the cost-optimal Maximum a Posteriori (MAP) estimate that minimises the weighted mismatch between the expected and true measurement values over all measurements. The PPGO problem formulation is given by

$$\begin{aligned} \mathbf{e}_{\mathcal{I}}(\mathcal{N}) &= f_{\mathcal{I}}(\mathcal{N}) \boxminus z_{\mathcal{I}}, \\ F_{\mathcal{I}}(\mathcal{N}) &= \|\mathbf{e}_{\mathcal{I}}(\mathcal{N})\|^2_{\Sigma_{\mathcal{I}}}, \\ F_{\text{ppgo}}(\mathcal{N}) &= \sum_{\mathcal{I}} F_{\mathcal{I}}(\mathcal{N}), \\ \mathcal{N}^* &= \arg\min_{\mathcal{N}} F_{\text{ppgo}}(\mathcal{N}), \end{aligned} \tag{3-1}$$

where the following differences with (2-4) are noted:

- $\mathcal{P} = \{p\}$ denotes the set of parameter-nodes;
- $\mathcal{N} = \mathcal{X} \cup \mathcal{P}$ denotes the set of all PPGO variables, which is union of the set of pose-nodes $\mathcal{X}$ and the parameter-nodes $\mathcal{P}$;
- $F_{\text{ppgo}}(\mathcal{N})$ denotes the scalar PPGO cost function that is the summation of the cost terms $\|\mathbf{e}_{\mathcal{I}}(\mathcal{N})\|^2_{\Sigma_{\mathcal{I}}}$ contributed by all constraints formalised in the pose-parameter graph, as a function of the set of nodes $\mathcal{N}$.

3-2-2 Connectivity strategies

In PGO, a trajectory is represented by a set of pose-nodes with a unique entry for every time instance. Correspondingly, the set of pose-nodes is able to provide an estimate for the robot pose at every instance. The extension to PPGO does not enable an identical modelling approach for the set of parameters: a unique-parameter node for every time instance could result in an under-constrained optimisation problem. This is because the parameters are not directly measured, but rather deduced from systematic inconsistencies between the measurement models and actual measurement values. As a result, no meaningful constraints can

be imposed on the set of parameter-nodes. Without constraints, the set of parameter-nodes can be manipulated to form a zero-cost solution irrespective of the value of the connecting pose-nodes. Alternative *connectivity strategies* are required to connect a set of (non-unique) parameter-nodes to the set of relevant constraints.

As given by (2-4) and (3-1), the constraint cost term function is defined as

$$F_{\mathcal{I}}(\mathcal{N}_{\mathcal{I}}) = \|\mathbf{e}_{\mathcal{I}}(\mathcal{N}_{\mathcal{I}})\|^2_{\Sigma_{\mathcal{I}}} = \|f_{\mathcal{I}}(\mathcal{N}_{\mathcal{I}}) \boxminus z_{\mathcal{I}}\|^2_{\Sigma_{\mathcal{I}}}.$$

This function has a unique minimum where the pose-nodes and parameter-nodes are configured such that the outcome of the measurement model $f_{\mathcal{I}}(\mathcal{N}_{\mathcal{I}})$ perfectly matches the observed measurement value $z_{\mathcal{I}}$, resulting in $\mathbf{e}_{\mathcal{I}}(\mathcal{N}_{\mathcal{I}}) = \mathbf{0}$. This node configuration is only part of the solution if the rest of the pose graph contains no conflicting accumulated constraints that relate the same set of nodes (see Section 2-4). The addition of a unique parameter-node for every instance could result in an underdetermined system for $\mathbf{e}_{\mathcal{I}}(\mathcal{N}_{\mathcal{I}}) = \mathbf{0}$, with infinitely many solutions for $\mathcal{N}_{\mathcal{I}}$.

Consider the example posed Figure 3-1.

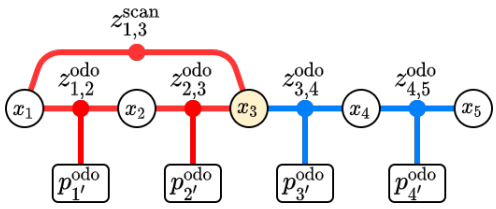


Figure 3-1: An underdetermined PPGO problem due to the addition of a unique parameter-node for every instance. Without constraints imposed on the parameter-nodes, the system $F_{\text{ppgo}}(\mathcal{N}) = 0$ has 8 unknowns (9 nodes of which x_1 is fixed at the origin, and 5 equations (or measurements).

The congruent solution to the PPGO cost function

$$F_{\text{ppgo}}(\mathcal{N}) = \sum_{\mathcal{I}} \|\mathbf{e}_{\mathcal{I}}(\mathcal{N}_{\mathcal{I}})\|^2_{\Sigma_{\mathcal{I}}} = 0$$

has 8 unknowns (4 parameter-nodes $\{p^{\text{odo}}_{1'}, p^{\text{odo}}_{2'}, p^{\text{odo}}_{3'}, p^{\text{odo}}_{4'}\}$ and 4 pose-nodes $\{x_2, x_3, x_4, x_5\}$, with x_1 fixed at the origin) and 5 equations:

$$\begin{aligned}
\mathbf{e}_{1,2,1'}(x_1, x_2, p^{\text{odo}}_{1'}) &= \mathbf{0}, \\
\mathbf{e}_{2,3,2'}(x_2, x_3, p^{\text{odo}}_{2'}) &= \mathbf{0}, \\
\mathbf{e}_{3,4,3'}(x_3, x_4, p^{\text{odo}}_{3'}) &= \mathbf{0}, \\
\mathbf{e}_{4,5,4'}(x_4, x_5, p^{\text{odo}}_{4'}) &= \mathbf{0}, \\
\mathbf{e}_{6,7,5'}(x_5, x_6, p^{\text{odo}}_{5'}) &= \mathbf{0}.
\end{aligned}$$

If no other constraint is imposed on the value of the parameter-node, its value can be chosen freely to always intersect with the congruent solution, irrespective of the values of $\mathcal{X}_{\mathcal{I}} \subset \mathcal{N}_{\mathcal{I}}$. This undesired behaviour can be mitigated by strategically connecting a set of parameter-nodes that is not unique for every time instance. This will be referred to as a *connectivity strategy*. The connectivity strategies proposed in this research are discussed in Section 3-4.

3-2-3 Measurement models

The measurement model function $f_{\mathcal{I}} : \mathcal{N} \to \mathcal{M}$ maps the set of connected nodes $\mathcal{N}_{\mathcal{I}}$ to an expected measurement value in $\mathcal{M}$. Its implementation depends on the sensor from which the measurement is generated, which in turn defines a dependency relation with a set of pose-nodes $\mathcal{X}_{\mathcal{I}} \subset \mathcal{N}_{\mathcal{I}}$ and a set of parameter-nodes $\mathcal{P}_{\mathcal{I}} \subset \mathcal{N}_{\mathcal{I}}$.

In Section 2-3-2, the measurement model functions are introduced for some of the common measurements, irrespective of sensor implementation. These models are repeated in Table 3-1.

Pose transformation:	$f_{i,j}^{\text{transf.}\cdot}(\mathcal{X}) = x_i^{-1}\, x_j$
Pose rotation:	$f_{i,j}^{\text{rot.}\cdot}(\mathcal{X}) = R_i^{\top}\, R_j = R\big[f_{i,j}^{\text{transf.}\cdot}(\mathcal{X})\big]$
Pose translation:	$f_{i,j}^{\text{transl.}\cdot}(\mathcal{X}) = \mathbf{t}_j - \mathbf{t}_i = \mathbf{t}\big[f_{i,j}^{\text{transf.}\cdot}(\mathcal{X})\big]$
Location prior:	$f_i^{\text{loc.}\cdot}(\mathcal{X}) = \quad \mathbf{t}_i \quad = \mathbf{t}\big[f_{0,i}^{\text{transf.}\cdot}(\mathcal{X})\big]$

Table 3-1: Measurement model functions for some of the most common measurements.

In order to include a dependency on a set of parameter-nodes, a relationship has to be defined that describes the effect of the parameter-node value on the expected measurement value. This relationship is dependent on the *interpretation* of the parameter-node. Whereas a pose-node has the unambiguous interpretation of a robot pose at a specific time instance, the interpretation of the parameter-node as e.g. a bias term or scaling factor inherently changes the definition of the corresponding measurement model function.

3-3 Parameter-nodes

This research investigates the reconstruction of two generally applicable parameters: a *bias* parameter and a *scaling* parameter. Furthermore, the reconstruction of the sensor frame transformation parameter (which is essentially a special case of the bias parameter) is also investigated. The parameters are modelled by corresponding parameter-nodes, which are defined in terms of a modified measurement model that includes a dependency on a corresponding parameter-node.

3-3-1 Bias

The *bias* of an estimator is the difference between this estimator's expected value and the true value. A biased estimator can be made unbiased if an unbiased estimate of the bias is subtracted from the biased estimator.

The implementation of a bias parameter is dependent on the type of constraint it is connected to. As discussed in Section 2-3-2, all measurement model functions can be defined in terms of the pose transformation model function $f_{\mathcal{I}}^{\text{transf.}\cdot}$. The same is true for the modified measurement model functions used in PPGO. The model function $f_{i,j}^{\text{transf.}\cdot}$ approximates the relative transformation $T_{\mathsf{R}_j}^{(\mathsf{R}_i)} \in \mathcal{SE}(2)$ of the robot-fixed frame R_j in terms of R_i, and is derived in

(2-6) as

$$f_{i,j}^{\text{transf.}}(\mathcal{X}) = x_i^{-1}\, x_j \in \mathcal{SE}(2).$$

The bias parameter can be modelled to be present on any of the components of the relative pose transformation measurement model function $f_{i,j}^{\text{transf.}}$.

First, consider a bias parameter-node $p^{\text{bias}(x,y,\theta)} \in \mathcal{SE}(2)$ applied to all components $\{x, y, \theta\}$ of $f_{i,j}^{\text{transf.}}$. The modified measurement model is given by

$$f_{i,j,k}^{\text{transf.}}(\mathcal{N}) = f_{i,j}^{\text{transf.}}(\mathcal{X})\, p_k^{\text{bias}(x,y,\theta)} = x_i^{-1}\, x_j\, p_k^{\text{bias}(x,y,\theta)} \in \mathcal{SE}(2), \tag{3-2}$$

where the bias parameter-node $p_k^{\text{bias}(x,y,\theta)}$ is 'added' onto the PGO measurement model function $f_{i,j}^{\text{transf.}}$ by right-multiplication.

Now, consider the case of a bias parameter-node $p_k^{\text{bias}(x,y)} \in \mathbb{R}^2$ applied to the translation component of $f_{i,j}^{\text{transf.}}$. Although composition of an element in $\mathcal{SE}(2)$ with an element $\mathbb{R}^2$ is undefined, the desired result can be achieved by constructing a *dummy transformation element* $T\langle p_k^{\text{bias}(x,y)}\rangle \in \mathcal{SE}(2)$ that has all unconsidered components set to zero. That is, $T\langle p_k^{\text{bias}(x,y)}\rangle = T(p_k^{\text{bias}(x,y)}, R(0))$, where $R(0) \in \mathcal{SO}(2)$ is the zero-rotation element. The dummy transformation element *can* be composed with $f_{i,j}^{\text{transf.}}(\mathcal{X})$, similarly to (3-2). For all lower-dimension parameter-nodes, the same trick can be applied:

$$T\langle p_k^{\text{bias}}\rangle = \begin{cases} T(p_k^{\text{bias}(x)}, 0, 0), & \text{for } x\text{-translation bias node } p_k^{\text{bias}(x)} \in \mathbb{R} \\ T(0, p_k^{\text{bias}(y)}, 0), & \text{for } y\text{-translation bias node } p_k^{\text{bias}(y)} \in \mathbb{R} \\ T(\mathbf{0}, p_k^{\text{bias}(\theta)}), & \text{for rotation bias node } p_k^{\text{bias}(\theta)} \in \mathcal{SO}(2) \\ T(p_k^{\text{bias}(x,y)}, R(0)), & \text{for } (x,y)\text{-translation bias node } p_k^{\text{bias}(x,y)} \in \mathbb{R}^2 \\ p_k^{\text{bias}(x,y,\theta)}, & \text{for transformation bias node } p_k^{\text{bias}(x,y,\theta)} \in \mathcal{SE}(2) \end{cases}$$

The generalised modified measurement model function abstracts the specific implementation of the parameter-node by containing the dummy transformation element in its definition, and is given by

$$f_{i,j,k}^{\text{transf.}}(\mathcal{N}) = f_{i,j}^{\text{transf.}}(\mathcal{X})\, T\langle p_k^{\text{bias}}\rangle = x_i^{-1}\, x_j\, T\langle p_k^{\text{bias}}\rangle.$$

Using the generalisation derived in Section 2-3-2, all measurement models can be defined in terms of the pose transformation measurement model as

$$\begin{aligned} f_{i,j,k}^{\text{rot.}}(\mathcal{N}) &= R\big[f_{i,j,k}^{\text{transf.}}(\mathcal{N})\big] \in \mathcal{SO}(2), \\ f_{i,j,k}^{\text{transl.}}(\mathcal{N}) &= \mathbf{t}\big[f_{i,j,k}^{\text{transf.}}(\mathcal{N})\big] \in \mathbb{R}^2, \\ f_{i,k}^{\text{or.}}(\mathcal{N}) &= R\big[f_{0,i,k}^{\text{transf.}}(\mathcal{N})\big] \in \mathcal{SO}(2) \\ f_{i,k}^{\text{rot.}}(\mathcal{N}) &= \mathbf{t}\big[f_{0,i,k}^{\text{transf.}}(\mathcal{N})\big] \in \mathbb{R}^2. \end{aligned} \tag{3-3}$$

3-3-2 Scaling factor

A *scaling factor* parameter is a multiplicative parameter that linearly scales the measurement model function. Similar to the case of the bias parameter considered in Section 3-3-1, the modification to the measurement model due to the addition of the scaling parameter node can be generalised by considering the pose transformation measurement model $f_{\mathcal{I}}^{\text{transf.}}$.

Consider a scaling parameter $p^{\text{scale}(x,y,\theta)} = (x_k^{\text{scale}}, y_k^{\text{scale}}, \theta_k^{\text{scale}}) \in \mathbb{R}^3$, where each element represents the scaling factor that corresponds to measurement components $\{x, y, \theta\}$ of $f_{i,j}^{\text{transf.}}$. The identifying vector of the measurement model is denoted by $\boldsymbol{\xi}_{f_{i,j}}(\mathcal{X}) = (x_{f_{i,j}}, y_{f_{i,j}}, \theta_{f_{i,j}})(\mathcal{X}) \in \mathbb{R}^3$, where $\boldsymbol{\xi}_{f_{i,j}}(\mathcal{X}) \cong f_{i,j}(\mathcal{X})$. The modified measurement model is defined by the element in $\mathcal{SE}(2)$ that follows from the element-wise vector product (denoted by operator $\odot$) of $p_k^{\text{scale}(x,y,\theta)}$ and $\boldsymbol{\xi}_{f_{i,j}}(\mathcal{X})$:

$$f_{i,j,k}^{\text{transf.}}(\mathcal{N}) = T\big(p_k^{\text{scale}(x,y,z)} \odot \boldsymbol{\xi}_{f_{i,j}}(\mathcal{X})\big) = T\left(\begin{bmatrix} x_k^{\text{scale}} \cdot x_{f_{i,j}}(\mathcal{X}) \\ y_k^{\text{scale}} \cdot y_{f_{i,j}}(\mathcal{X}) \\ \theta_k^{\text{scale}} \cdot \theta_{f_{i,j}}(\mathcal{X}) \end{bmatrix}\right).$$

For all lower-dimension parameter-nodes, a dummy scaling vector in $\mathbb{R}^3$ can be constructed that has the unconsidered elements set to the default scaling vector of 1. This operation is denoted by operator $\boldsymbol{\upsilon}\langle p_k^{\text{scale}}\rangle \in \mathbb{R}^3$. Consequently, the generalised modified measurement model is defined as

$$f_{\mathcal{I}}^{\text{transf.}}(\mathcal{N}) = T\big(\boldsymbol{\upsilon}\langle p_k^{\text{scale}}\rangle \odot \boldsymbol{\xi}_{f_{i,j}}(\mathcal{X})\big).$$

For constraints with different measurements (i.e., pose translation, rotation, and location prior), the error function can be obtained similarly to (3-3).

3-3-3 Sensor frame

The *sensor frame* parameter is a special case of the bias parameter that models the transformation component between the sensor-fixed coordinate S frame and the robot-fixed coordinate frame R, denoted by $T_{\mathsf{S}}^{(\mathsf{R})} \in \mathcal{SE}(2)$. Consider, for instance, a sensor that measures the transformation between the sensor coordinate frame at two different instants, S_i and S_j, denoted by $T_{\mathsf{S}_j}^{(\mathsf{S}_i)}$. Accordingly, the measurement model defines

$$f_{i,j,k}^{\text{transf.}}(\mathcal{N}) = T_{\mathsf{S}_j}^{(\mathsf{S}_i)}(x_i, x_j, p_k^{\text{frame}}) \in \mathcal{SE}(2),$$

where

$$T_{\mathsf{S}_j}^{(\mathsf{S}_i)} = T_{\mathsf{S}_j}^{(\mathsf{R}_j)}\, T_{\mathsf{R}_j}^{(\mathsf{R}_i)}\, T_{\mathsf{R}_i}^{(\mathsf{S}_i)}.$$

If the sensor frame transformation is assumed constant, it can be denoted by $\ell = T_{\mathsf{R}_i}^{(\mathsf{S}_i)} = \big(T_{\mathsf{S}_j}^{(\mathsf{R}_j)}\big)^{-1}$. The closed form can then be is written as

$$T_{\mathsf{S}_j}^{(\mathsf{S}_i)} = \begin{bmatrix} R_{\mathsf{R}_j}^{(\mathsf{R}_i)} & R_\ell^\top\big[\mathbf{t}_{\mathsf{R}_j}^{(\mathsf{R}_i)} + (R_{\mathsf{R}_j}^{(\mathsf{R}_i)} - I_2)\mathbf{t}_\ell\big] \\ 0 & 1 \end{bmatrix}$$

The closed form reveals that for a constant sensor frame transformation, only the translation component of the pose transformation is affected. Furthermore, for a small pose rotation, $R_{\mathsf{R}_j}^{(\mathsf{R}_i)} \approx I_2$ and

$$T_{\mathsf{S}_j}^{(\mathsf{S}_i)} = \begin{bmatrix} R_{\mathsf{R}_j}^{(\mathsf{R}_i)} & R_\ell^\top \mathbf{t}_{\mathsf{R}_j}^{(\mathsf{R}_i)} \\ 0 & 1 \end{bmatrix}, \tag{3-4}$$

of which only the translation component is affected by the sensor frame transformation. Moreover, for small R_ℓ, following the small-angle approximation $\{\sin(\theta) \approx \theta, \cos(\theta) \approx 1\}$, only the y-component of $\mathbf{t}_{\mathsf{R}_j}^{(\mathsf{R}_i)}$ is significantly affected.

$\ell \in \mathcal{SE}(2)$ can be modelled as a sensor frame parameter-node over all dimensions, given by $p_k^{\text{frame}(x,y,\theta)} \in \mathcal{SE}(2)$. The corresponding measurement model is written as

$$f_{i,j,k}^{\text{transf.}}(\mathcal{N}) = (p_k^{\text{frame}(x,y,\theta)})^{-1}\, f_{i,j}^{\text{transf.}}(\mathcal{X})\, p_k^{\text{frame}(x,y,\theta)}.$$

For lower-dimension parameter-nodes, a dummy transformation can be constructed as $T\langle p_k^{\text{frame}}\rangle \in \mathcal{SE}(2)$. The measurement model function is then written as

$$f_{i,j,k}^{\text{transf.}}(\mathcal{N}) = (T\langle p_k^{\text{frame}}\rangle)^{-1}\, f_{i,j}^{\text{transf.}}(\mathcal{X})\, T\langle p_k^{\text{frame}}\rangle.$$

The measurement model can be generalised as a sub-component of the relative pose transformation measurement model for the relative pose rotation and translation similarly to (3-3). However, this generalisation does not hold for the location prior, because only a single sensor frame transformation is present in the measurement model function:

$$f_{i,k}^{\text{transf.}}(\mathcal{N}) = f_i^{\text{loc.}}(\mathcal{X})\, T\langle p_k^{\text{frame}}\rangle.$$

3-3-4 Summary

In this section, the measurement model functions were defined for two generally applicable parameters: a *bias* parameter and a *scaling factor* parameter. Furthermore, a special case of the bias parameter waas introduced that models the *sensor frame* transformation $T_{\mathsf{S}}^{(\mathsf{R})}$. The parameters are defined in terms of modified measurement model functions that include an extra dependency on a corresponding parameter-node. The modified pose-transformation measurement models are summarised in Table 3-2. The operators $T\langle p\rangle \in \mathcal{SE}(2)$ and $\boldsymbol{v}\langle p\rangle \in \mathbb{R}^3$ are used to construct dummy variables of appropriate dimensions for lower-dimension parameter-nodes p.

