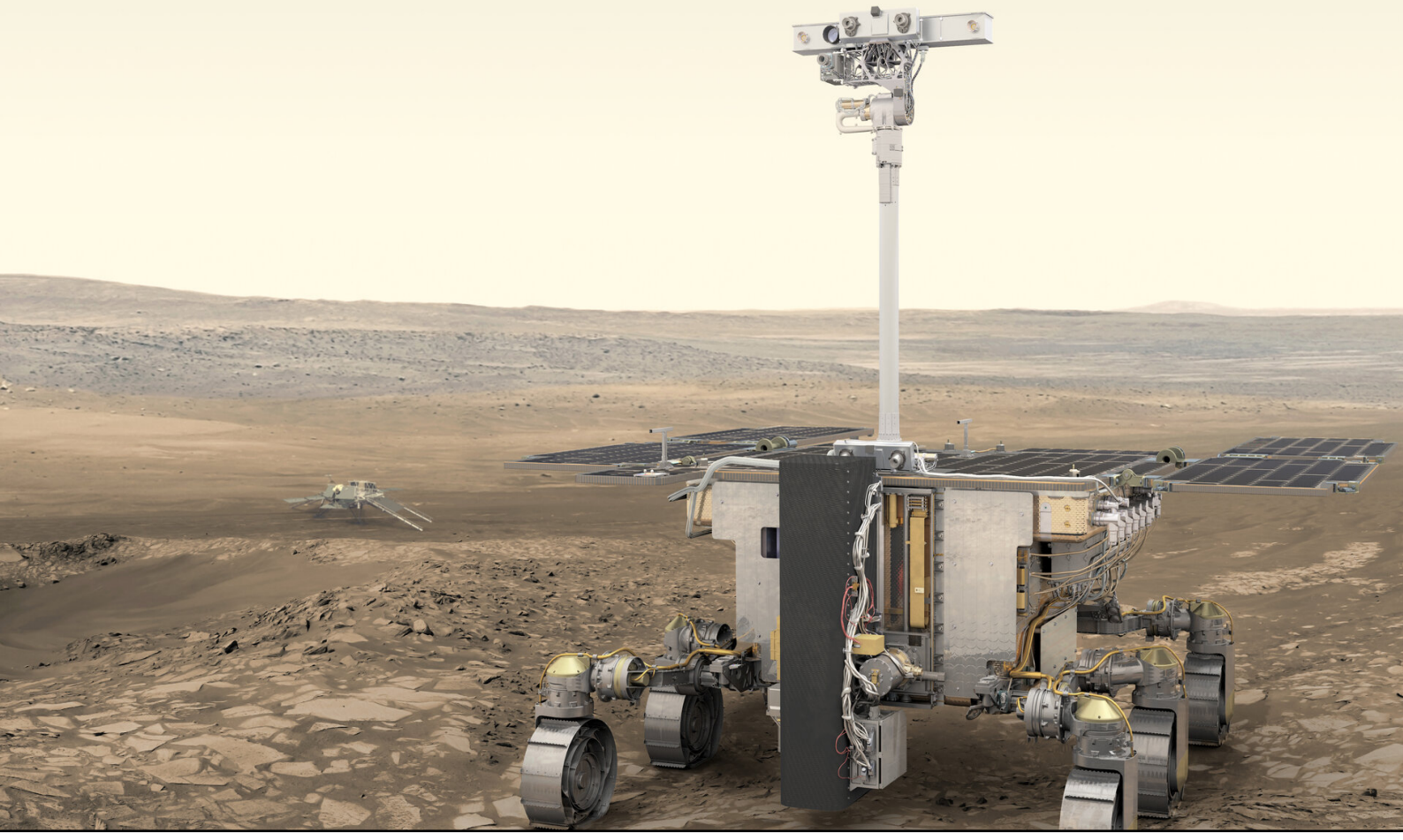
Sensor Fusion for Visual-Inertial Simultaneous Localisation and Mapping