The accompanying measurement model functions for pose rotation, translation, and location prior follow from the generalisation derived in Section 2-3-2:

$$\begin{aligned} f_{i,j,k}^{\text{rot.}}(\mathcal{N}) &= R\big[f_{i,j,k}^{\text{transf.}}(\mathcal{N})\big] \in \mathcal{SO}(2), \\ f_{i,j,k}^{\text{transl.}}(\mathcal{N}) &= \mathbf{t}\big[f_{i,j,k}^{\text{transf.}}(\mathcal{N})\big] \in \mathbb{R}^2, \\ f_{i,k}^{\text{or.}}(\mathcal{N}) &= R\big[f_{0,i,k}^{\text{transf.}}(\mathcal{N})\big] \in \mathcal{SO}(2) \\ f_{i,k}^{\text{rot.}}(\mathcal{N}) &= \mathbf{t}\big[f_{0,i,k}^{\text{transf.}}(\mathcal{N})\big] \in \mathbb{R}^2. \end{aligned}$$

Note that the modified measurement model for the sensor frame parameter is only defined for models that relate two nodes (i.e., pose transformation, rotation, and translation; not location prior).

Bias:	$f_{i,j,k}^{\text{transf.}}(\mathcal{N}) = [f_{i,j}^{\text{transf.}}(\mathcal{X})]\,[T\langle p_k^{\text{bias}}\rangle]$
Scaling factor:	$f_{i,j,k}^{\text{transf.}}(\mathcal{N}) = T(\boldsymbol{\upsilon}\langle p_k^{\text{scale}}\rangle \odot \boldsymbol{\xi}_{f_{i,j}}(\mathcal{X}))$
Sensor frame:	$f_{i,j,k}^{\text{transf.}}(\mathcal{N}) = [T\langle p_k^{\text{frame}}\rangle]^{-1}\,[f_{i,j}^{\text{transf.}}(\mathcal{X})]\,[T\langle p_k^{\text{frame}}\rangle]$

Table 3-2: Modified measurement models for the *bias*, *scaling factor*, and *sensor frame* parameters.

3-4 Connectivity strategies

Connectivity strategies use a set of one or multiple parameter-nodes to model the parameter value over time (or space). As discussed in Section 3-2-2, the addition of a unique parameter-node for every constraint could result in an underdetermined system for the constraint optimum with infinitely many solutions. Therefore, alternative connectivity strategies are necessary that induce the approximation of the true parameter value.

In this section, connectivity strategies are proposed for the following scenarios:

- Constant parameters: *static strategy* in Section 3-4-1;
- Time-dependent parameters: *sliding window strategy* in Section 3-4-2 and *timely batch strategy* in Section 3-4-3;
- Space-dependent parameters: *spatial batch strategy* in Section 3-4-4.

3-4-1 Static strategy

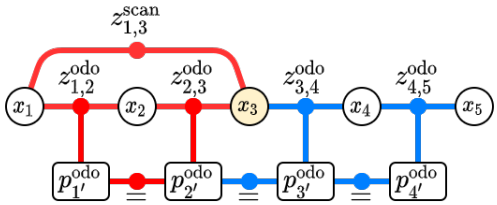
The *static strategy* connects a single parameter-node to all constraints that originate from the corresponding sensor. Static refers to the type of parameter that this connectivity strategy is best applied to parameters that are unchanging. An example of such a parameter is the sensor frame transformation for a sensor system that is rigidly secured to the robot.

The connection of a single parameter-node to multiple constraints is equivalent to connecting a unique parameter-node per constraint, with the set interconnected by equality constraints, as shown in Figure 3-2.

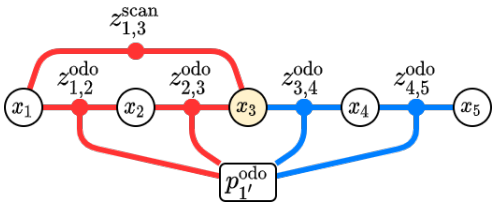
If the measurement model covariance is equal over all constraints, the error terms contributed by the connected constraints are weighted equally. Thus, the value for $p_{1'}^{\text{odo}}$ that induces the largest average cost reduction also induces the largest total cost reduction. The connectivity strategy of Figure 3-2b ensures that a value is found for $p_{1'}^{\text{odo}}$ that *on average* results in the largest constraint cost reduction. This essentially follows the law of large numbers, which states that the average of the results obtained from a large number of trials should be close to the expected value and will tend to become closer to the expected value as more trials are performed. As such, the more constraints connect the parameter-node, the better the estimate will be.

A few things are to be noted about the pose-parameter graph of Figure 3-2b:

- The scan-matching loop closure constraint $z_{1,3}^{\text{scan}}$ induces incongruence only within sub-



(a) Connectivity strategy of a unique parameter-node per constraint, with the set interconnected by equality constraints.



(b) Connectivity strategy of a single parameter-node connected to all constraints.

Figure 3-2: A pose-parameter graph where a parameter on the odometry sensor is modelled with the *static connectivity strategy*. The connection of a single parameter-node $p^{\text{odo}}_{1'}$ to constraints $\{z^{\text{odo}}_{1,2}, z^{\text{odo}}_{2,3}, z^{\text{odo}}_{3,4}, z^{\text{odo}}_{4,5}\}$ is equivalent to connecting a unique parameter-node per constraint, with the set interconnected by equality constraints, denoted by '='.

graph $\{x_1, x_2, x_3, p^{\text{odo}}_{1'}\}$ (indicated by **red** edge colours), whereas subgraph $\{x_3, x_4, x_5, p^{\text{odo}}_{1'}\}$ is left congruent (indicated by **blue** edge colours).

- Congruent subgraph $\{x_3, x_4, x_5, p^{\text{odo}}_{1'}\}$ does not contribute to the solution for $p^{\text{odo}}_{1'}$ because of the lack of conflicting measurements. Therefore, the solution for $p^{\text{odo}}_{1'}$ is determined simultaneously with nodes $\{x_1, x_2, x_3\}$ to satisfy connecting constraints $\{z^{\text{odo}}_{1,2}, z^{\text{odo}}_{2,3}, z^{\text{odo}}_{1,3}\}$ as best as possible. Furthermore, the solution that is found $p^{\text{odo}}_{1'}$ should improve the accuracy of the congruent solution for x_4 and x_5.

3-4-2 Sliding window strategy

The *sliding window strategy* estimates the dynamic behaviour of the parameter using a set of constraints of which the content moves with the current pose. In other words, the set of constraints represents a sliding window of recent constraints that are used to estimate the current value of the parameter. When the sliding window moves beyond a constraint it is disconnected from the parameter-node. To account for the change in dependency, the most up-to-date parameter estimate is embedded in the constraint value. That is, the measurement value encoded in the constraint is updated with a value that includes composition with the most up to date estimate of the parameter. This results in an identical measurement function value for the same pose-node configuration, irrespective of the contents of the parameter-node.

Consider the example posed in Figure 3-3, which shows the evolution of a graph with a (single) parameter $p^{\text{odo}}_{1'}$ that is implemented following the constant-size sliding window connectivity strategy of window size 2. Each new graph contains an extra robot pose, along with all constraints established from this new pose. A few things are to be noted about the pose-parameter graph of Figure 3-3:

- For graph instance 1 the number of available odometry constraints is less than the window size. Only in such a case is the number of connected constraints less than what is stipulated by the window size.
- The parameter node $p^{\text{odo}}_{1'}$ is identical over all graph instances; only the set of constraints it is connected to changes. When reconstructing the parameter behaviour over time, the

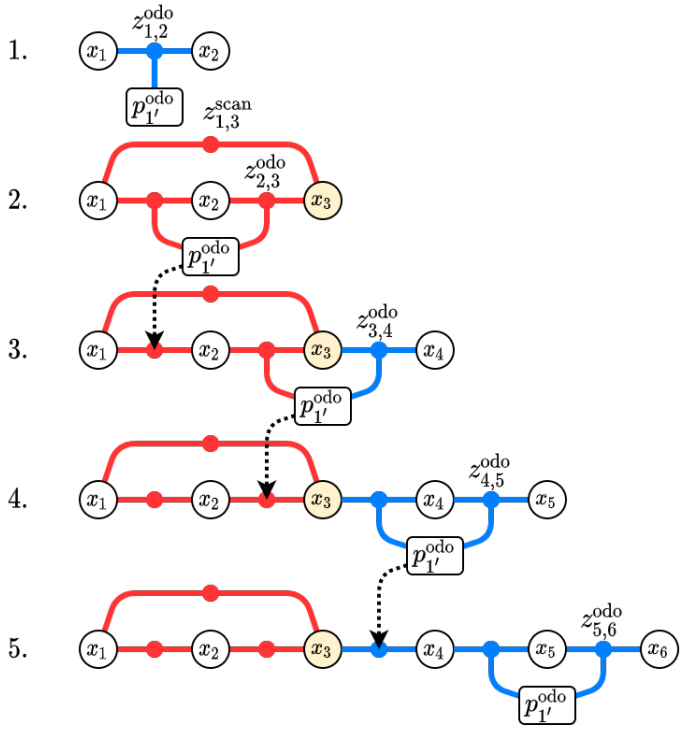


Figure 3-3: Various phases of pose-parameter graph where a parameter on the odometry sensor is modelled with the *sliding window connectivity strategy* of window size 2. Each instance shows the graph after the addition of a new robot pose, along with all relations established from this new pose. Constraints that fall beyond the sliding window, have the most up to date parameter estimate embedded in the constraint value, which is represented by the dotted arrow.

solution for $p^{\text{odo}}_{1'}$ is taken to correspond to the time instance of the most recent pose-node.

- After a constraint has moved beyond the sliding window, the most up-to-date parameter estimate is permanently embedded in the constraint value, which is represented by the dotted arrow. The accuracy of the aforementioned 'most up-to-date' parameter estimate depends on the local topology surrounding the sliding window. Three cases can be distinguished, where the parameter-node is estimated from:
  - An incongruent set of constraints, as in graph instance 2. Here, the value of $p^{\text{odo}}_{1'}$ is chosen to be cost-optimal with respect to all constraints $\{z^{\text{odo}}_{1,2}, z^{\text{odo}}_{2,3}\}$ in the sliding window.
  - A partially incongruent set of constraints, as in graph instance 3. Here, the value of $p^{\text{odo}}_{1'}$ is chosen to be cost-optimal with respect to only $z^{\text{odo}}_{2,3}$. For such a case, the parameter is estimated from a set of constraints that smaller than what is stipulated by the sliding window size.
  - A congruent set of constraints, as in graph instances 4 and 5. Here, the set of constraints is congruent, and $p^{\text{odo}}_{1'}$ is left unchanged with respect to the previous graph instance. For such a case, the value of the parameter-node does not reflect recent observations.

  The accuracy of the parameter estimate is important because its effect is permanently embedded in the constraints that fall beyond the sliding window. However, with the above-

mentioned approach, the implemented parameter estimates cannot be guaranteed to be of minimum quality (i.e., to follow from a minimum amount of incongruent constraints).

The proposed sliding window strategy falls short in the presence of a (partly) congruent sliding window set of constraints. It can be improved by dynamically adjusting the window size such that the sliding window set of constraints encompasses at least two pose-nodes at which a loop closure constraint is established. This also induces graph optimisation to be performed. After optimisation, the parameter estimate is embedded in overdue constraints and the window size is reduced to its default size. This approach should guarantee each constraint to be updated with a parameter value that is estimated from a set of constraints that is *at least* as large as stipulated by the sliding window size.

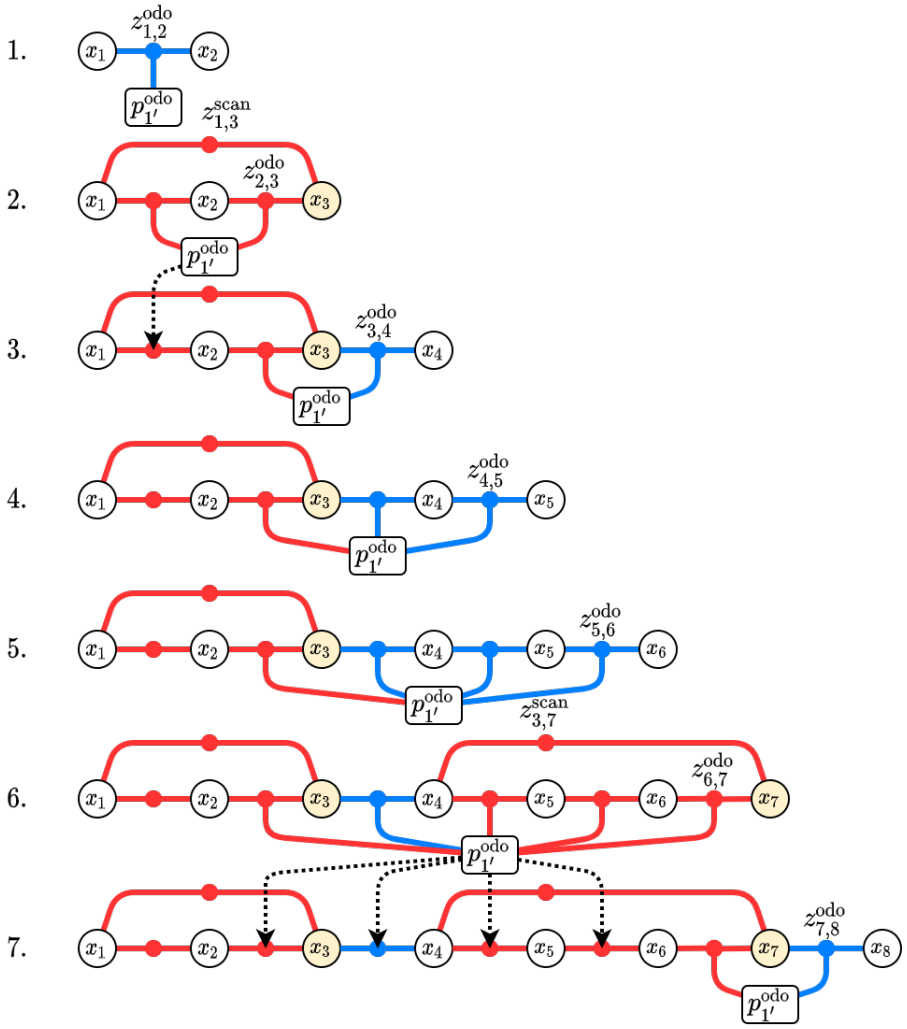


Figure 3-4: A modification to Figure 3-3: Various phases of pose-parameter graph where a parameter on the odometry sensor is modelled with the dynamic *sliding window connectivity strategy* of window size 2.

Consider the example posed in Figure 3-4, which, similarly to Figure 3-3, shows the evolution of a graph with a parameter $p^{odo}_{1'}$ that is implemented with the modified sliding window

strategy. A few things are to be noted about the pose-parameter graph of Figure 3-4:

- Up until graph instance 3, the composition is identical to that of the unmodified sliding window approach. After the addition of x_5, the sliding window would have consisted of the congruent set $\{z^{\text{odo}}_{3,4}, z^{\text{odo}}_{4,5}\}$, but with the modified strategy it is extended until loop closure is established.
- At graph instance 6, the loop closure constraint $z^{\text{scan}}_{3,7}$ is established between x_7 and x_3, and the sliding window is adjusted to include five constraints. The value of $p^{\text{odo}}_{1'}$ is chosen to be cost-optimal with respect to constraints $\{z^{\text{odo}}_{2,2}, z^{\text{odo}}_{4,5}, z^{\text{odo}}_{5,6}, z^{\text{odo}}_{6,7}\}$. Although constraint $z^{\text{odo}}_{3,4}$ is congruent, it is still able to benefit from the improved estimate due to the surrounding incongruent constraints.

Figure 3-4 shows the modified sliding window strategy is better able to cope with the shortcomings that plague the naive strategy shown in Figure 3-3. The downside of this approach is that when applied to scenarios for which loop closures are scarce, the sliding window might expand by a considerable amount. This decreases the 'recency' of the set of constraints, as the parameter value is estimated with measurements made over a longer duration of time.

3-4-3 Timely batch strategy

A *batch* strategy tries to capture the dynamic behaviour of a parameter by using a set of parameter-nodes, each of which is connected to a 'batch' of constraints. The *timely batch* approach organises these batches of constraints by timely correlation. Essentially, for each period of time, a new parameter-node is added to the graph, and constraints added within that time period are connected to this parameter. The set of parameter-nodes models the value of the parameter at a subset of all considered time instances. In contrast to the sliding window strategy of Section 3-4-2, the parameter estimate of the timely batch strategy has the ability to improve as more constraints are added to the graph. As a result, the parameter estimate covered by the first parameter-node is a function of the time.

Consider the example posed in Figure 3-5, which shows a pose-parameter graph where a parameter on the odometry sensor is modelled with the timely batch connectivity strategy of batch size 2. That is, parameter-node $p^{\text{odo}}_{1'}$ is connected to batch $\{z^{\text{odo}}_{1,2}, z^{\text{odo}}_{2,3}\}$ and $p^{\text{odo}}_{2'}$ is connected to batch $\{z^{\text{odo}}_{3,4}, z^{\text{odo}}_{4,5}\}$.

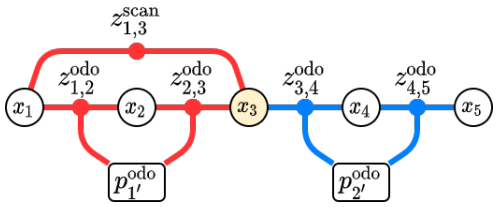


Figure 3-5: A pose-parameter graph where a parameter on the odometry sensor is modelled with the *timely batch connectivity strategy* of batch size 2.

A few things are to be noted about the pose-parameter graph of Figure 3-5:

- The batch set of constraints connected to $p^{\text{odo}}_{1'}$ is incongruent (due to loop closure constraint $z^{\text{scan}}_{1,3}$), whereas the batch set connected to $p^{\text{odo}}_{2'}$ is congruent. As such, for $p^{\text{odo}}_{2'}$, the

congruent solution exists until a conflicting constraint is established within that subgraph.

- With the availability of four odometry constraints $\{z^{\text{odo}}_{1,2}, z^{\text{odo}}_{2,3}, z^{\text{odo}}_{3,4}, z^{\text{odo}}_{4,5}\}$, and a batch size of two, only two data points (i.e., the estimates of $\{p^{\text{odo}}_{1'}, p^{\text{odo}}_{2'}\}$) are available to reconstruct the dynamics of the parameter. This is contrast to the sliding window strategy, which proposes a new parameter estimate for every time instance.

3-4-4 Spatial batch strategy

The *spatial batch* strategy is a batch strategy that organises the constraints by spatial correlation. That is, a set of parameter-nodes is introduced with each representing a region of space. Constraints are then allocated to the parameter-nodes based on the *constraint centroid*: the centroid of the translation components of the connected pose-nodes. The motivation for this is quite straightforward: under the influence of a space-varying parameter, constraints that are established from similar points in space should encounter similar parameter values.

Consider the example posed in Figure 3-6, which shows a pose-parameter graph with a parameter on the odometry sensor modelled with the spatial batch connectivity strategy of batch quantity 2.

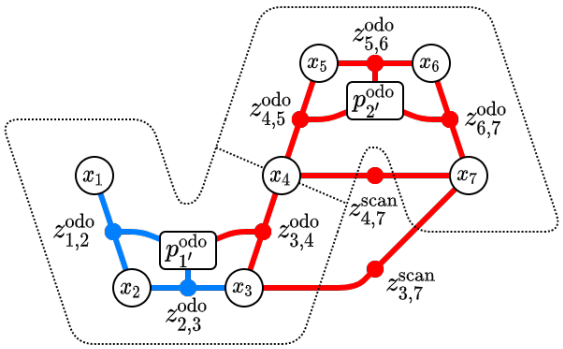


Figure 3-6: A pose-parameter graph where a parameter on the odometry sensor is modelled with the *spatial batch connectivity strategy* of batch quantity 2. The constraint-to-batch allocation is marked by the dotted line, with parameter-node $p^{\text{odo}}_{1'}$ connected to batch set of constraints $\{z^{\text{odo}}_{1,2}, z^{\text{odo}}_{2,3}, z^{\text{odo}}_{3,4}\}$ and parameter-node $p^{\text{odo}}_{2'}$ to $\{z^{\text{odo}}_{4,5}, z^{\text{odo}}_{5,6}, z^{\text{odo}}_{6,7}\}$.

A few things are to be noted about Figure 3-6:

- Parameter-node $p^{\text{odo}}_{1'}$ connects to batch set of constraints $\{z^{\text{odo}}_{1,2}, z^{\text{odo}}_{2,3}, z^{\text{odo}}_{3,4}\}$ and parameter-node $p^{\text{odo}}_{2'}$ to $\{z^{\text{odo}}_{4,5}, z^{\text{odo}}_{5,6}, z^{\text{odo}}_{6,7}\}$. The parameter-nodes are placed approximately at the spatial centre of the set of connected constraints, which represents the space components of the parameter estimate. That is, the parameter-node represents an estimate of the parameter at a location it encodes.
- The loop closure constraints $z^{\text{scan}}_{4,7}$ and $z^{\text{scan}}_{3,7}$ results in the congruent set of odometry constraints $p^{\text{odo}}_{2'}$ to $\{z^{\text{odo}}_{3,4}, z^{\text{odo}}_{4,5}, z^{\text{odo}}_{5,6}, z^{\text{odo}}_{6,7}\}$. Although the batch that connects to $p^{\text{odo}}_{2'}$ is incongruent, the batch that connects to $p^{\text{odo}}_{1'}$ is not. Similar to the case of the timely batch strategy discussed in Section 3-4-3, the parameter-node can be connected to a (partly) congruent set of constraints, which results in not all constraints being taken into account

for the parameter estimation.

The thing that sets the spatial batch connectivity strategy apart from the strategies that model time-dependent parameters is that the constraint-to-batch allocation is not deterministic. That is, a circular dependency relationship exists between the node configuration $\mathcal{N}$ and the constraint-to-batch allocation: the set of pose-nodes defines which constraints are connected to which parameter-nodes, and the connection of constraints with parameter-nodes defines the solution for the pose-nodes.

To resolve the circular dependency, an iterative strategy is proposed based on the *k-means clustering* algorithm [18] (or Lloyd's algorithm [19]), which aims to partition a set of points into k clusters with each point belonging to the cluster with the nearest mean (i.e., cluster centres or centroids). In other words, k-means clustering identifies k number of centroids, and then allocates every data point to the nearest cluster, while minimising the in-cluster sum-of-squares. The algorithm can be used with the constraint centroids to find the optimal constraint-to-batch allocation by placing the respective parameter-nodes at the calculated cluster centroids. The iterative strategy is applied until it converges to stable constraint-to-batch allocation, as highlighted by Algorithm 1.

Algorithm 1 Converging to an optimal constraint-to-batch allocation

Input: Pose-parameter graph g, constraints $\mathcal{E}$, number of clusters k
Output: Populated graph g'

procedure ALLOCATE-CONSTRAINT-TO-BATCH(g, $\mathcal{E}$, k)
  $g' \leftarrow$ OPTIMISE(g) ▷ Optimise pose-parameter graph g
  $\mathcal{C} \leftarrow k$-MEANS$\big(\{\text{centroid}(\{\mathbf{t}(x_i)$ **for** $x_i \in e\})$ **for** $e \in \mathcal{E}\}\big)$ ▷ Perform k-means clustering
  do
    $\mathcal{C}' \leftarrow \mathcal{C}$
    $\mathcal{P} \leftarrow \mathcal{P}(\mathcal{C})$ ▷ Create parameter-nodes $\mathcal{P}$ from clustering $\mathcal{C}$
    for $p \in \mathcal{P}$ **do**
      ASSIGN($p, \mathcal{E}, \mathcal{C}, g'$) ▷ Connect constraints $\mathcal{E}$ to node p based on clustering $\mathcal{C}$ in g'
    end for
    $g' \leftarrow$ OPTIMISE(g') ▷ Optimise updated graph g'
    $\mathcal{C} \leftarrow k$-MEANS$\big(\{\text{centroid}(\{\mathbf{t}(x_i)$ **for** $x_i \in e\})$ **for** $e \in \mathcal{E}\}\big)$ ▷ Perform k-means clustering
  while $\mathcal{C} \neq \mathcal{C}'$
  return g' ▷ Return the solution g' containing k parameter-nodes
end procedure

Note that the accuracy of the constraint-to-batch allocation is directly dependent on the accuracy of the set of poses. Adding a set of parameter-nodes based on the spatial batch strategy is therefore performed *after* the graph has been completed, at which point all information has been gathered and the pose accuracy is likely to be highest. Application of the spatial batch strategy can thus be considered a *post-processing step*. Besides, there is no meaningful continuity of the clustering between consecutive graph instances, which complicates any temporal analysis of its behaviour.

3-5 Summary

Pose-Parameter Graph Optimisation (PPGO) is an extension of Pose Graph Optimisation (PGO) that includes a set of parameter-nodes $\mathcal{P}$ in the optimisation problem over $\mathcal{N} =$

$\mathcal{X} \cup \mathcal{P}$. The set of parameter-nodes model a parameter that influences the accuracy of the measurement models. The extended problem formulation is given by

$$\mathcal{N}^* = \arg\min_{\mathcal{N}} \sum_{\mathcal{I}} F_{\mathcal{I}}(\mathcal{N}),$$

where $F_{\mathcal{I}}(\mathcal{N}) = \|f_{\mathcal{I}}(\mathcal{N}) \boxminus z_{\mathcal{I}}\|^2_{\Sigma_{\mathcal{I}}}$ comprises the extended measurement model function $f_{\mathcal{I}}(\mathcal{N})$ that defines a dependency on the set of parameter-nodes. For the purpose of this research, the generally applicable *bias* and *scaling factor* parameters are modelled, along with a special case of the bias parameter: the *sensor frame* parameter. The measurement models are defined in Table 3-2.

Because the parameters are not directly measured, but rather deduced from systematic inconsistencies between the measurement models and the actual measurement values, PPGO is not able to provide an estimate for the parameter value at every instance (unless extra constraints are imposed). Therefore, alternate connectivity strategies are required that strategically connect a set of parameter-nodes that is not unique for every time instance. The following connectivity strategies are proposed for the following scenarios:

- Constant parameters are modelled with the *static strategy*;
- Time-dependent parameters are modelled with the *sliding window strategy* and the *timely batch strategy*;
- Space-dependent parameters are modelled with the *spatial batch strategy*.

Chapter 4

Simulation framework

In order to assess the performance of Pose-Parameter Graph Optimisation (PPGO), a simulation framework is set up that is used to construct pose-parameter graphs similar to what would be constructed by a real robot. This simulation framework incrementally builds a graph based on an input trajectory and sensor models with parametric dependencies. The framework includes analytic tools that aid in the verification of the simulations. Unfortunately, no real-world experiments have been performed to validate the simulations. The analytic tools are used to assess the performance of PPGO, the results of which are presented in Chapter 5.

This chapter covers the following:

- Section 4-1 *'Overview'* describes the simulation framework configuration, and evolution and optimisation procedures.
- Section 4-2 *'Trajectories'* introduces the two trajectories used in this research: a procedurally generated Manhattan grid and a dataset obtained at the Intel Research Lab in Seattle.
- Section 4-3 *'Summary'* summarises the content of this chapter.

4-1 Overview

The software implementation of the simulation framework is thoroughly discussed in Appendix A. To summarise: it consists of a `python` framework that utilises the General (Hyper) Graph Optimisation framework (or `g2o` [11]) for optimisation. Graphs are constructed via simulation, written to a `.g2o` file, read by the `g2o` library, its solution written to another `.g2o` file, which is finally interpreted by the simulation framework for further use.

The advantage of relying on simulations to investigate the performance is that a ground truth is available. The ground truth can be used to assess the accuracy of the solution, both in terms of the pose accuracy and parameter accuracy.

During simulation, two graphs are constructed in parallel by two sub-simulations:

1. The *truth graph* g_{truth}, which represents the ground truth. The measurements in this graph are unperturbed my measurement noise, and therefore exactly satisfy the measurement model. The result is a fully congruent truth graph that is always in its optimal state with $F_{\text{ppgo}}(\mathcal{N}^{\text{truth}}) = 0$.
2. The *estimate graph* $g_{\text{est.}}$, which represents what the robot would construct from the incoming measurements. The measurements are derived from the truth graph but are perturbed my measurement noise. As such, the estimate graph is likely to be incongruent with $F_{\text{ppgo}}(\mathcal{N}^{\text{est.}}) \neq 0$.

Between the two graphs, the set of measurements is identical in size. As a result, both graphs establish the same number of constraints and introduce the same number of pose-nodes. The modelling of any parameters, however, is not necessarily identical. Whereas g_{truth} is in its optimal state with a unique parameter-node per time instance, the corresponding estimate graph $g_{\text{est.}}$ would be under-constrained (as discussed in Section 3-4-1). As such, $g_{\text{est.}}$ is configured with a different parameter connectivity strategy. Furthermore, different sensor models are utilised per sub-simulations to account for modelling deficiencies in the estimate sub-simulations.

4-1-1 Configuration

The configuration of a simulation comprises the following components:

1. *Trajectory.* The robot trajectory consists of a set of poses $\{x_i^{\text{traj.}} \in \mathcal{SE}(2)\}$ from which the pose-parameter graphs are derived. It is a simulation input and serves as the ground-truth path of the robot. As such, after simulation, the truth graph will perfectly match the input trajectory with its pose-nodes. The trajectories that are used in this research are discussed in Section 4-2.
2. *Parameters.* The influence of any parameters is modelled as a set of parameter-nodes in the ground truth graph g_{truth} that is uniquely added to every constraint of a specific sensor. This set of parameter-nodes represents the ground truth parameter value over time (or space) and is referred to as the *input parameter*. The corresponding estimate graph $g_{\text{est.}}$ models the parameter by adding its own set of parameter-nodes according to a defined connectivity strategy. Because the addition of a unique parameter-node per constraint would result in an under-constrained optimisation problem, the set of parameter-nodes in $g_{\text{est.}}$ is inherently different (i.e., smaller) than that of g_{truth}. The parameter estimate that is reconstructed from the set of parameter-nodes is referred to as the *output parameter*.
3. *Sensors.* A sensor generates measurements from the input trajectory and input parameters. This operation is similar to what is defined by the measurement models $(f_{\mathcal{I}}, \Sigma_{\mathcal{I}})$ that are introduced in Section 2-3-2. For the truth graph $g_{\text{truth.}}$, measurements are directly generated by g_{truth} and encoded in the truth graph with truth measurement model $(f_{\mathcal{I}}^{\text{truth}}, \Sigma_{\mathcal{I}}^{\text{truth}})$. For the estimate graph $f_{\mathcal{I}}^{\text{est.}}$, the same measurements are perturbed by measurement noise as described by $\Sigma_{\mathcal{I}}^{\text{truth}}$. The perturbed measurements are encoded in the $g_{\text{est.}}$ along with an estimate measurement model $(f_{\mathcal{I}}^{\text{est.}}, \Sigma_{\mathcal{I}}^{\text{est.}})$. Note that the estimate measurement model $(f_{\mathcal{I}}^{\text{est.}}, \Sigma_{\mathcal{I}}^{\text{est.}})$ is not necessarily accurate for how the corresponding measurements are generated.
4. *Random Number Generator (RNG).* RNGs are used to simulate the uncertainties that oc-

cur in real-life experiments. Furthermore, they are used to procedurally generate 'random' trajectories. In order to maintain reproducibility of simulations, all RNGs are seeded.

The following set of seeded RNGs are used:

- *Sensor noise.* As explained in Section 2-1-3, the measurement noise is assumed to be zero-mean Gaussian. For each sensor, a separate seeded RNG is used.
- *Constraint generation.* Odometry constraints are established between each consecutive pair of robot poses of the input trajectory. Loop closure constraints, on the other hand, are only established when the exteroceptive sensor successfully finds a scan match between an incoming measurement and a previously recorded measurement. Modelling the probability of a successful scan match as a function of nearby poses requires precise modelling of environment features, which is well beyond the scope of this simulation. Therefore, for each robot pose, a set of eligible loop closure constraints is proposed based purely on the translation component of the current pose and its proximity to previous poses. The choice of loop closure constraint from this set is determined with a separate seeded RNG.
- *Path generation.* Some trajectories are procedurally generated, which requires a separate seeded RNG.

Both the g_{truth} and $g_{\text{est.}}$ are initialised with starting pose $x_0 = T(0,0,0) \in \mathcal{SE}(2)$. This pose is fixed relative to the inertial coordinate system O and, as a result, all poses are indirectly defined relative to x_0. Furthermore, the equivalence of x_0 in both graphs allows for comparisons to be made in terms of divergence from an identical starting point.

The initialisation of the input parameter-nodes is considered a simulation input, whereas the output parameter-nodes are initialised to a default value (usually zero), as its true value is considered unknown within the estimate sub-simulation.

4-1-2 Evolution

To build up the desired graphs, the simulation is continuously evolved by adding new robot poses and connecting constraints. The addition of a constraint relating poses $\mathcal{X}_{\mathcal{I}}$ using a given sensor is done as follows:

1. A *ground truth measurement value* $z_{\mathcal{I}}^{\text{truth}}$ is derived by applying the measurement model of the corresponding sensor in the ground truth sub-simulation on the relevant pose-nodes and sensor parameter-nodes $\mathcal{N}_{\mathcal{I}}^{\text{truth}} = \mathcal{X}_{\mathcal{I}}^{\text{truth}} \cup \mathcal{P}_{\mathcal{I}}^{\text{truth}}$. That is,

$$z_{\mathcal{I}}^{\text{truth}} = f_{\mathcal{I}}^{\text{truth}}(\mathcal{N}_{\mathcal{I}}^{\text{truth}}).$$

2. A constraint is established in the ground truth graph of the form

$$F_{\mathcal{I}}(\mathcal{N}_{\mathcal{I}}^{\text{truth}}) = \| f_{\mathcal{I}}^{\text{truth}}(\mathcal{N}_{\mathcal{I}}^{\text{truth}}) \boxminus z_{\mathcal{I}}^{\text{truth}} \|^2_{\Sigma_{\mathcal{I}}^{\text{truth}}}.$$

3. An *estimate measurement value* $z_{\mathcal{I}}^{\text{est.}}$ is derived by perturbing the ground truth measurement $z_{\mathcal{I}}^{\text{truth}}$ with additive zero-mean Gaussian noise $\boldsymbol{\epsilon} \sim \mathtt{N}(\mathbf{0}, \Sigma_{\mathcal{I}}^{\text{truth}})$. That is,

$$z_{\mathcal{I}}^{\text{est.}} = z_{\mathcal{I}}^{\text{truth}} \boxplus \boldsymbol{\epsilon}.$$

4. A constraint is established in the estimate graph of the form

$$F_{\mathcal{I}}(\mathcal{N}_{\mathcal{I}}^{\text{est.}}) = \| f_{\mathcal{I}}^{\text{est}}(\mathcal{N}_{\mathcal{I}}^{\text{est.}}) \boxminus z_{\mathcal{I}}^{\text{est.}} \|^2_{\Sigma_{\mathcal{I}}^{\text{est.}}},$$

where $\mathcal{N}_{\mathcal{I}}^{\text{est.}} = \mathcal{X}_{\mathcal{I}}^{\text{est.}} \cup \mathcal{P}_{\mathcal{I}}^{\text{est.}}$. Note that the measurement model that is embedded in the constraint does not necessarily match the model that the corresponding measurement was generated with, which is the result of modelling deficiencies.

The procedure of adding a constraint is summarised in Algorithm 2.

Algorithm 2 Adding a constraint between nodes $\mathcal{I}$

Input: Node indices $\mathcal{I}$, truth graph g_{truth} and sensor $(f_{\mathcal{I}}^{\text{truth}}, \Sigma_{\mathcal{I}}^{\text{truth}})$, estimate graph $g_{\text{est.}}$ and sensor $(f_{\mathcal{I}}^{\text{est.}}, \Sigma_{\mathcal{I}}^{\text{est.}})$

procedure ADD-CONSTRAINT($\mathcal{I}, g_{\text{truth}}, (f_{\mathcal{I}}^{\text{truth}}, \Sigma_{\mathcal{I}}^{\text{truth}}), g_{\text{est.}}, (f_{\mathcal{I}}^{\text{est.}}, \Sigma_{\mathcal{I}}^{\text{est.}})$)
- $z_{\mathcal{I}}^{\text{truth}} \leftarrow f_{\mathcal{I}}^{\text{truth}}(\mathcal{N}_{\mathcal{I}}^{\text{truth}})$ ▷ Calculate truth measurement
- $F_{\mathcal{I}}^{\text{truth}} \leftarrow \left\| f_{\mathcal{I}}^{\text{truth}}(\mathcal{N}_{\mathcal{I}}^{\text{truth}}) \boxminus z_{\mathcal{I}}^{\text{truth}} \right\|^2_{\Sigma_{\mathcal{I}}^{\text{truth}}}$ ▷ Construct truth constraint
- g_{truth}.add($F_{\mathcal{I}}^{\text{truth}}$) ▷ Add constraint to truth graph
- $\boldsymbol{\epsilon} \sim \mathtt{N}(\mathbf{0}, \Sigma_{\mathcal{I}}^{\text{truth}}$ ▷ Draw measurement noise vector
- $z_{\mathcal{I}}^{\text{est.}} \leftarrow z_{\mathcal{I}}^{\text{truth}} \boxplus \boldsymbol{\epsilon}$ ▷ Calculate estimate measurement
- $F_{\mathcal{I}}^{\text{est.}} \leftarrow \left\| f_{\mathcal{I}}^{\text{est}}(\mathcal{N}_{\mathcal{I}}^{\text{est.}}) \boxminus z_{\mathcal{I}}^{\text{est.}} \right\|^2_{\Sigma_{\mathcal{I}}^{\text{est.}}}$ ▷ Construct estimate constraint
- $g_{\text{est.}}$.add($F_{\mathcal{I}}^{\text{est.}}$) ▷ Add constraint to estimate graph

end procedure

From starting position $x_0 = T(0, 0, 0)$, the simulation uses the following procedural simulation step to build up its graphs. Consider x_i to be the current pose. The simulation step establishes one or more of the following constraints:

1. *Odometry constraint.* The odometry measurement defines the location of new pose x_j with respect to x_i. In the truth graph, the pose $x_j^{\text{truth}} = x_j^{\text{traj.}}$ is drawn from the input trajectory. Following the procedure of adding a new constraint described above, the pose $x_j^{\text{est.}}$ is set such that $f_{\mathcal{I}}^{\text{est.}}(x_i^{\text{est.}}, x_j^{\text{est.}}, \mathcal{P}_{\mathcal{I}}) = z_{\mathcal{I}}^{\text{est.}}$. That is, the estimated pose is placed exactly in line with the measurement value. As such, the constraints in both graphs will contribute a cost term of zero.

2. *Loop closure constraint.* A loop closure constraint is established after a robot has returned to a previously visited location. Based on the translation proximity in the ground truth graph of the current pose with previous poses, a set of eligible loop closure constraints is proposed. After consulting the constraint generation RNG, a loop closure constraint is added in both graphs based on the ground truth graph. Note that the poses related by the loop closure constraint might not be close in the estimate graph.

3. *Proximity constraint.* Similar to loop closure constraints, proximity constraints are established between poses that are close in the truth graph. The difference is that proximity constraints are usually established from *short term memory* between poses separated by only a few odometry constraints, whereas loop closure constraints are established from *long term memory* between poses that are established at potentially vastly different time instances.

After each simulation step, both the truth and estimate graph states are saved. Therefore, after the simulation is completed, the entire graph history is available for analysis.

4-1-3 Optimisation

During simulation, the estimate graph is continuously optimised to yield the instantaneous optimal solution. However, this optimisation step is not necessary for every simulation step. As mentioned above, the new pose introduced at every step is placed such that the relative pose transformation satisfies the odometry measurement model. That is, the odometry constraint contributes an error term that has an instantaneous value of zero. Therefore, for a simulation step that only establishes an odometry constraint, the current pose is placed in a graph tail, and the addition of an odometry constraint will not result in a different solution for the nodes that were present in the subgraph before addition. Accordingly, optimisation is only performed after a loop closure constraint and/or proximity constraint has been established and the current subgraph has been made inconsistent.

Graph ptimisation is performed using `g2o` [11]. This framework is widely used, fast, and allows for the definition of custom hyper-edges, which is exactly what is required for using the measurement models and accompanying error functions discussed in Section 3-3. Although `g2o` seems to work consistently for Pose Graph Optimisation (PGO), it does not always converge to an optimal solution after parameter-nodes have been added to form a PPGO problem. More specifically, a solution was occasionally found with a cost function value that was significantly higher than that of the same graph without the parameter-nodes. This is inherently non-optimal, as the same (and in this case *lower cost*) solution could be obtained by matching the solution of the graph without parameters while negating the presence of parameter-nodes by re-initialising them to their default value (zero and one for additive and multiplicative parameters, respectively). Note that optimal refers to the solution that minimises the cost function of the non-linear least-square PPGO problem; not the solution that minimises all error metrics.

In an effort to filter these non-optimal results, a graph is optimised according to Algorithm 3. The procedure contains the following steps:

1. All parameter-nodes are re-initialised to their default value. This ensures that the initial condition of the optimisation step is as close to standard PGO as possible. That is, the cost function has convex properties at the initial condition, which should aid convergence [15].
2. The solution out of `g2o` is checked with respect to a cost threshold. If the solution has a cost function value beyond the threshold, it is not accepted. Of course, the cost threshold should be chosen carefully as not to exclude the optimal solution.
3. If `g2o` is not able to find a solution with a cost value within the threshold, all parameter-nodes are fixed at their initial value, which results in a regular PGO problem. Since PGO is proven to converge to a value within the threshold, its solution is by definition more optimal than what `g2o` can come up with.

   All parameter-nodes are un-fixed for the next optimisation step, and the solution is accepted.

For this research, it was found that the cost threshold of *twice the cost at the previous simulation step* was sufficient to exclude non-converging solutions, while still allowing for an optimal solution to be found. Solving the issue of non-convergence for PPGO generalises this research, as the conclusions should still be valid for other optimisers.