Applied and tested on a small ground-based mini rover

Sam Marijn Bekkers



TU Delft Delft University of Technology

DLR German Aerospace Center

Sensor Fusion for Visual-Inertial Simultaneous Localisation and Mapping

Applied and tested on a small ground-based mini rover

Master Thesis

Sam Marijn Bekkers
4571770

January 23, 2024

Supervisor: Dr. C. Della Santina

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

Cover image: ESA's Mars rover has a name – Rosalind Franklin

Deutsches Zentrum
für Luft- und Raumfahrt
German Aerospace Center

Delft University of Technology
Department of
CoR

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Sensor Fusion for Visual-Inertial Simultaneous Localisation and Mapping

by

Sam Marijn Bekkers

in partial fulfillment of the requirements for the degree of

Master of Science Robotics

Dated: January 23, 2024

Supervisor(s):

dr. Cosimo Della Santina Supervisor

dr.ir. Armin Wedler Daily supervisor

Reader(s):

dr. Javier Alonso Mora First Reader

Abstract

The generation of a 3D map of an unseen environment, obtained through solving the SLAM problem, is a popular topic currently in the robotics domain. The Lunar Rover Mini (LRM) at the German Aerospace Center solves this problem using a RGB-D camera system, which is favourable in space applications due to its lightweight characteristics and energy-efficiency. Performing SLAM based on camera images is based on visual odometry: the science of estimating the rover's trajectory trough a sequence of images. However, the dependency on a single sensor to perform mapping and navigation poses a threat to the reliability of the system. To increase the reliability and robustness of the SLAM algorithm, an inertial measurement unit (IMU) is incorporated in the robot hardware.

This thesis describes the design for a visual-inertial SLAM algorithm that incorporates both visual and inertial measurements to solve the SLAM problem through performing tightly coupled sensor fusion, which estimates and corrects for IMU biases. The solution is based on a non-linear factor graph, which is a graphical model to represent the relationships between the rover's measurements and the unknown variables which are optimised for. This is done using the open-source GTSAM framework. Using experimental data, the robustness of the novel visual-inertial SLAM algorithm is demonstrated by simulating specific sensor failures. Moreover, the novel algorithm shows its capability to incorporate a degree of certainty regarding specific areas of the generated map, closely resembling how a human being would generate a map of an unknown area.

An additional use case for tightly coupled sensor fusion is the increased accuracy of the estimated trajectory. Assuming Gaussian noise models for both measurement models, averaging the two can yield a higher accuracy than either of the two sensors could have obtained by itself. This hypothesis was tested in another experiment. As the main mechanism behind bias estimation is reducing the error between visual and inertial measurements, bias estimation is quickly affected by this drifting visual odometry, which in its turn deteriorates the accuracy of the visual-inertial odometry module. This observation proves that the bias estimation is not correlated to the underlying physical process, but is rather just a numerical value in the optimisation reducing the residual error. It raises the question whether this strategy of tightly coupled sensor fusion can actually be used to increase the accuracy of a visual odometry algorithm.

Table of Contents

Preface

This document contains my thesis concerning sensor fusion for a small ground-based mini rover. This thesis was conducted at the German Aerospace Center (DLR) in Oberpfaffenhofen, Germany at the Institute for Robotics and Mechatronics (RMC).

My gratitude goes out towards my colleagues at DLR who have supported this work. Dr.-Ing. Armin Wedler has established a very professional yet comfortable environment in his student team, in which I was able to work, learn and develop freely. In particular the discussions with Dr. Riccardo Giubilato have provided me with a lot of insight into probabilistic robotics, a topic which is very applicable to this thesis.

The developments in this thesis rely heavily on the GTSAM toolbox, developed and published by Frank Dellaert and the Georgia Institute of Technology. This work is financially supported by the Erasmus+ Study Scholarship.