Algorithm 3 Filtering non-optimal solutions

Input: Unoptimised pose-parameter graph g_0, cost threshold α
Output: Optimised graph g_1 or g_2

function ADD-CONSTRAINT(g_0, α)
  for $p \in \mathcal{P}(g_0)$ **do**
    re-initialise(p) ▷ Re-initialise all parameter-nodes of g_0
  end for
  if α not given **then**
    $\alpha \leftarrow \text{cost}(g_0)$ ▷ Set the cost threshold to that of g_0 prior to optimisation
  end if
  $g_1 = \texttt{g2o}.\text{optimise}(g_0)$ ▷ Optimise g_0 with initialised parameter-nodes
  if $\text{cost}(g_1) < \alpha$ **then** ▷ Check if g_1 is optimal
    return g_1 ▷ Accept solution g_1
  end if
  for $p \in \mathcal{P}(g_0)$ **do** ▷ $\text{cost}(g_1) > \alpha \implies g_1$ is not optimal
    fix(p) ▷ Fix all parameters at the re-initialised value
  end for
  $g_2 = \texttt{g2o}.\text{optimise}(g_0)$ ▷ Optimise g_0 with fixed and initialised parameter-nodes
  assert $\text{cost}(g_2) < \alpha$ ▷ Assert g_2 to be optimal
  for $p \in \mathcal{P}(g_2)$ **do**
    un-fix(p) ▷ Un-fix all parameter-nodes of g_2
  end for
  return g_2 ▷ Accept solution g_2
end function

4-2 Trajectories

The performance of PPGO is assessed based on the following two trajectories [14]:

1. *Manhattan grid* [25] (see Section 4-2-1), which fulfils the function of ideal case, with shorts loops and numerous possibilities for loop closing. Furthermore, it is easily interpretable with a grid-like trajectory.
2. *Intel Research Lab dataset* [14] (see Section 4-2-2), which is derived from real data. This trajectory contains an long initial loop and should provide interesting insights, especially for fluctuating parameters.

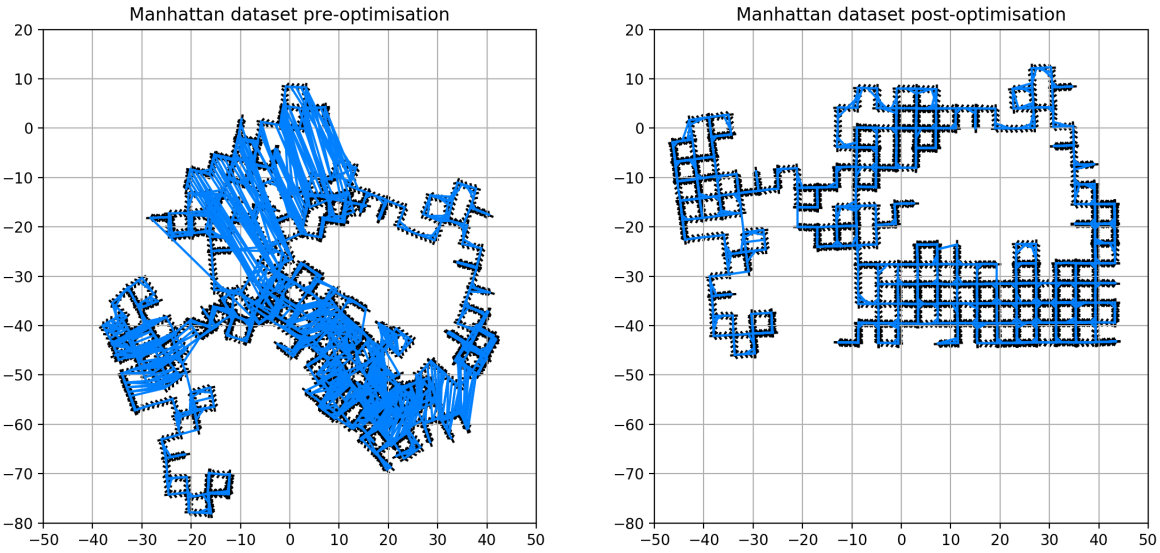
Both graphs are discussed in subsequent subsections.

4-2-1 Manhattan grid

The Manhattan grid is not a dataset per se, but more a *method* for a procedurally generated path. The path moves in a straight line for a given number of steps, after which it takes either a ninety-degree left or right turn, as though a robot was travelling the city streets in upper Manhattan [25]. This makes it easy to visually appraise a map: all corners should be perpendicular and paths should overlap exactly.

Figure 4-1 shows one such instance, consisting of 3500 nodes and 5453 constraints [4].

For some cases, the path might be distorted to ensure that a non-zero y-translation is likely

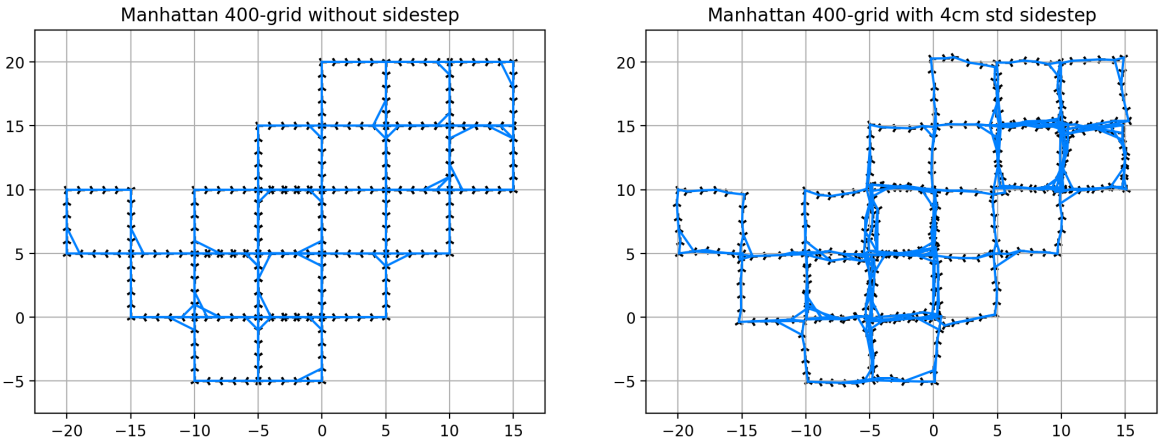


(a) Unoptimised pose graph

(b) Optimised pose graph

Figure 4-1: Pose graph derived with the Manhattan method for 3500 nodes [4]. Figure **(a)** shows the pose graph that is constructed from the raw data [4]. Figure **(b)** shows the graph after optimisation.

to occur at each step. This is done by generating for each step a small translation 'sidestep' value from a zero-mean Gaussian distribution, and adding the average of the last two sidestep values to the next step translation. The result of which is shown in Figure 4-2



(a) Without sidestep

(b) With sidestep

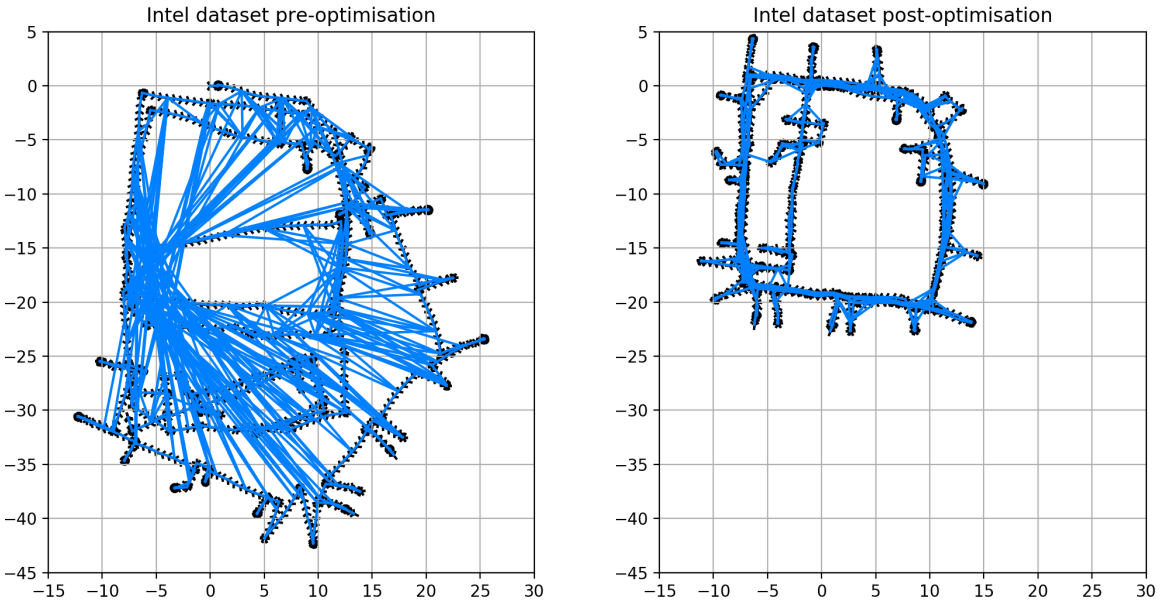
Figure 4-2: Comparison of two Manhattan 300-grid trajectories with and without sidestep. Figure **(a)** shows the graph without sidestep, and Figure **(b)** shows the same graph with sidestep drawn from a zero-mean Gaussian distribution with a standard deviation of 4 cm.

4-2-2 Intel dataset

The Intel dataset consists of a pose graph obtained by processing the raw measurements from wheel odometry and laser range finder, acquired at the Intel Research Lab in Seattle [4]. A solution to this dataset in the form of an occupancy grid is shown in Figure 4-3.

Figure 4-3: Occupancy grid derived from the data set of Intel Research Lab in Seattle [14].

The pose graph that is derived from the raw data is shown in Figure 4-4a and consists of 1228 poses and 1483 constraints. The graph is unoptimised, which is indicated by the (usually short) loop closure constraints stretching all over the graph domain. The optimised graph is shown in Figure 4-4b.



(a) Unoptimised pose graph

(b) Optimised pose graph

Figure 4-4: Pose graph derived from the Intel dataset [4]. Figure **(a)** shows the pose graph that is constructed from the raw data [4]. Figure **(b)** shows the graph after optimisation.

What is interesting about the Intel dataset is that, during the first traversal of the ground floor (consisting of 166 steps), no loop closures are established until the robot reaches its starting point. During this time, the current robot pose exists in a graph tail; a scenario in which PPGO can showcase its strengths by potentially nullifying any systematic components of the drift.

Since the Intel dataset already contains poses and constraints that are based on measurements without the presence of any systematic deviations, its measurement cannot directly be used to assess the performance of PPGO. The poses of the optimised graph are taken as a ground truth trajectory and measurements that include deviations are generated as per Section 4-1-2.

4-3 Summary

The simulation framework developed for the purpose of investigation the performance of PPGO constructs two graphs in parallel:

- The *truth graph*, which is unperturbed by sensor noise;
- The *estimate graph*, which represents what a robot would construct in a realistic scenario from incoming measurements.

The framework is developed in `python` and utilises the `g2o` framework for graph optimisation.

Two trajectories are used to assess PPGO's performance:

- The *Manhattan grid*, which is a procedurally generated path that is easy to visually appraise. The Manhattan grid comprises a grid-like path that is similar to a path taken in the city streets of upper Manhattan;
- The *Intel dataset*, which is derived from real measurements performed at the Intel Research Lab in Seattle. The graph contains a large traversal of its environment, which results in graph segment with only a small amount of loop closure constraints.

Chapter 5

Results

The goal of Pose-Parameter Graph Optimisation (PPGO) is to estimate the value of parameters that influence the measurement models (in addition to robot trajectory represented by a set of poses). This ability should allow PPGO to be applied to scenarios that traditional Pose Graph Optimisation (PGO) struggles with. The performance of PPGO depends on its ability to accurately estimate the set of poses (referred to as *pose accuracy*) and its ability to estimate the parameter value over time or space (referred to as *tracking accuracy* or *parameter accuracy*).

In this chapter, the performance of PPGO is assessed for a variety of scenarios. Furthermore, its performance is compared to that of traditional PGO to highlight the benefits of simultaneous parameter estimation. First, the feasibility of the parameter implementations in terms of the modified measurement models are examined. Second, the ability of the connectivity strategies to track both time-dependent and space-dependent parameter fluctuations is examined.

This chapter covers the following:

- Section 5-1 *'Simulation'* introduces the simulation used to generate the results.
- Section 5-2 *'Constant parameters'* examines the performance of PPGO for estimation of constant parameter, which are best modelled with the *static connectivity strategy*.
- Section 5-3 *'Time-dependent parameters'* examines the performance of PPGO for estimation of time-dependent parameters, for which two connectivity strategies are proposed: the *sliding window strategy* and the *timely batch strategy*.
- Section 5-4 *'Space-dependent parameters'* examines the performance of PPGO for estimation of space-dependent parameters, which are best modelled with the proposed *spatial batch approach*.
- Section 5-5 *'Summary'* summarises the content of this chapter.

5-1 Simulation

The performance of PPGO is examined in simulation, as discussed in Chapter 4. Simulations are run according to the following procedure:

1. First, an input parameter is defined to influence the measurement of the odometry sensor. The odometry sensor is chosen because it generates the most constraints, and its modelling deficiencies are therefore most likely to influence the solution quality. Furthermore, it allows for all connectivity strategies to be tested without apprehension that constraint availability might be a bottleneck. The value of the input parameter is kept constant or is varied with time or space, depending on the scenario. In case of a varying parameter value, a new parameter-node is added to the truth graph for every time instance.
2. A set of PGO simulations are run that are configured *without* means of modelling the unknown parameter value. The result of these simulations serves as an indication of the performance of traditional PGO when provided with measurements that do not match the measurement models encoded in the graph.
3. A second set of PPGO simulations are run that are configured *with* a matching parameter model. The connectivity strategy that defines the set of parameter-nodes that are added to the estimate graph is varied depending on the scenario. The result of these simulations should highlight the abilities of PPGO to correctly reconstruct the input parameter when provided with the correct measurement models. Furthermore, it should show potential performance improvements as well as limitations compared to traditional PGO.

Each simulation is run 20 times in a Monte Carlo-type fashion with a different sensor noise seed in order to model the probability of different outcomes in the process that cannot easily be predicted due to the intervention of random variables.

Sensors	
Odometry	
Sensor type:	Wheel encoder
Measurement type:	$\mathcal{SE}(2)$
$f_{i,j}^{\text{odo}}(\mathcal{X})$:	$f_{i,j}^{\text{transf.}}(\mathcal{X}) = x_i^{-1} x_j$
$\Sigma_{i,j}^{\text{odo}}$:	$\text{diag}(400, 400, 400)$
Loop closure / proximity	
Sensor type:	LiDAR scan-matching
Measurement type:	$\mathcal{SE}(2)$
$f_{i,j}^{\text{LiDAR}}(\mathcal{X})$:	$f_{i,j}^{\text{transf.}}(\mathcal{X}) = x_i^{-1} x_j$
$\Sigma_{i,j}^{\text{LiDAR}}$:	$\text{diag}(8000, 8000, 12000)$
Location prior	
Sensor type:	GPS
Measurement type:	$\mathbb{R}^2$
$f_i^{\text{GPS}}(\mathcal{X})$:	$f_i^{\text{loc.}}(\mathcal{X}) = \mathbf{t}_i$
Σ_i^{GPS}:	$\text{diag}(1, 1)$

Table 5-1: Simulation configuration, where $\text{diag}(\dots)$ denotes a diagonal matrix.

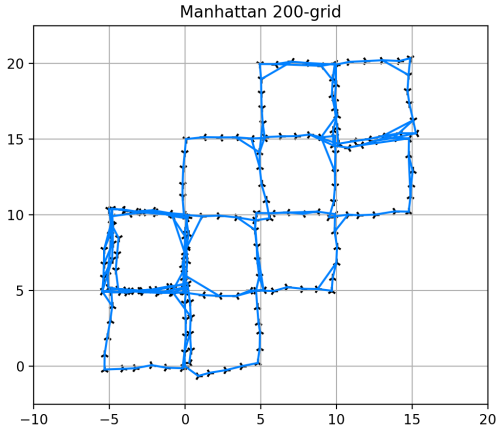
The simulation configuration is detailed in Table 5-1, which comprises the following sensors:

- *Wheel encoder* used to establish for odometry, with a measurement standard deviation of 5 cm and 0.05 rad for translation and rotation, respectively.

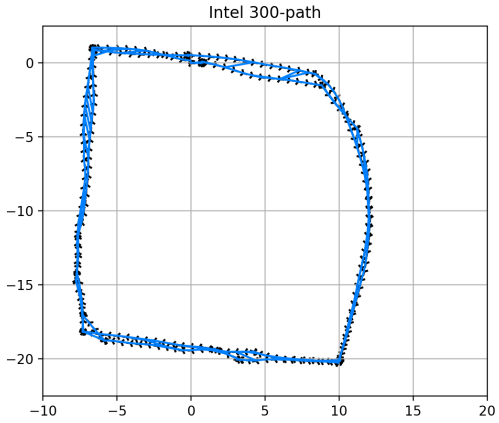
- *LiDAR scan-matching system* used to establish loop closure constraints, with a measurement standard deviation of 1.1 cm and 0.009 rad for translation and rotation, respectively.

- *GPS system* used to establish location prior constraints, with a translation measurement standard deviation of 1 m.

For each simulation, the two trajectories shown in Figure 5-1 are considered (see Section 4-2). The Manhattan path has a step size of 1 m, a grid size of 5 steps, and a sidestep standard deviation of 4 cm to ensure that a non-zero y-translation is likely to be present.



(a) Manhattan grid graph $(200, 271)$: 200 pose nodes, 199 wheel odometry constraints, 62 LiDAR loop closure / proximity constraints, 10 GPS constraints.



(b) Intel path graph $(300, 403)$: 300 pose nodes, 299 wheel odometry constraints, 94 LiDAR loop closure / proximity constraints, 10 GPS constraints.

Figure 5-1: Simulation trajectories. Figure **(a)** shows the Manhattan grid graph $(200, 271)$. Figure **(b)** shows the Intel path graph $(300, 403)$.

5-2 Constant parameters

First, PPGO is applied to the simplest case: *constant parameters*. As discussed in Section 3-4-1, constant parameters are best modelled with the *static connectivity strategy* with a single parameter-node connected to all constraints of the corresponding sensor.

Because of the abundance of parameter implementations available (e.g. a bias can be applied on any combination of components of a pose transformation measurement, namely x or y or θ or (x, y) or (x, θ) or (y, θ) or (x, y, θ)), only case with the highest plausible dimensionality is explicitly examined; the rhetoric being that if PPGO is able to reconstruct a parameter applied over all measurement components, it is also able to reconstruct a parameter applied over just a subset. The highest-dimension parameter results in the most measurement inconsistencies and offers the most degrees of freedom in the optimisation problem. As such, this parameter implementation is considered worst-case.

5-2-1 Bias

A baseline is established by running a simulation without the influence of any parameters. The corresponding PGO solution represents the best-case performance for the defined simulation configuration. The PPGO sub-simulation is configured with a parameter-node $p^{\text{bias}(x,y,\theta)} \in \mathcal{SE}(2)$ to test whether its estimate approximates the zero bias value of $T(\mathbf{0}) \in \mathcal{SE}(2)$. The result on the Manhattan path is shown in Figure 5-2, in which the Absolute Trajectory Error (ATE) and parameter estimate are plotted for an increasing graph size.

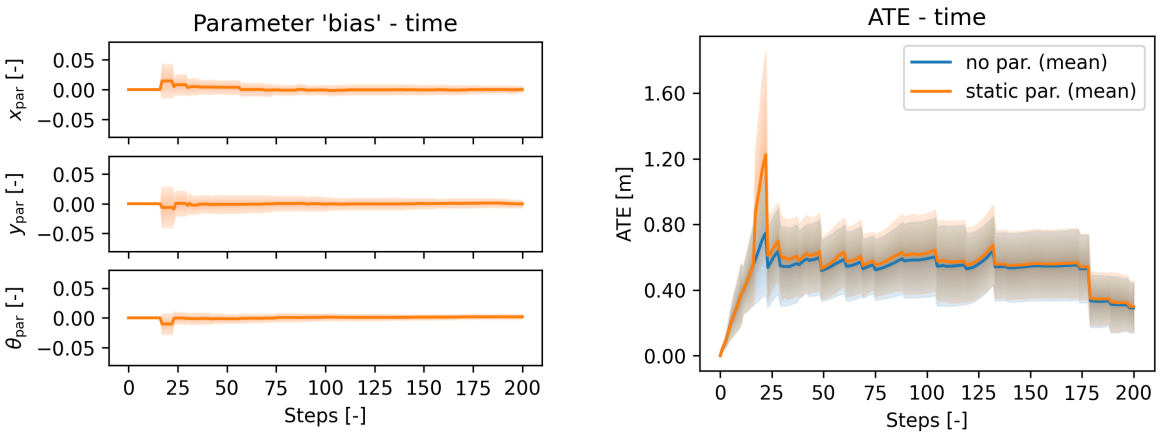


Figure 5-2: Manhattan path unparameterised with a static output parameter on the odometry sensor: **PGO graph without parameters (blue)** versus **PPGO graph with static parameter (orange)**.

The following conclusions can be drawn:

- *Tracking accuracy*: the vector representation of parameter-node $p^{\text{bias}(x,y,\theta)} \in \mathcal{SE}(2)$ is found to stay close to the true value of $\mathbf{0} \in \mathbb{R}^3$. The least accurate parameter estimate is derived at the moment the first loop closure constraint is established. The standard deviation of the estimate decreases with the number of constraints added.
- *Pose accuracy*: the ATE over time is almost identical for the PGO and PPGO solutions (apart from the small error spike at the beginning of the simulation, which can be attributed to an inaccurate parameter estimate). This corresponds with the correct estimation of the parameter value, where an estimate of zero effectively negates its effect on the measurement model. The result is a similar optimisation problem with a similar solution.

  This verifies that the addition of any redundant parameter-node components in the PPGO graph does not have a significant adverse effect on the solution quality.

The pose accuracy can be summarised using a single value: the *average ATE over the period of graph construction*. This value represents the average pose error one would encounter for this given scenario. The average ATE is similar for both graphs: 0.540 m and 0.566 m for PGO and PPGO, respectively. The average ATE for all parameter implementations are summarised in Table 5-2, which makes for easy comparison with the baseline performance.

Table 5-2 shows that the PPGO performance is relatively consistent over the period of graph construction in terms of ATE, irrespective of the parameter implementation. More specifically, the PPGO solution always is within 26 % of the baseline on the Manhattan path, and within

Simulation	Manhattan (200, 271)		Intel (300, 403)	
	no par.	static par.	no par.	static par.
Baseline	0.540	0.566	0.826	0.916
bias$(x) = 0.1$	0.955	0.570	1.641	0.834
bias$(y) = 0.1$	0.651	0.556	1.048	0.830
bias$(\theta) = 0.1$	1.126	0.632	6.219	0.915
bias$(x, y) = (0.1, 0.1)$	1.098	0.582	1.569	0.836
bias$(x, \theta) = (0.1, 0.1)$	1.432	0.645	6.434	0.918
bias$(y, \theta) = (0.1, 0.1)$	1.247	0.678	6.375	0.925
bias$(x, y, \theta) = (0.1, 0.1, 0.1)$	1.614	0.681	6.599	0.925

Table 5-2: Average ATE (in meters) over the period of graph construction for all *bias* parameter implementations (averaged over 20 simulations).

12 % on the Intel path. The PGO performance more than triples in ATE for the worst-case on the Manhattan path and shows an even more substantial increase under the influence of any orientation bias on the Intel path.

Consider the worst-case simulation on the Manhattan path with a constant input parameter 'bias(x, y, θ)' $= (0.1\,\text{m}, 0.1\,\text{m}, 0.1\,\text{rad})$ and a parameter-node $p^{\text{bias}(x,y,\theta)} \in \mathcal{SE}(2)$ added with the static connectivity strategy.

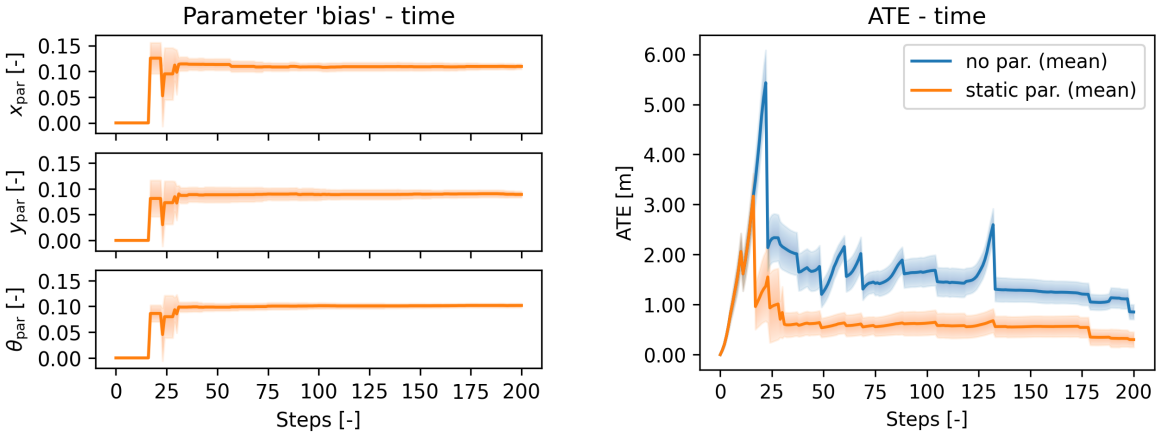


Figure 5-3: Manhattan path with constant input parameter 'bias(x, y, θ)' $= (0.1\,\text{m},\ 0.1\,\text{m},\ 0.1\,\text{rad})$ and a static output parameter on the odometry sensor: **PGO graph without parameters (blue)** versus **PPGO graph with static parameter (orange)**.

The result is shown in Figure 5-3, which indicates that:

- *Tracking accuracy*: the vector representation of parameter-node $p^{\text{bias}(x,y,\theta)} \in \mathcal{SE}(2)$ quickly converges to the true value $(0.1, 0.1, 0.1) \in \mathbb{R}^3$. What is notable is that the estimate stays at its initialisation value of $\mathbf{0} \in \mathbb{R}^3$ until a loop-closure constraints is established. This constraint conflicts with the string of prior odometry constraints, and the introduced congruence allows for estimation of the bias parameter. Similar to the case of Figure 5-2, the more constraints are added, the more the parameter estimate steadies.
- *Pose accuracy*: the ATE over time shows clear improvements with respect to the PGO solution. Before the establishment of the loop closure constraint, the value for ATE is

identical between both graphs (because the parameter value is still at its initialised value of zero). Once the parameter can be estimated, the PPGO solution shows approximately a 66 % decrease in ATE over the period of graph construction, with an average error more or less in line with the baseline.

A more striking relative performance gain is found in Figure 5-4, where the same simulation with 'bias(x, y, θ)' $= (0.1\,\text{m}, 0.1\,\text{m}, 0.1\,\text{rad})$ is run over the Intel path. Whereas the PPGO solution is again consistent with the baseline, the PGO solution shows a substantial increase in ATE. The increased ATE corresponds with the solution path, which does not agree with the input trajectory shown in Figure 5-1b. The incorrect solution can be attributed to the accumulated orientation bias, which induces a path that rotates in on itself. Performing PGO from this node configuration does not result in the correct solution, resulting in consistently large ATE.

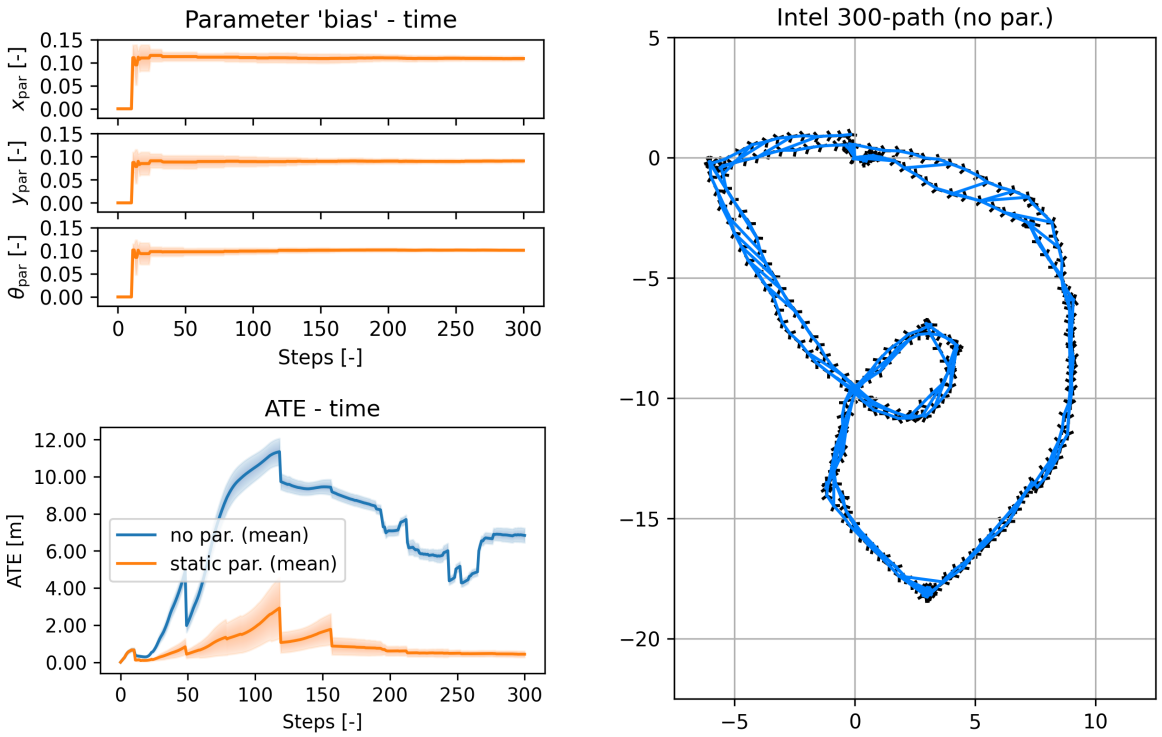


Figure 5-4: Intel path with constant input parameter 'bias(x, y, θ)' $= (0.1\,\text{m},\ 0.1\,\text{m},\ 0.1\,\text{rad})$ and a static output parameter on the odometry sensor: **PGO graph without parameters (blue)** versus **PPGO graph with static parameter (orange)**.

5-2-2 Sensor frame

As discussed in Section 3-3-3, the sensor frame is a special case of the bias parameter that is only defined for transformations that relate two poses. Consider a simulation on the Manhattan path with a constant input parameter 'frame(x, y, θ)' $= (0.1\,\text{m}, 0.1\,\text{m}, 0.1\,\text{rad})$ and a similar configuration of parameter-node $p^{\text{frame}(x,y,\theta)} \in \mathcal{SE}(2)$ added with the static connectivity strategy.

The result shown in Figure 5-5, which indicates improved performance for the unparameterised PGO graph with respect to the consistent PPGO solution. This can be attributed to the fact that for small pose rotations, the y-translation component is mostly affected by the modelling deficiency, as given by (3-4) in Section 3-3-3. Because the majority of the pose transformations comprise a small pose rotation, the unmodelled sensor frame transformation parameter has a similar effect as an unmodelled y-bias. For this case, a low-cost solution exists that compensates for a positive y-bias with a combination of increased orientation and y-translation, which explains the improved PGO performance.

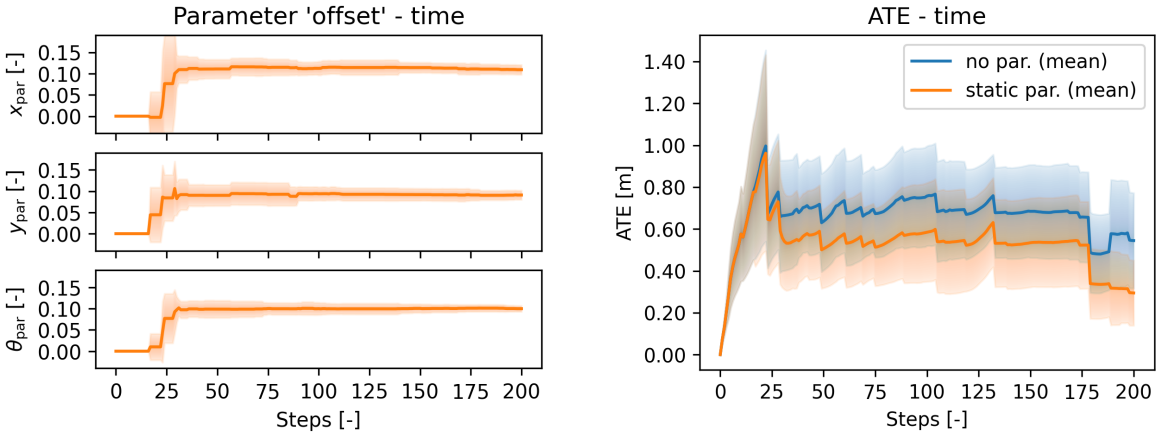


Figure 5-5: Manhattan path with constant input parameter '$\text{frame}(x, y, \theta)$' $= (0.1\,\text{m},\ 0.1\,\text{m},\ 0.1\,\text{rad})$ and a static output parameter on the odometry sensor: **PGO graph without parameters (blue)** versus **PPGO graph with static parameter (orange)**.

5-2-3 Scaling factor

Consider the case of a scaling factor parameter-node $p^{\text{scale}(y)} \in \mathbb{R}$ connected to all odometry constraints in the estimate graph of the PPGO sub-simulation. The value of $p^{\text{scale}(y)} \in \mathbb{R}$ is varied as to minimise the graph cost function. The trajectories considered in this research are derived from the case of a wheeled robot, which is considered a *non-holonomic system* (i.e., a system whose state depends on the path taken) that is barred from pure y-translation without a change in x-translation and orientation. Accordingly, for the considered trajectories, the lateral translation component is consistently found to be close to zero, whereas the orientation component is only non-zero when making a turn, as shown in Figure 5-6.

This is a characteristic of the trajectories derived from wheeled robots, which influences the ability to reconstruct scaling parameters. With the ground truth lateral translation component consistently small, the corresponding measurement component is subject to domination by sensor noise. As a result the parameter-node $p^{\text{scale}(y)} \in \mathbb{R}$ in the parameterised sub-simulation is optimised to find the amplification of y-translation measurement components (which consists largely of sensor noise) that minimises the cost function value. Without a perceivable *true* y-translation component to scale, such a parameter implementation is considered illogical. Therefore, due to the nature of the trajectories considered in this research, a scaling parameter on the y-translation is not considered. The occasional non-zero orientation component, however, does allow for estimation with the static connectivity strategy.

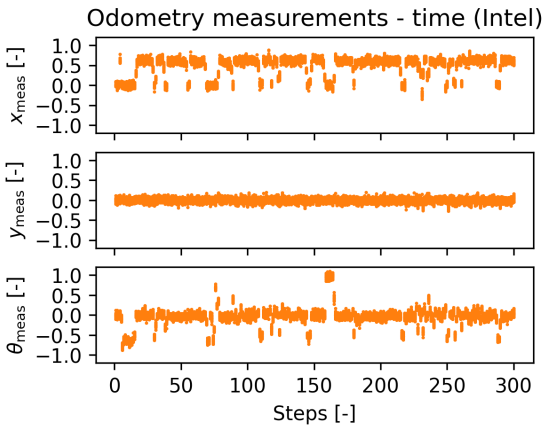


Figure 5-6: Measurements of the Intel path with constant input parameter 'scale(y)' $= 1.1$.

Table 5-3 summarises the average ATE over the period of graph construction for all parameter implementations. It again shows consistently better performance of the PPGO solutions with respect to that of PGO. Furthermore, the PPGO performance is always within 6 % of the baseline on the Manhattan path, and within 5 % on the Intel path.

Simulation	Manhattan (200, 271)		Intel (300, 403)	
	no par.	static par.	no par.	static par.
Baseline	0.540	0.559	0.826	0.838
scale$(x) = 0.1$	0.915	0.575	1.078	0.832
scale$(\theta) = 0.1$	0.591	0.566	0.826	0.872
scale$(x, \theta) = (0.1, 0.1)$	0.773	0.590	1.057	0.875

Table 5-3: Average ATE (in meters) over the period of graph construction for all *scaling factor* parameter configurations (averaged over 20 simulations).

Consider the worst-case simulation on the Manhattan path with a constant input parameter 'scale(x, θ)' $= (1.1, 1.1)$ and a parameter-node $p^{\text{scale}(x,\theta)} \in \mathbb{R}^2$ added with the static connectivity strategy. The result shown in Figure 5-7 which shows similar performance to the case of Figure 5-5. indicates improved PGO performance with respect to the consistent PPGO performance. The improved PGO performance can be attributed to the fact that the inconsistency due to an unmodelled scaling factor parameter scales linearly with the magnitude of the measurement value.

5-3 Time-dependent parameters

For the estimation of time-dependent parameters, two connectivity strategies are proposed in Section 3-4:

- The *sliding window strategy* (introduced in Section 3-4-2), which estimates the dynamic behaviour of a parameter using a single parameter-node connected to a set of constraints of which the content moves with the most recent pose. The measurement value is therefore estimated using a set of only the most recent constraints.

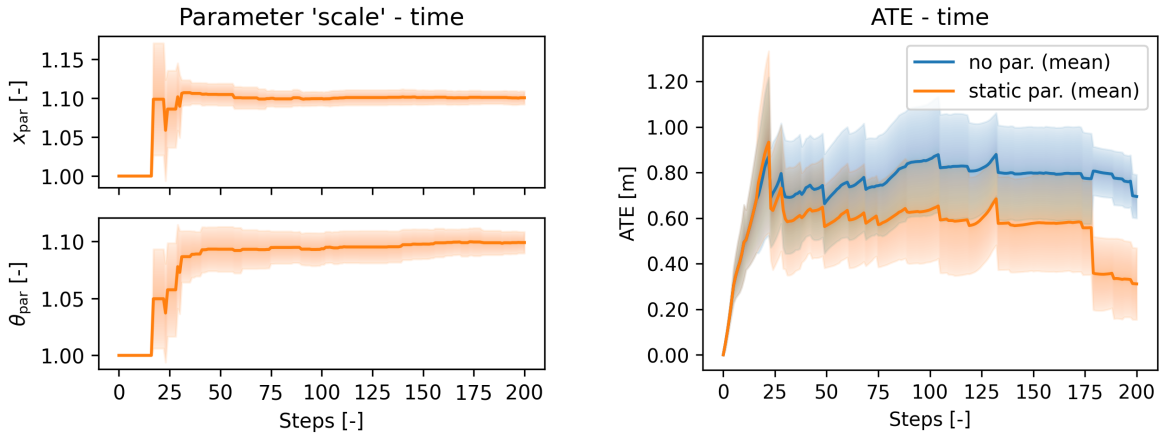


Figure 5-7: Manhattan path with constant input parameter 'scale'$(x, \theta) = (1.1, 1.1)$ and a static output parameter on the odometry sensor.: **PGO graph without parameters (blue)** versus **PPGO graph with static parameter (orange)**.

- The *timely batch strategy* (introduced in Section 3-4-3), which estimates the dynamic behaviour of a parameter using a set of parameter-nodes, each of which is connected to a batch of similarly aged constraints. Each parameter-node estimate represents the average parameter estimate over a period of time, as covered by the corresponding constraints.

Following the discussion of Section 5-2, only the highest-dimension (or worst-case) parameter implementations are considered. To test the tracking accuracy of both approaches, the parameter is varied according to the sinusoidal function

$$f_{\sin}(\alpha, t) = 0.1 \sin\left(\frac{\pi}{100}\alpha t\right),$$

where $\alpha \in \mathbb{N}$ is used to scale the function frequency. A sinusoidal function is chosen because:

- It is symmetric about the origin. As a result, both positive and negative parameter values are estimated.
- It is continuously changing.
- Its frequency can easily be manipulated in order to test the reactivity of the parameter estimate.

5-3-1 Sliding window strategy

The *sliding window connectivity strategy* is introduced in Section 3-4-2. It uses a dynamically-sized set of constraints that extends such that its constraints encompass at least two pose-nodes at which a loop closure has been established. This approach guarantees that each constraint is updated with a parameter value that is estimated from a set of constraints *at least* as large as stipulated by the *sliding window size*.

Consider the simulation on the Manhattan path with a time-dependent input parameter 'bias'$(x, y, \theta) = (f_{\sin}(1, t), f_{\sin}(2, t), f_{\sin}(3, t))$ and a parameter-node $p^{\text{bias}(x,y,\theta)} \in \mathcal{SE}(2)$ added with the sliding window connectivity strategy with window size 5. This window size was

found optimal by running the same simulation for a range of window sizes and choosing the value with the lowest average squared parameter tracking error. The result of the simulation is shown in Figure 5-8.

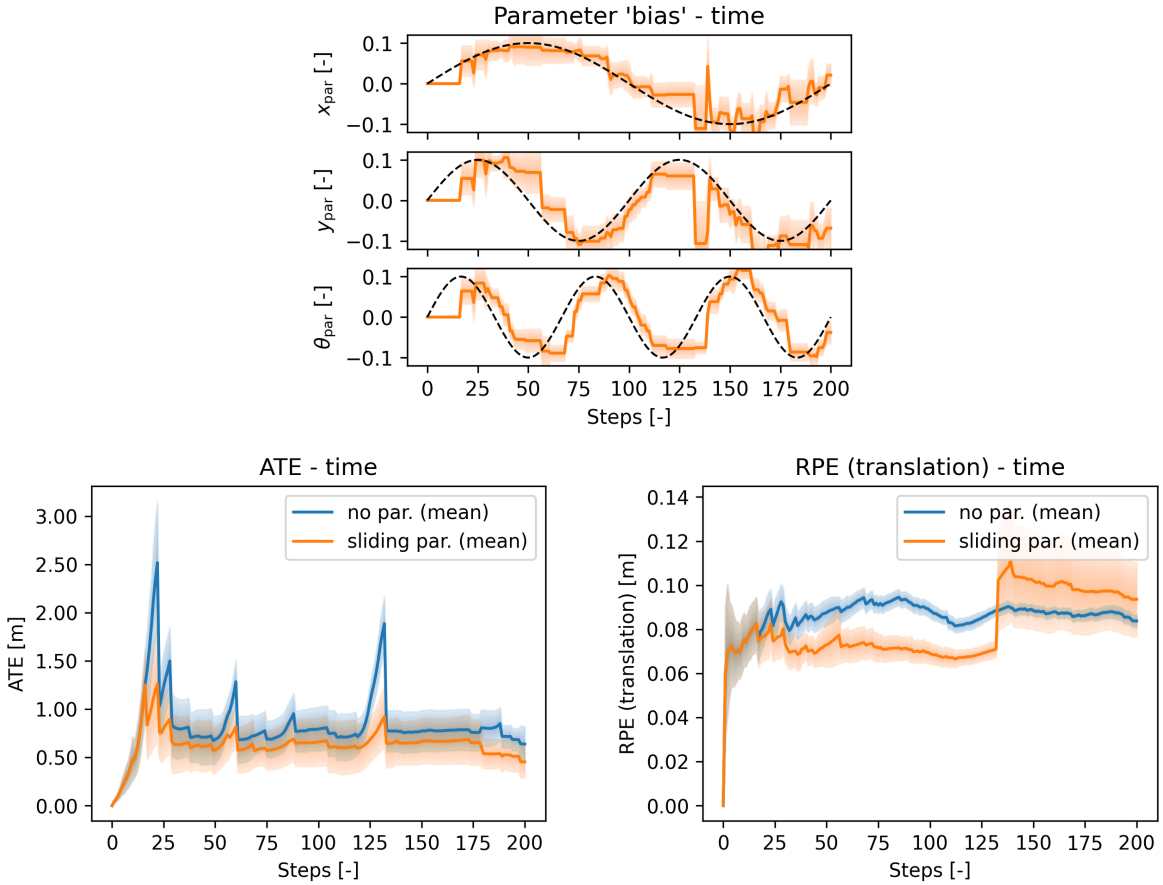


Figure 5-8: Manhattan path with sinusoidal input parameter 'bias(x, y, θ)' $= \big(f_{\sin}(1,t),\ f_{\sin}(2,t),\ f_{\sin}(3,t)\big)$ and *sliding window* (with window size 5) output parameter on the odometry sensor: **PGO graph without parameters (blue)** versus **PPGO graph with sliding window parameter (orange)**.