Sam Marijn Bekkers

Chapter 1

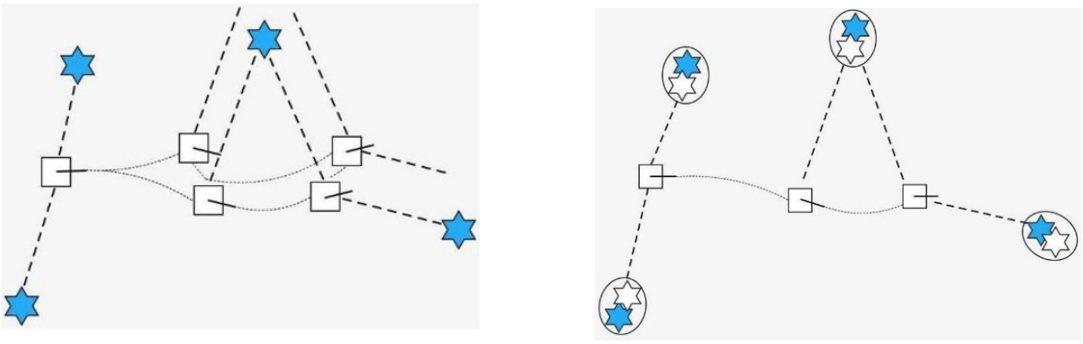
Introduction

In the light of exploration of our solar system, a lot of research has been conducted in the last decades regarding Earth's fellow planets. Because of its solid surface, polar ice caps, extinct volcanoes and traces of liquid water, the planet Mars is of high interest to researchers and space agencies. In order to explore the surface of this planet, robotic rovers have been sent to the red planet, with the aim of gaining knowledge regarding the planet's climate, surface structure, soil composition and other characteristics, all with the aim to prepare mankind for the first human step on another planet. A substantial part of this objective is to construct a map of the explored terrain, which contains valuable information for future manned or unmanned missions.

Due to the long round-trip communication time, which can take up to 20 minutes for Mars, tele-operation of these robots from an Earth-based control station is impractical. It is therefore important that the rovers can navigate and explore the surface of distant celestial bodies autonomously. To perform mapping and navigation tasks, the rover uses its sensors to answer the following questions questions:

- *"Where am I now?"*
- *"What does my environment look like?"*

Finding the answers to these questions is commonly known in robotics as solving the **localisation problem** and the **mapping problem**, respectively. Assuming that the rover is equipped with sensors that allow it to determine the distance to objects in its rover, the generation a map of the environment might seem straightforward. It quickly becomes clear however that the mapping problem can only be solved if the rovers exact location is known at the time of making a depth measurement. On the other hand, exactly knowing one's location can only be done if a complete map is at hand, to which the depth measurements can be compared to figure out what the current position is. The complexity of the problem becomes evident now, as the problem starts to resemble a chicken-and-egg problem: to solve the mapping problem, the localisation problem has to be solved, for which in its turn first the



(a) Localisation using the known position of landmarks

(b) Mapping using the robot's known position

Figure 1-1: The localisation and the mapping problem visualised

mapping problem has to be solved. This problem is illustrated in figure 1-1 [1]. Finding the answers to the localisation and mapping problem thus requires a more sophisticated solution, one that can jointly and simultaneously solve both problems at the same time. The problem is hence referred to as the Simultaneous Localisation and Mapping problem, or shortly, SLAM.

In the domain of space exploration, boundaries exist to the amount of sensors that can be utilised by a Mars rover. A commonly used sensor in autonomous cars for example is a LiDAR (Light Detection and Ranging). These sensors are typically quite heavy and consume a lot of energy, due to their measuring technique. As it is very costly to launch a rocket carrying a rover equipped with heavy batteries, this type of sensor is not suitable for solving the SLAM problem on distant planets. In the space domain, rovers are limited to the use of cameras or a fusion of camera's and inertial measurement units (IMU's). Therefore, this thesis focuses on what is called *visual SLAM* and *visual-inertial SLAM*.

The thesis project revolves around the improvement of an existing SLAM algorithm on the Lunar Rover Mini, or LRM, in close collaboration with the German Aerospace Center (DLR) in Oberpfaffenhofen, Germany. This small-ground based rover is a long-term project which aims to serve as an open-source experimental robotic platform for researchers and students. The LRM is equipped with a RealSense D435i carrying an RGB-D camera and an IMU. As a product from a previous master thesis, the rover already utilised the camera measurements into a visual SLAM pipeline to perform localisation and mapping. The inertial measurements from the IMU were not utilised yet however. Herein lied an opportunity to improve the SLAM capabilities of the rover. The research question considered throughout this thesis is as follows:

"How can an inertial measurement unit (IMU) assist to increase the robustness of a visual SLAM pipeline?"