The following conclusions can be drawn:

- *Tracking accuracy*: the single parameter-node with the sliding window connectivity strategy is able to reconstruct the sinusoidal parameter relatively well. Due to the nature of the strategy, where the parameter value at any time instance is estimated from a set of constraints that all have been established *before* that instance, a delay can be observed between the true parameter value and its estimate.
- *Pose accuracy*: the difference in ATE between the PGO and PPGO solution is relatively small compared to the results achieved in Section 5-2. This can be attributed to the sinusoidal nature of the parameter, of which the mean value is zero. That is, the systematic measurement deviation due to the modelling inconsistency are both negative and positive over time. The Relative Position Error (RPE) shows a large increase around step 140, which corresponds to the inaccurate instantaneous parameter estimate.

The Manhattan grid is considered a more ideal path, due to the abundance of loop closure constraints. In Section 3-4-2 it was discussed that a key characteristic of the sliding window strategy was its ability to dynamically adjust its resize to encompass at least two pose-nodes at which a loop closure has been established. The same simulation is run on the Intel path, which has considerably fewer loop closure constraints in its first traversal of the map. The result is shown in Figure 5-9.

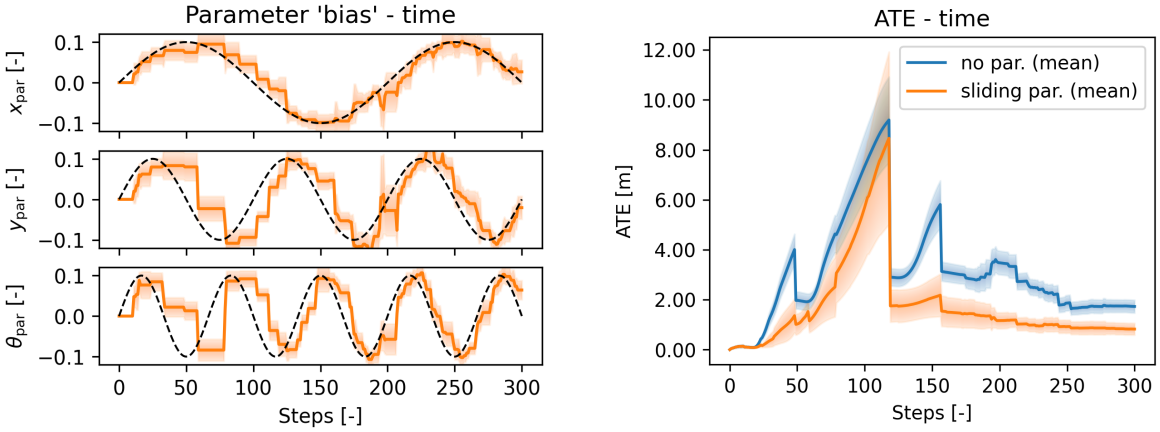


Figure 5-9: Intel path with sinusoidal input parameter '$\text{bias}(x, y, \theta)$' $= \big(f_{\sin}(1,t),\ f_{\sin}(2,t),\ f_{\sin}(3,t)\big)$ and *sliding window* (with window size 5) output parameter on the odometry sensor: **PGO graph without parameters (blue)** versus **PPGO graph with timely batch parameter (orange)**.

The following conclusions can be drawn:

- *Tracking accuracy*: the parameter estimate over time clearly exhibits a step-like behaviour due to the absence of loop closure constraints. After each optimisation, however, the parameter estimate is close to the true value, which is also the value that is embedded in the intermediary odometry constraints.
- *Pose accuracy*: the ATE shows improved pose accuracy for the PPGO solution compared to the PGO solution. The relative performance difference is larger than observed in Figure 5-8, which is due to the worse performance of the PGO solution.

5-3-2 Timely batch strategy

The *timely batch connectivity strategy* is introduced in Section 3-4-3 for the purpose of estimating time-dependent parameters. It is a batch strategy that organises the constraints by timely correlation, where each batch of similarly aged constraints is represented by a parameter-node that corresponds to a time period. In contrast to the sliding window strategy, the parameter-nodes are permanently connected to its set of constraints, and any information added in a later phase will benefit its estimate. Therefore, the parameter estimate is constantly improving as more constraints are added to the graph, similar to the pose estimates.

Consider the simulation on the Manhattan path with a time-dependent input parameter '$\text{bias}(x, y, \theta)$' $= (f_{\sin}(1,t), f_{\sin}(2,t), f_{\sin}(3,t))$ and a set of parameter-nodes $\{p^{\text{bias}(x,y,\theta)} \in \mathcal{SE}(2)\}$

added with the timely batch connectivity strategy with batch size 10. The result is shown in Figure 5-10.

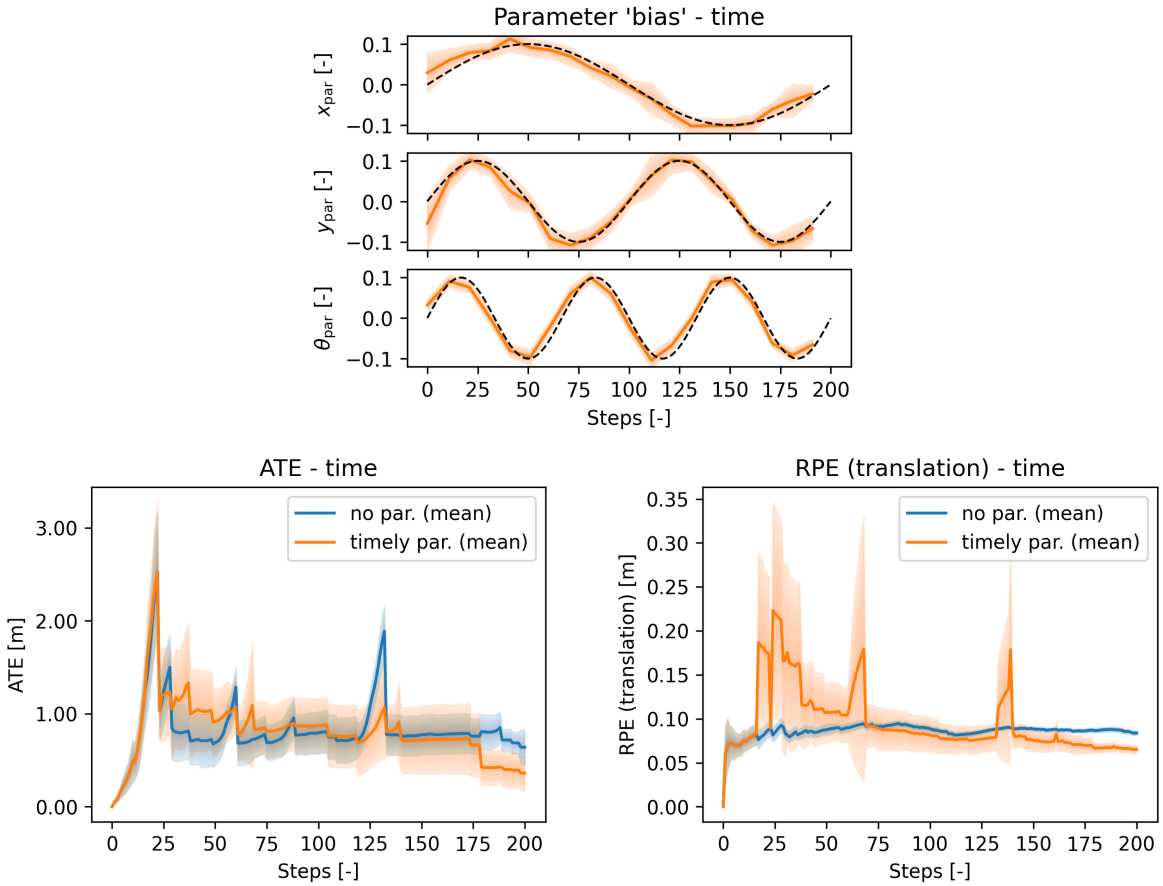


Figure 5-10: Manhattan path with sinusoidal input parameter '$\text{bias}(x, y, \theta)$' $= \big(f_{\sin}(1,t),\ f_{\sin}(2,t),\ f_{\sin}(3,t)\big)$ and *timely batch* (with batch size 10) output parameter on the odometry sensor: **PGO graph without parameters (blue)** versus **PPGO graph with static parameter (orange)**.

The following conclusions can be drawn:

- *Tracking accuracy*: the estimated values of the parameter-nodes are data points from which the parameter estimate is reconstructed through linear interpolation. As indicated by the plot, the interpolated estimate is an accurate reconstruction of the true parameter value. In contrast to the sliding window solution of Figure 5-8, the timely batch solution shows no indication of a delay.
- *Pose accuracy*: remarkably, the pose accuracy of the PPGO solution is less than that of the PGO solution, even though the tracking accuracy seems high. This is because the parameter reconstruction is taken from the *final* estimate of all parameter-nodes. The estimate for the parameter-nodes is continually improved as more constraints were added to the graph. As such, the parameter interpolation is not a representation of the parameter estimate *during* graph construction, but rather *after* graph construction.

  The increased ATE can be attributed to an inaccurate estimate of a parameter-node at

the corresponding graph instance. The inaccuracy in the parameter-node estimates is reflected in the RPE, which show periods of large error.

The same simulation run over the Intel dataset is more problematic and yields no solution until the graph contains one full map traversal. The lack of incongruence within the graph due to an absence of loop closure constraints results in problems for some parameter-nodes. This can be attributed to insufficient graph complexity and an excess of degrees of freedom; i.e., an under-constrained optimisation problem.

A solution to the fact that accuracy (and in some cases convergence) can not be guaranteed for all applications of the timely batch strategy is to consider it a *post-processing step*. The same parameter tracking result can be obtained by first considering the PGO solution; until the graph is fully constructed, at which point all parameter-nodes are added in a single instance according to the timely batch strategy. The result is a satisfactory estimate of the parameter value over time, but only *after* the graph has been constructed.

Consider the same simulation on the Intel path with the timely batch strategy applied as a post-processing step. This time, the graph converges, with the result is shown in Figure 5-11.

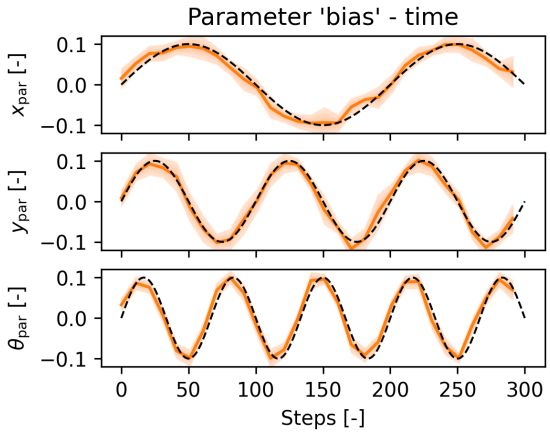


Figure 5-11: Parameter tracking on the Intel path with sinusoidal input parameter '$\text{bias}(x, y, \theta)$' $= \big(f_{\sin}(1,t),\ f_{\sin}(2,t),\ f_{\sin}(3,t)\big)$ and *timely batch* (with batch size 10) output parameter on the odometry sensor.

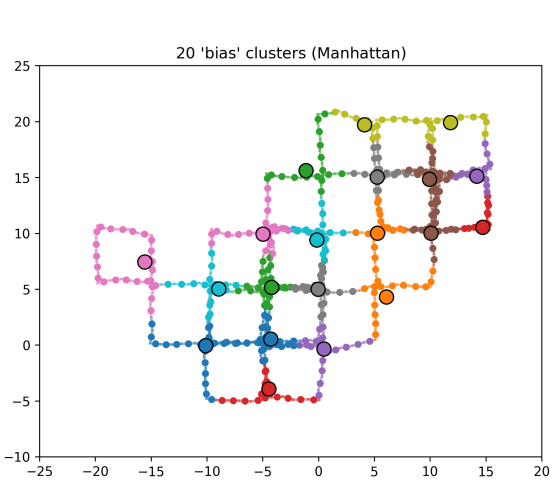
5-4 Space-dependent parameters

The *spatial batch strategy* is proposed in Section 3-4-4 for the purpose of estimating space-dependent parameters. It is a batch strategy that organises the constraints by spatial correlation, where each batch of nearby constraints is represented by a parameter-node that corresponds to a region of space. The constraints are allocated to the batches based on the centroid of the translation components of the connected pose-nodes. Note that the spatial batch strategy is considered a post-processing step, with the parameter-nodes added to the graph after all measurements have been collected.

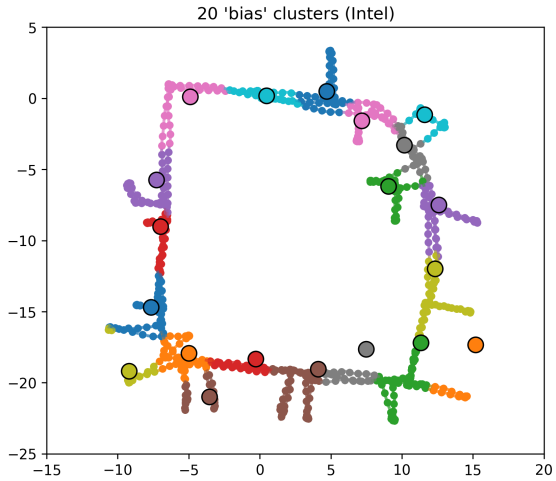
The k-means clustering algorithm is used to find the optimal constraint-to-batch allocation that minimises in-cluster sum-of-squares. The algorithm is applied in an iterative fashion

because of the circular dependency relationship between the node configuration and the constraint-to-batch allocation. That is, the pose-parameter graph is optimised and its constraint reallocated until the batch composition has converged (see Algorithm 1).

Figure 5-12 shows the converged constraint-to-batch allocation for 20 parameter-nodes on the Manhattan grid $(400, 541)$ and the Intel path $(800, 1106)$. This allocation is independent of any parameters that might apply. The plots are colour-coordinated to highlight the correspondence of constraints to parameter-nodes.



(a) Manhattan grid $(400, 541)$ with 399 odometry constraints.



(b) Intel path $(800, 1106)$ with 799 odometry constraints

Figure 5-12: Converged constraint-to-batch allocation for 20 parameter-nodes on **(a)** Manhattan grid and **(b)** Intel path.

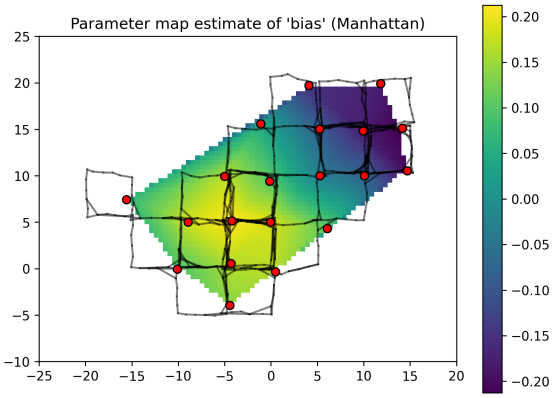
The spatial batch connectivity strategy only provides a data point in space for each parameter-node. Using the constraint-to-batch allocation illustrated in Figure 5-12, only 20 of such data points are available. Therefore, the estimation of the spatial relationship of any parameters is restricted to slow moving functions.

To test the tracking accuracy of the spatial batch strategy, a 'bias(x)' parameter is applied to the odometry constraints and varied according to the sinusoidal function

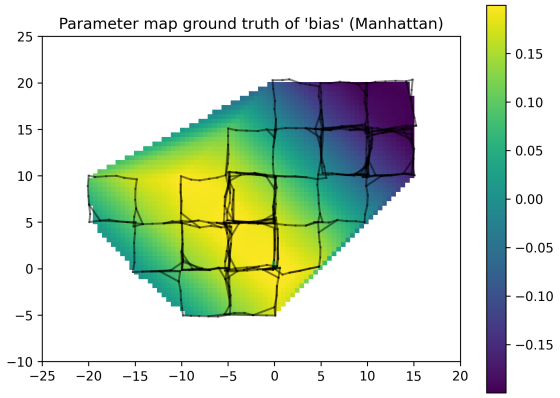
$$f_{\text{space}}(x, y) = 0.2 \cos\left(\frac{x + y}{10}\right).$$

The results of estimating this parameter are shown in Figure 5-13 and Figure 5-14 for the Manhattan grid and Intel path, respectively. The plots compare the parameter estimate interpolated over the data points represented by the parameter-nodes in the estimate graph (on the left) with interpolation of the ground truth parameter values over the data points represented by each odometry constraint (on the right). The interpolation represents a *parameter map*, with the parameter value in 2-dimensional space indicated by colour in a heat map. The ground truth parameter map represents the best possible interpretation of the parameter value as a function of the space covered by the trajectory. For the ground truth, a data point is available for each constraint, which constitutes a larger domain to interpolate over.

The parameter estimate interpolation clearly resembles the ground truth interpolation, even though it is derived from a significantly reduced set of data points.

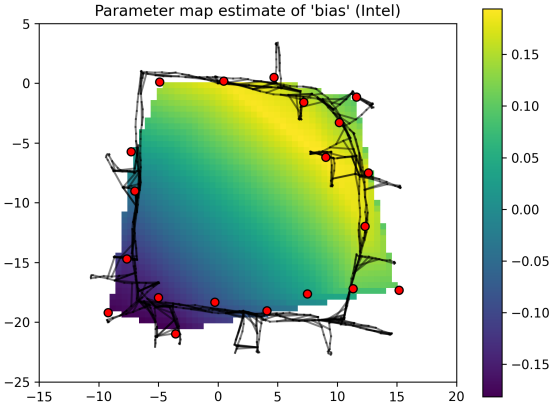


(a) Estimate parameter map interpolated over 20 parameter-nodes.

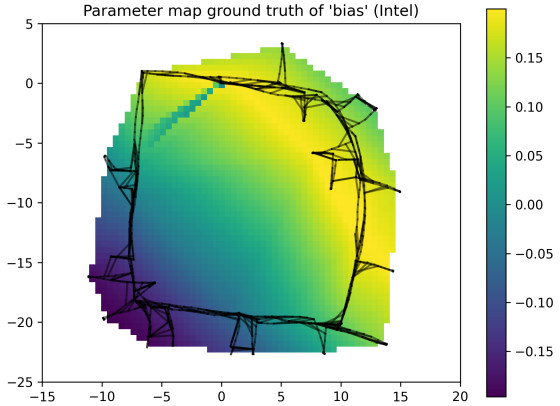


(b) Ground truth parameter map interpolated over all 400 odometry constraints.

Figure 5-13: Interpolated parameter map on the Manhattan path $(400, 541)$ of a space-dependent sinusoidal input parameter, modelled by a *spatial batch* (with 20 batches) output parameter on the odometry sensor.



(a) Estimate parameter map interpolated over 20 parameter-nodes.



(b) Ground truth parameter map interpolated over all 799 odometry constraints.

Figure 5-14: Interpolated parameter map on the Intel path $(800, 1106)$ of a space-dependent sinusoidal input parameter, modelled by a *spatial batch* (with 20 batches) output parameter on the odometry sensor.

5-5 Summary

The capabilities of PPGO are assessed by comparing an adequately configured PPGO graph with a PGO counterpart, which does not have any means of estimating the parameter value. The simulations are configured with a wheel encoder for odometry, a LiDAR scan-matching setup for loop closures and proximity constraints, and a GPS sensor for location priors. Each

simulation is tested on the Manhattan grid and Intel path for 20 runs to account for different outcomes due to the influences of sensor noise.

The estimation of the bias and sensor frame parameters is found to be reliable over all pose transformation measurement components with consistent pose accuracy over all graphs. For the scaling factor, only the rotation and longitudinal translation components are considered, with the lateral translation component disregarded. This is because, for the data sets used in this research, the ground truth lateral translation component is found close to zero, which results in the measurement component likely to be dominated by sensor noise. When considering only the two above-mentioned measurement components, the scaling factor parameter estimation is found to be reliable with consistent pose accuracy.

The *sliding window* and *timely batch* connectivity strategies are found to be able to reconstruct a time-dependent sinusoidal parameter:

- The sliding window strategy offers an instantaneous parameter estimate, albeit with a slight delay. The delay is inherent to the sliding approach, as its parameter estimate is the value that minimises the cost over a set of past constraints. Since all estimations are embedded in the overdue constraints, any estimation inaccuracy has a permanent effect on the solution.
- The timely batch strategy is found to be better applied as a post-processing step because its accuracy and convergence cannot be guaranteed when applied during graph construction. This can be attributed to a resulting under-constrained optimisation problem.

The *spatial batch* connectivity strategy is also applied as a post-processing step. It is found to be able to reconstruct a space-dependent sinusoidal parameter by interpolating the parameter-node estimates over space. Although the parameter-nodes offer only a limited set of data points over a limited domain, the estimate graph is able to successfully approximate the ground truth parameter map.

Chapter 6

Conclusion

In this chapter, a conclusion is formed on the research performed during this thesis. More specifically, this chapter covers the following:

- Section 6-1 *'Discussion'* reflects back on the research questions posed in Chapter 1 and provides an answer based on the results presented in Chapter 5.
- Section 6-2 *'Recommendations'* provides recommendations for future work.

6-1 Discussion

The main goal of this thesis was to serve as a starting guide for implementing Pose-Parameter Graph Optimisation (PPGO) for a variety of commonly occurring use-cases. For this purpose, a set of generally applicable basis parameters was defined alongside corresponding measurement models. With the addition of a unique parameter-node for every instance yielding an under-constrained optimisation problem, alternative connectivity strategies are proposed that take into account the fluctuation of the underlying parameter: the *static connectivity strategy* for constant parameters; the *sliding window* and *timely batch connectivity strategies* for time-dependent parameters; the *spatial batch connectivity strategy* for space-dependent parameters. To investigate the broad range of applications that this generalised approach to PPGO hopes to cover, the following research questions were posed in Chapter 1:

- *Which types of parameters can be estimated with the use of Pose-Parameter Graph Optimisation?*

  Section 3-1 provides a motivation for the definition of the two generally applicable basis parameters, which comprise the additive *bias* parameter (with the *sensor frame* parameter as a special case) and the multiplicative *scaling factor* parameter. These parameters form the basis of an affine transformation of the unmodified measurement models. Any parameter influence can at least be approximated by an affine transformation, which makes the bias and scaling factor good candidates to be generally applicable.

  In Section 5-2, the static connectivity strategy is investigated for all parameter implement-

ations applied to the set of odometry constraints. The estimation of the bias and sensor frame parameters is found to be reliable over all pose transformation measurement components. For the scaling factor, only the rotation and longitudinal translation components are considered, over which estimation was found to be reliable. For the lateral translation component it was argued that, considering the nature of the trajectories with consistent near-zero lateral translation measurements, such an application is illogical and therefore disregarded in this research.

- *How does the performance of Pose-Parameter Graph Optimisation compare with Pose Graph Optimisation under appropriate conditions?*

  For PPGO, two performance criteria are considered: *pose accuracy* and *tracking accuracy* (or *parameter accuracy*), with only pose accuracy applicable to Pose Graph Optimisation (PGO). For the case of a constant parameter influence, PPGO is quick to converge to an accurate parameter estimate, which results in an improved Absolute Trajectory Error (ATE) with respect to the PGO counterpart. This performance is consistent across all parameter implementations on all trajectories, with only a slight performance penalty observed for increased parameter dimensionality. Even in the presence of any redundant parameter-node components, PPGO performance is on par with PGO, with the redundant components quickly estimated to be zero.

  In the presence of varying parameters (see Section 5-3 and Section 5-4), it was found that PPGO can be implemented to reconstruct both time-varying as space-varying parameters. The sliding window strategy is able to provide an accurate instantaneous parameter estimate, which results in a consistently lower ATE over the period of graph construction. The batch strategies (both timely and spatial) are found to be best employed as a post-processing step, where the accuracy of the parameter reconstruction also provides a decrease in ATE.

- *Under what conditions is Pose-Parameter Graph Optimisation able to capture the dynamics of a parameter?*

  A pose-parameter graph can only be optimised when its constraints are incongruent (i.e., conflicting; see Section 2-4). Furthermore, because the parameters are not directly measured, but rather deduced from measurement inconsistencies, no meaningful constraints can be imposed on the value of the set of parameter-nodes. As such, a parameter-node must be connected to multiple constraints, which results in a parameter estimate that *on average* compensates for the measurement inconsistency over all connected constraints. The parameter estimate accuracy inherently increases with the number of connected constraints. As a result, sufficient connectivity of the parameter-nodes is required for accurate estimation. Insufficient connectivity is the culprit that weakens the instantaneous performance of the timely batch strategy, as discussed in Section 3-4-3.

  - *What insights can be gained by modelling a parameter's temporal correlation using the sliding window and timely batch connectivity strategies?*

    In Section 5-3 it was found that the sliding window and timely batch connectivity strategies are both able to reconstruct a time-dependent sinusoidal parameter value.

    * The sliding window strategy is able to provide an instantaneous parameter estimate for every time instance, of which the fluctuations are subject to a slight delay. The delay is inherent to the sliding approach because its estimation is based on a set

of *past* constraints. The accuracy of the parameter estimate results in a decreased ATE for PPGO relative to PGO.

  - The timely batch strategy is found to be better applied as a post-processing step because its accuracy and convergence cannot be guaranteed when applied during graph construction for all implementations. The set of parameter-node estimates can be interpolated into an accurate reconstruction of the parameter value over time, without any delays in the fluctuations. This also results in a decrease in ATE with respect to the PGO solution.

- *What insights can be gained by modelling a parameter's spatial correlation using the spatial batch strategy?*

  The spatial batch connectivity strategy is also applied as a post-processing step. In Section 5-4 it was found that the spatial batch strategy is able to provide an accurate reconstruction of the underlying space-dependent parameter. Each batch represents a data point at the centroid of the translation components of all related poses. As such, the parameter-node estimates can be interpolated into a parameter map estimate. The reconstruction is limited by the availability of data points, which depends on the domain covered by the robot trajectory. Although the parameter-nodes offer only a limited set of data points over a limited domain, the estimate map is able to approximate the ground truth parameter map.

Overall, it can be concluded that PPGO can be applied to solve a variety of calibration scenarios for a variety of parameters, with both time-dependent and space-dependent parameters reconstructable with the proposed connectivity strategies. The correct estimation of a parameter consistently results in better pose accuracy, with only a slight performance penalty observed for higher-dimensional parameters.

The benefit of PPGO is that it can be applied on any existing robot that was already using PGO: all that is required is the definition of a parameter implementation and a connectivity strategy. The SLAM front-end [2] can stay the same and all hardware can stay the same.

6-2 Recommendations

Listed below are some ideas that could be addressed in future work:

- *Explore the use of constraints strictly within the set of parameter-nodes.*

  In Section 3-2-2 it was posed that the addition of a single parameter-node per time instance would result in an under-constrained optimisation problem because no meaningful constraints could be imposed on the parameter-nodes without the availability of direct measurements. However, one could argue that a random walk model could be imposed on the set of parameter-nodes, which would effectively be equivalent to an uncertain equality constraint. For this case, the random walk covariance would need to be tuned depending on the characteristics of the parameter fluctuations. However, it would interesting to see whether this strategy could be employed to allow for the addition of a unique parameter-node per time instance, which would therefore alleviate the need for the connectivity strategies posed in this thesis altogether.

  The random walk parameter model could be used to generalise the work proposed in this research; i.e., the static connectivity strategy is simply a random walk parameter model

with zero covariance. The batch approaches can be described by a random walk parameter model that defines zero covariance to nodes within a batch and infinite covariance to nodes beyond the batch. It would be interesting to find out what challenges are posed by the tuning of the random walk covariance.

- *Investigate the sensitivity of the noise model assumptions for PPGO.*

  The Gaussian noise assumption is the key factor that enables the formulation of the PGO and PPGO problems as least-square minimisation problems. In real-world scenarios, the encountered measurement and/or process noise are unlikely to be perfectly Gaussian. Therefore, it would interesting to see how PPGO is able to cope with measurements influenced by other noise characteristics. Furthermore, the implications of coloured noise could also be investigated.

- *Validate results with real-world experiments.*

  With this research being based solely on simulation data, it would be interesting to see how these results translate to real-world performance. Such experiments would inherently expose the framework to different graph characteristics and different noise characteristics. The simulations performed in this research should be considered an ideal case under the influence of strictly Gaussian noise and perfect measurement models.

- *Investigate the modelling of parameters on other sensors.*

  This research focuses solely on parameters that are applied only to the odometry sensor system, which would unlikely be the only sensor system affected by unknown parameters in a real-world scenario. Therefore, it could be interesting to investigate whether e.g. a GPS bias parameter map could be estimated.

Appendix A

Software

The software used in this research is written in `python` and utilises the General (Hyper) Graph Optimisation `C++` framework (or `g2o` [11]) for optimisation.

The source code is available on https://github.com/artjvl/self-calibrating-slam.

A-1 GUI

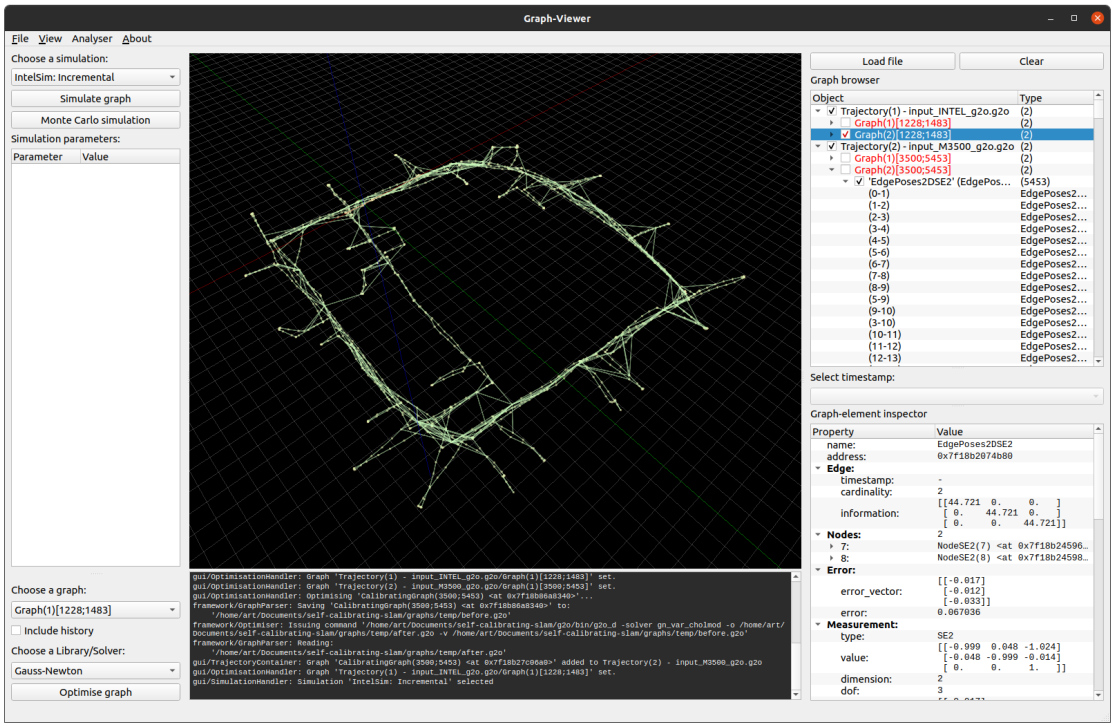


Figure A-1: The Graphical User Interface (GUI) of the Self-Calibrating SLAM visualiser.

A-2 Graph

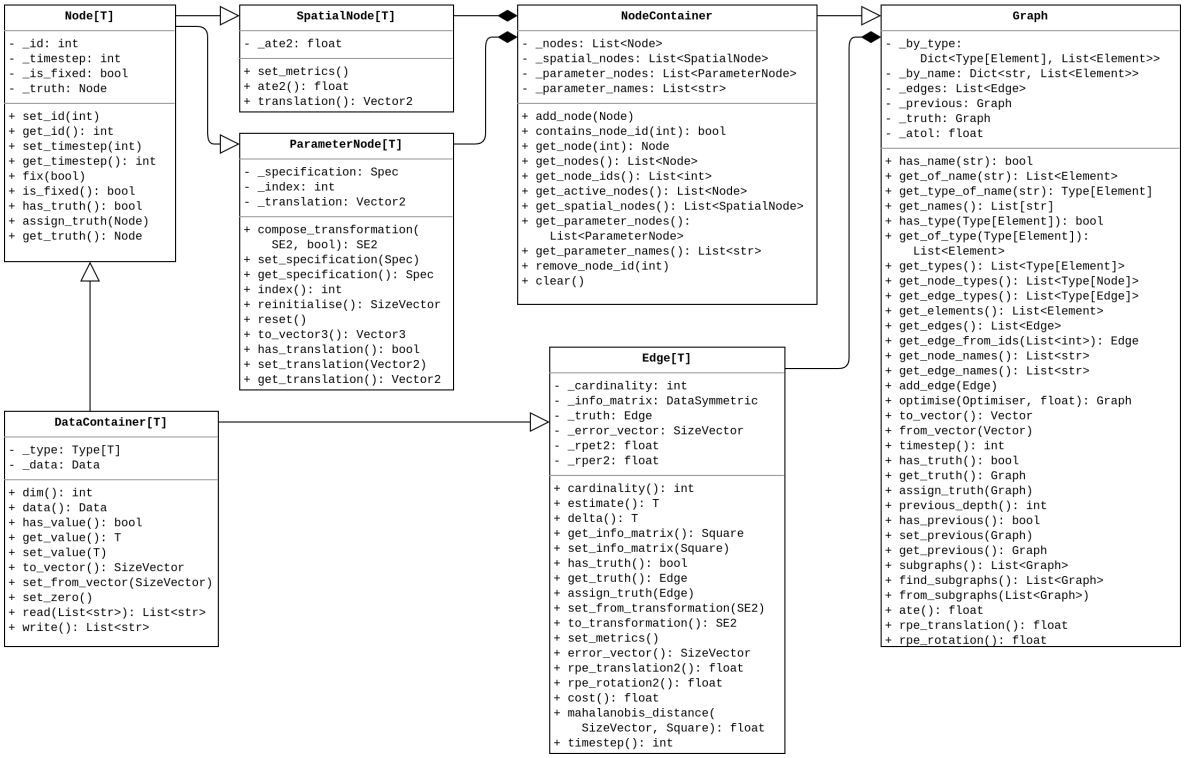


Figure A-2: UML - Graph

- `Graph` represents a pose-parameter graph. It sorts its elements (`Node`s and `Edge`s) by name and by type. It stores a ground truth `Graph`reference (`_truth`), a previous instance graph reference (`_previous`). It provides access to the performance metrics by e.g. accumulating the Absolute Trajectory Error (ATE) terms of all corresponding `SpatialNode`s or the Relative Position Error (RPE) terms of all corresponding `Edge`s.

  A `Graph` can optimise itself by saving to a `.g2o`-file and calling the `g2o` framework.

- `Node` represents a factor graph node. Each `Node` contains a unique identifier (`_id`) with which it is identified in any `NodeContainer` sub-class (i.e., `Edge` and `Graph`). Furthermore, it stores a time instance (`_timestep`) of creation, a reference to a ground truth `Node` (`_truth`), and a boolean to indicate whether its value is fixed (`_is_fixed`).
  - `SpatialNode` represents a node with a 'spatial' interpretation; i.e., a pose-node (modelled by `NodeSE2`) or a landmark-node (modelled by `NodeV2`). Its value is drawn in space when visualising the graph. `SpatialNode`s can be implemented to contribute to the ATE performance metric by assigning the squared absolute error (`_ate2`), if a truth reference is assigned.
    * `NodeSE2` stores a value of type `SE2`, which typically represents a robot pose.
    * `NodeV2` stores a value of type `Vector2`, which typically represents a landmark.
  - `ParameterNode` represents a parameter-node. It is defined by its value, its specification (or interpretation, `_specification`) and its index (`_index`). The specification defines

how the stored value should be interpreted by the constraint error function (which is defined in the corresponding `Edge`) and which measurement model function should be applied. For this research, the following specifications are defined: `BIAS`, `SCALE`, and (sensor) `FRAME`. The index defines how the stored value is converted to a `Vector3` when used in the error function.

The `ParameterNode` sub-classes implement the `compose_transformation` method, which defines how the stored value is used to manipulate a transformation measurement value in `SE2`.

* `ParameterNodeSE2` stores a value of type `SE2` and can represents either a `BIAS` or (sensor) `FRAME` in 3 dimensions. The index attribute is not used, as only a single `Vector3` representation is defined for the `SE2` data-type.
* `ParameterNodeV1` stores a value of type `Vector1` and can represent all scalar specifications. The index attribute defines which dimension is represented by the scalar parameter value. For a stored value of `1.0`, a default value of `0.0`, and an index of `0`, the `Vector3` representation is given by `[1.0 0.0 0.0]`.
* `ParameterNodeV2` stores a value of type `Vector2` and can represent all 2-dimensional specifications. The index attribute defines which dimension is *not* represented by the 2-dimensional parameter vector value. For a stored value of `[2.0 1.0]`, a default value of `0.0` and an index of `1`, the `Vector3` representation is given by `[2.0 0.0 1.0]`.
* `ParameterNodeV3` stores a value of type `Vector3` and can represent a 3-dimensional `SCALE`. The index attribute is not used.

- `Edge` represents a factor `Graph`(hyper) edge, which is used to model a pose-parameter graph constraint. Each `Edge` stores a cardinality value (`_cardinality`), an appropriately-sized information matrix (`_info_matrix`), a truth reference (`_truth`), and an error vector value (`_error_vector`). `Edge`s can be implemented to contribute to the RPE performance metrics by assigning the squared relative translation and rotation errors (`_rpet2` and `_rper2`, respectively).

  The `Edge` sub-classes override the `error_vector` method to define how its stored value and connected nodes are manipulated into an error vector value. This is where the `specification` and `Vector3` representation of the `ParameterNode` come in.

  – `EdgeSE2` represents an edge that stores a measurement in `SE2`.
    * `EdgePosesSE2` connects to two pose-nodes (`NodeSE2`) and represents a pose transformation constraint.
  – `EdgeV2` represents an edge that stores a `Vector2` measurement.
    * `EdgePoseV2` connects to a single pose-node (`NodeSE2`) and represents a location prior constraint.
    * `EdgePosePointV2` connects to a pose-node and a landmark-node and represents a landmark observation.

A-3 Simulation

- `BiSimulation` contains a *ground truth* sub-simulation (`_truth_sim`) and an *estimate* sub-simulation (`_estimate_sim`), which are both run simultaneously. It stores the ground

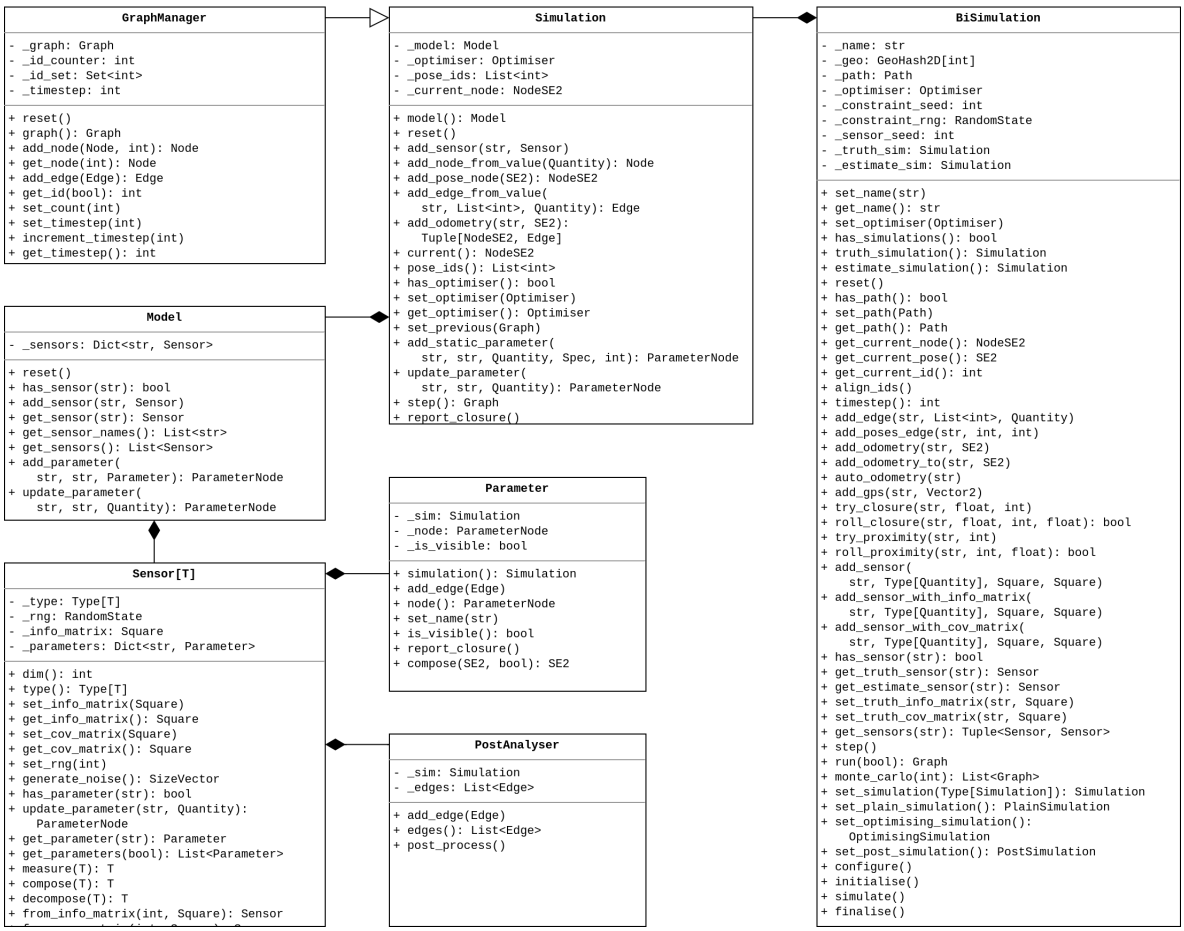


Figure A-3: UML - Simulation

truth poses in hash-map (`_geo`) based on a 2-dimensional grid. Furthermore, it stores an optimiser (`_optimiser`) and seeds for the constraints (`_constraint_seed`) and for the sensors (`_sensor_seed`).

Sub-classes of the `BiSimulation` are required to implement the methods:

- `configure()`, which configures most importantly the estimate simulation model using `set_plain_simulation()` or `set_optimising_simulation()` or `set_post_simulation()`.
- `initialise()`, which configures the sensors (with `add_sensor`) and the model parameters.
- `simulate()`, which defines the simulation loop. A simulation loop typically comprises the addition of an odometry constraint (with `add_odometry()`), a 'roll' of the `_constraint_rng` to establish a loop closure constraint (with `roll_closure()`) or proximity constraint (with `roll_proximity()`, any parameter updates, and a step forward in time (with `step()`).
- `finalise()`, which typically defines the post-processing steps.

- `Simulation` represents single-`Graph`sub-simulation instance. Sub-classes of the `Simulation` are required to implement the `step()` method.

All simulations have the ability to add parameter-nodes according to the static connectivity strategy (with `add_static_parameter()` and `update_parameter()`, which is typically only used in the truth simulation to define a truth parameter fluctuation).

  - `PlainSimulation` represents the simulation that does not optimise its `Graph`when a loop closure has been established. This simulation instance is the fastest to execute, and is typically used to as a 'path-finder', to verify the trajectory implementation.
  - `OptimisingSimulation` represents the simulation that optimises at every loop closure constraint. It has the ability to add parameter-nodes according to the sliding window strategy (with `add_sliding_parameter()`) and the timely batch strategy (with `add_timely_parameter()`).
  - `PostSimulation` represents the simulation that performs a post-processing step. It has the ability to add a spatial batch parameter (with `add_spatial_parameter()`, which is technically a `PostAnalyser`) and perform the corresponding post-processing step (with `post_process()`).

- `GraphManager` handles the `Graph` construction by making sure new `Node`s are assigned a unique identifier.
- `Model` stores all `Sensor`s.
- `Sensor` defines that is able to generate measurements according to its noise model (`_rng` and `_info_matrix`) and its assigned `Parameter`s.
  - `SensorSE2` measures transformations in `SE2`.
  - `SensorV2` measures translations in `Vector2`.
- `Parameter` represents a parameter model that creates a *set* of `ParameterNode`s, depending on its own implementation. It stores a specification and the corresponding simulation (`_sim`, which it needs to freely create `Node`s).
  - `StaticParameter` creates a single `ParameterNode` and connects it to all incoming constraints.
  - `SlidingParameter` creates a single `ParameterNode` and handles the dynamic resizing (from `_window_size`) until its set of constraint encompass at least two nodes at which a loop closure is established. Furthermore, it handles the embedding of the most recent parameter estimate into the outgoing constraints.
  - `TimelyBatchParameter` creates a new `ParameterNode` for each set of constraints, as stipulated by its batch size (`_batch_size`)
- `PostAnalyser` represents a post-processing action. It stores the `Edge`s that correspond to the `Sensor` it is applied to.
  - `SpatialBatchAnalyser` represents the post-processing step of adding `ParameterNode`s according to the spatial batch connectivity strategy. The `post_process()` method performs the k-means clustering step.

Appendix B

Lie groups for 2D and 3D transformations

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 $\mathbb{R}^3$ [23]. 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*.

Rotation matrices are used to describe a rotation of one reference frame relative to 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 [12], 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 of rigid motion and kinematics. Lie theory can be exploited to devise a coherent and robust framework for representing 3D transformation.

B-1 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.

B-1-1 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 [28]. 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.

B-1-2 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 [9]. 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.

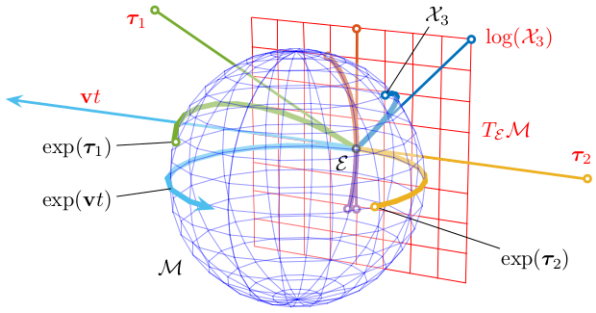


Figure B-1: Representation of the relation between the Lie group $\mathcal{M}$ (blue sphere) and Lie algebra $\mathfrak{m}$ (red plane), which is the tangent space at the identity $\mathcal{E} \in \mathcal{M}$ [28]. Through the exponential map, each straight path $\mathbf{v}t$ through the origin on the Lie algebra produces a path $\exp(\mathbf{v}t)$ around the manifold.

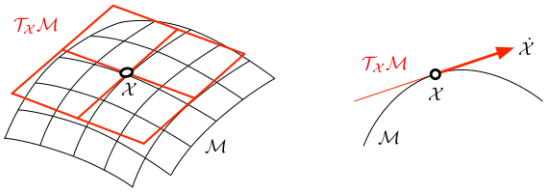


Figure B-2: A manifold $\mathcal{M}$ and the vector space $T_\mathcal{X}\mathcal{M} \cong \mathbb{R}^2$ tangent at the point $\mathcal{X} \in \mathcal{M}$, and a convenient side-cut [28]. The velocity element, $\dot{\mathcal{X}} = \partial\mathcal{X}/\partial t$, does not belong to the manifold $\mathcal{M}$, 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 B-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 [9]:

- 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

$$(\cdot)^\wedge \text{ (hat)}: \quad \mathbb{R}^{n_\mathcal{G}} \to \mathfrak{g}: \quad \boldsymbol{\tau} \mapsto \boldsymbol{\tau}^\wedge = \sum_{i=1}^{n_\mathcal{G}} \tau_i E_i,$$

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

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 B-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 $g^{-1} \circ g = g \circ g^{-1} = e$) [28]. 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

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

- 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,

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

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

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

- 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

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

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 $\boldsymbol{\tau}_e^\wedge = \mathrm{Ad}_x(\boldsymbol{\tau}_x^\wedge)$. 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 $\boldsymbol{\tau}_x \cong \boldsymbol{\tau}_x^\wedge$:

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

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.

B-1-3 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 manifold can be treated as a black box with only two possibilities for access: via the *plus* and *minus* operators, denoted by $\oplus$ and $\ominus$ [13, 28]. As such, the encapsulation limits operations on the manifolds that might void the constraints internal to its definition.

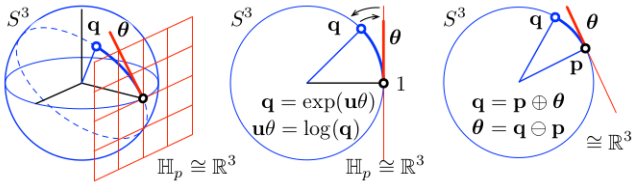


Figure B-3: The $\mathcal{S}^3$ manifold is a unit 3-sphere (blue) in the 4-space of quaternions [28]. The Lie algebra $\mathbb{H}_p$ is isomorphic to the hyper-plane $\mathbb{R}^3$ (red grid). The tangent vector $\boldsymbol{\theta} \in \mathbb{R}^3$ (red segment) wraps the manifold over the geodesic (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 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

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

  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 B-3.

- 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

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

  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 B-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

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

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

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

B-2 Poses

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.

B-2-1 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 (B-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:
   $$E_{\mathcal{SO}(3)}^{(1)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \quad E_{\mathcal{SO}(3)}^{(2)} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad E_{\mathcal{SO}(3)}^{(3)} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$
   The Lie algebra is a vector space whose elements can be decomposed into
   $$\boldsymbol{\theta}^\wedge = [\boldsymbol{\omega} t]_\times = \theta_1 E_{\mathcal{SO}(3)}^{(1)} + \theta_2 E_{\mathcal{SO}(3)}^{(2)} + \theta_3 E_{\mathcal{SO}(3)}^{(3)} \in \mathfrak{so}(3),$$
   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 (B-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* [9, 23, 28], which is defined as
   $$R(\boldsymbol{\theta}) = \exp([\mathbf{u}]_\times \theta) = I + [\mathbf{u}]_\times \sin(\theta) + [\mathbf{u}]_\times^2 (1 - \cos(\theta)) \in \mathcal{SO}(3). \tag{B-5}$$

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

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

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

B-2-2 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:

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

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 $\bar{T}_{\mathsf{B}}^{(\mathsf{A})} \in \mathbb{R}^{4\times 4}$ 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 cases:

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* [23]. The differential equation

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

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 (B-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

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

where $\mathbf{e}_i$ are the Euclidean base vectors and $E_{\mathcal{SO}(3)}^{(j)}$ 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

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

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 (B-2) as

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

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

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

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

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

     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

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

     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 (B-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 (B-7) and $\mathbf{s} = V^{-1}\mathbf{t}$.

Bibliography

[1] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE robotics & automation magazine*, 13(3):108–117, 2006.

[2] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[3] Luca Carlone. A convergence analysis for pose graph optimization via gauss-newton methods. In *2013 IEEE international conference on robotics and automation*, pages 965–972. IEEE, 2013.

[4] Luca Carlone and Andrea Censi. From angular manifolds to the integer lattice: Guaranteed orientation estimation with application to pose graph optimization. *IEEE Transactions on Robotics*, 30(2):475–492, 2014.

[5] Andrea Censi, Antonio Franchi, Luca Marchionni, and Giuseppe Oriolo. Simultaneous calibration of odometry and sensor parameters for mobile robots. *IEEE Transactions on Robotics*, 29(2):475–492, 2013.

[6] Andrea Censi, Luca Marchionni, and Giuseppe Oriolo. Simultaneous maximum-likelihood calibration of odometry and sensor parameters. In *2008 IEEE International Conference on Robotics and Automation*, pages 2098–2103. IEEE, 2008.

[7] Frank Dellaert and Michael Kaess. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017.

[8] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[9] Ethan Eade. Lie groups for 2d and 3d transformations. *URL http://ethaneade.com/lie.pdf, revised Dec*, 117:118, 2013.

[10] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[11] Giorgio Grisetti, Rainer Kümmerle, Hauke Strasdat, and Kurt Konolige. g2o: A general framework for (hyper) graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*, pages 9–13, 2011.

[12] Evan G Hemingway and Oliver M O'Reilly. Perspectives on euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments. *Multibody System Dynamics*, 44(1):31–56, 2018.

[13] Christoph Hertzberg. A framework for sparse, non-linear least squares problems on manifolds. In *Universität Bremen*. Citeseer, 2008.

[14] Andrew Howard and Nicholas Roy. The robotics data set repository (radish), 2003.

[15] Shoudong Huang, Yingwu Lai, Udo Frese, and Gamini Dissanayake. How far is slam from a linear least squares problem? In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3011–3016. IEEE, 2010.

[16] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[17] Rainer Kümmerle, Giorgio Grisetti, and Wolfram Burgard. Simultaneous calibration, localization, and mapping. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3716–3721. IEEE, 2011.

[18] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[19] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[20] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, 4(4):333–349, 1997.

[21] Will Maddern, Alastair Harrison, and Paul Newman. Lost in translation (and rotation): Rapid extrinsic calibration for 2d and 3d lidars. In *2012 IEEE International Conference on Robotics and Automation*, pages 3096–3102. IEEE, 2012.

[22] Jérôme Maye, Paul Furgale, and Roland Siegwart. Self-supervised calibration for robotic systems. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 473–480. IEEE, 2013.

[23] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[24] Edwin Olson and Michael Kaess. Evaluating the performance of map optimization algorithms. In *RSS Workshop on Good Experimental Methodology in Robotics*, volume 15, 2009.

[25] Edwin Olson, John Leonard, and Seth Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2262–2269. IEEE, 2006.

[26] Tim Pfeifer, Sven Lange, and Peter Protzel. Dynamic covariance estimation—a parameter free approach to robust sensor fusion. In *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 359–365. IEEE, 2017.

[27] David Prokhorov, Dmitry Zhukov, Olga Barinova, Konushin Anton, and Anna Vorontsova. Measuring robustness of visual slam. In *2019 16th International Conference on Machine Vision Applications (MVA)*, pages 1–6. IEEE, 2019.

[28] Joan Sola, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.

[29] Cyrill Stachniss, Giorgio Grisetti, Wolfram Burgard, and Nicholas Roy. Analyzing gaussian proposal distributions for mapping with rao-blackwellized particle filters. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3485–3490. IEEE, 2007.

[30] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE, 2012.

[31] Alex Teichman, Stephen Miller, and Sebastian Thrun. Unsupervised intrinsic calibration of depth sensors via slam. In *Robotics: Science and Systems*, volume 248, page 3. Citeseer, 2013.

[32] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

Glossary

List of Acronyms

PGO	Pose Graph Optimisation
PPGO	Pose-Parameter Graph Optimisation
SLAM	Simultaneous Localisation and Mapping
GUI	Graphical User Interface
API	Application Programming Interface
ATE	Absolute Trajectory Error
RPE	Relative Position Error
RNG	Random Number Generator
MAP	Maximum a Posteriori

List of Symbols

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

General

ℓ	Constant sensor frame transformation, where $\ell = T_{\mathsf{R}}^{(\mathsf{S})} \in \mathcal{SE}(2)$.
O	Fixed global (or inertial) reference frame.
R_t	Robot-fixed reference frame at time t.
S_t	Sensor-fixed reference frame at time t.
$\Sigma_{\mathcal{I}}$	Measurement model covariance over node indices $\mathcal{I}$.
$\boldsymbol{v}\langle p^{\text{scale}}\rangle$	Dummy scaling factor vector constructor for parameter-node p^{scale}, where $\boldsymbol{v}\langle p^{\text{scale}}\rangle \in \mathbb{R}^3$.
$\mathbf{e}_{\mathcal{I}}(\cdot)$	Measurement error function over node indices $\mathcal{I}$.
$\mathbf{t}(x, y)$	Translation vector constructor for elements $x, y \in \mathbb{R}$, where $\mathbf{t}(x, y) \in \mathbb{R}^2$.
$\mathbf{t}[T]$	Translation element of transformation $T \in \mathcal{SE}(2)$, where $\mathbf{t}[T] \in \mathbb{R}^2$.

$\mathbf{t}\langle p \rangle$	Dummy translation vector constructor for parameter-node p, where $\mathbf{t}\langle p \rangle \in \mathbb{R}^2$.
$\mathbf{t}_{\mathsf{A}}^{(\mathsf{B})}$	Relative coordinate frame translation of A with respect to B, where $\mathbf{t}_{\mathsf{A}}^{(\mathsf{B})} \in \mathbb{R}^2$.
$\mathbf{t}_i$	Translation component of pose x_i, where $\mathbf{t}_i = \mathbf{t}_{\mathsf{R}_i}^{(\mathsf{O})} \in \mathbb{R}^2$.
$F_{\mathcal{I}}(\cdot)$	Constraint cost function over node indices $\mathcal{I}$.
$f_{\mathcal{I}}(\cdot)$	Measurement model function over node indices $\mathcal{I}$.
$f_{\mathcal{I}}^{\text{loc.}}(\cdot)$	Location prior measurement model over node indices $\mathcal{I}$, with $f_{\mathcal{I}}^{\text{loc.}} : \mathcal{N} \to \mathbb{R}^2$.
$f_{\mathcal{I}}^{\text{rot.}}(\cdot)$	Pose rotation measurement model over node indices $\mathcal{I}$, with $f_{\mathcal{I}}^{\text{rot.}} : \mathcal{N} \to \mathcal{SO}(2)$.
$f_{\mathcal{I}}^{\text{transf.}}(\cdot)$	Pose transformation measurement model over node indices $\mathcal{I}$, with $f_{\mathcal{I}}^{\text{transf.}} : \mathcal{N} \to \mathcal{SE}(2)$.
$f_{\mathcal{I}}^{\text{transl.}}(\cdot)$	Pose translation measurement model over node indices $\mathcal{I}$, with $f_{\mathcal{I}}^{\text{transl.}} : \mathcal{N} \to \mathbb{R}^2$.
$F_{\text{pgo}}(\cdot)$	PGO cost function formalised by the pose graph.
$F_{\text{ppgo}}(\cdot)$	PPGO cost function formalised by the pose-parameter graph.
g	Factor graph consisting of nodes and poses.
$m_{\mathcal{I}}$	Dimensionality of the identifying measurement vector of the measurement relating nodes with indices $\mathcal{I}$.
n	Dimensionality of the Cartesian space in which the SLAM problem is defined.
$R(\theta)$	Rotation element constructor for angle $\theta \in \mathbb{R}$, where $R(\theta) \in \mathcal{SO}(2)$.
$R[T]$	Rotation element of transformation $T \in \mathcal{SE}(2)$, where $R[T] \in \mathcal{SO}(2)$.
$R\langle p \rangle$	Dummy rotation element constructor for parameter-node p, where $R\langle p \rangle \in \mathcal{SO}(2)$.
$R_{\mathsf{A}}^{(\mathsf{B})}$	Relative coordinate frame rotation of A with respect to B, where $R_{\mathsf{A}}^{(\mathsf{B})} \in \mathcal{SO}(2)$.
R_i	Rotation component of pose x_i, where $R_i = R_{\mathsf{R}_i}^{(\mathsf{O})} \in \mathcal{SO}(2)$.
$T(\boldsymbol{\xi})$	Transformation element constructor for identifying vector $\boldsymbol{\xi} = (x, y, \theta) \in \mathbb{R}^2$, where $T(\boldsymbol{\xi}) \in \mathcal{SE}(2)$.
$T(\mathbf{t}, R)$	Transformation element constructor for translation $\mathbf{t} \in \mathbb{R}^2$ and rotation $R \in \mathcal{SO}(2)$, where $T(\mathbf{t}, R) \in \mathcal{SE}(2)$.
$T(x, y, \theta)$	Transformation element constructor for elements $x, y, \theta \in \mathbb{R}$, where $T(x, y, \theta) \in \mathcal{SE}(2)$.
$T\langle p \rangle$	Dummy transformation element constructor for parameter-node p, where $T\langle p \rangle \in \mathcal{SE}(2)$.
$T_{\mathsf{A}}^{(\mathsf{B})}$	Relative coordinate frame transformation of A with respect to B, where $T_{\mathsf{A}}^{(\mathsf{B})} \in \mathcal{SE}(2)$.
x_i	Robot pose with index i, where $x_i = T_{\mathsf{R}_i}^{(\mathsf{O})} \in \mathcal{X} \subset \mathcal{SE}(2)$.
$z_{\mathcal{I}}$	Measurement relating nodes with indices $\mathcal{I}$, where $z_{\mathcal{I}} \in \mathcal{M}$.
Operators	
$\boxminus$	Generalised subtraction operator, $\boxminus : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^m$ for measurement set $\mathcal{Z}$.
$\boxplus$	Generalised addition operator, $\boxplus : \mathcal{M} \times \mathbb{R}^m \to \mathcal{M}$ for measurement set $\mathcal{Z}$.
$\odot$	Element-wise vector product.
$\ominus$	Minus operator, $\ominus : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^{n_{\mathcal{G}}}$ for Lie group $\mathcal{G}$.
$\oplus$	Plus operator, $\oplus : \mathcal{G} \times \mathbb{R}^{n_{\mathcal{G}}} \to \mathcal{G}$ for Lie group $\mathcal{G}$.
Sets	
$\mathcal{I}$	Set of node indices, where $\mathcal{I} = \{i\}$.
$\mathcal{M}$	Measurement space, with $\mathcal{M} = \mathcal{SE}(2) \cup \mathcal{SO}(2) \cup \mathbb{R}^2$.
$\mathcal{N}$	Set of Pose-Parameter Graph Optimisation (PPGO) nodes, where $\mathcal{N} = \mathcal{X} \cup \mathcal{P}$.
$\mathcal{P}$	Set of parameter-nodes, where $\mathcal{P} = \{p\}$.
$\mathcal{SE}(n)$	Special Euclidean group of transformations, with dimension n.
$\mathcal{SO}(n)$	Special Orthogonal group of rotations, with dimension n.
$\mathcal{X}$	Set of robot poses, where $\mathcal{X} = \{x\} \subset \mathcal{SE}(2)$.
$\mathcal{Z}$	Set of measurements, where $\mathcal{Z} = \{z\} \subset \mathcal{SE}(2) \cup \mathcal{SO}(2) \cup \mathbb{R}^2$.