The next chapter presents a summary of preliminary knowledge regarding the material discussed in the thesis. In chapter 3, the issues related to pure visual SLAM and the initial state of the thesis project is laid out. Chapter 4 elaborates on the theory behind visual-inertial SLAM systems is discussed, which is needed to understand the methodology and software architecture of the work done for this thesis, which is introduced in chapter 5. This chapter also explains the experimental setup in which the performance of the algorithms is evaluated.

The results from these experiments are presented in chapter 6 and lastly, the research question is answered in the chapter 7, which offers the conclusion to this thesis and poses some points of discussion.

Chapter 2

Preliminary Knowledge

This chapter discusses some preliminary knowledge regarding the *photogrammetry*, the science of extracting 3D information from a set of 2D images. The mathematics behind this topic provide tools to understand how these cameras can serve as the main sensor for robots to construct a 3D map of their environment, and estimate their position in it.

2-1 Pinhole Camera Model

In a mathematical sense, a camera performs a *mapping* of $\mathbb{R}^3$, the 3D world, to $\mathbb{R}^2$, the 2D image plane. The *pinhole camera model* is the simplest mathematical model that describes this mapping, i.e. the perception of 3D points through projection onto an image plane [2]. Let the location of a 3D point be denoted by $\mathbf{X} = \{x, y, z\}$, as illustrated in figure 2-1. The projection of $\mathbf{X}$ is the intersection between the line that connects the camera centre $\mathbf{C}$ and the image plane. The coordinates of $\mathbf{x}$, the mapping of $\mathbf{X}$ onto the image plane, are:

$$\mathbf{x} = \left[f\frac{X}{Z}, f\frac{Y}{Z} \right] \tag{2-1}$$

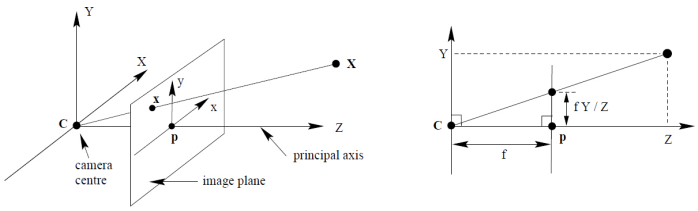


Figure 2-1: Pinhole camera model

where the parameter f is the *focal length* of the camera. In matrix formulation, the general expression of the pinhole model is as follows:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \rho & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \tag{2-2}$$

in which u and v are the pixel coordinates on the image plane, f_x and f_y are the focal lengths in x- and y-direction, respectively, and ρ is a factor for skewness of the sensor. Notice here that by performing a mapping from $\mathbb{R}^3$ to $\mathbb{R}^2$, information regarding one of the dimensions is disregarded. This becomes clear by virtually moving point $\mathbf{X}$ in figure 2-1 along its projection line. The coordinates of projection $\mathbf{x}$ on the image plane will not be affected by this linear movement.

As described above, the pinhole camera model can be used to describe the 3D to 2D mapping of points. However, in the domain of visual SLAM, the aim is to retrieve depth information from 2D images. Recall that in the 3D-to-2D mapping, depth information is discarded. This poses a problem for the reverse mapping, as it is impossible to retrieve this 3D information solely from a 2D image, where the only information known about a point is its position in the image plane, denoted by u, v. Figure 2-2 illustrates this information is just enough to reconstruct the ray in the 3D world on which the point has to exist. The scaling factor λ however is unknown, which determines where exactly on this ray the point resides [3]:

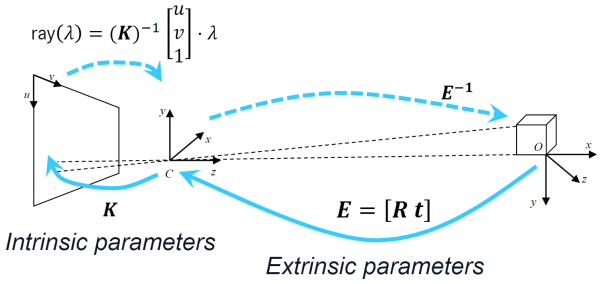
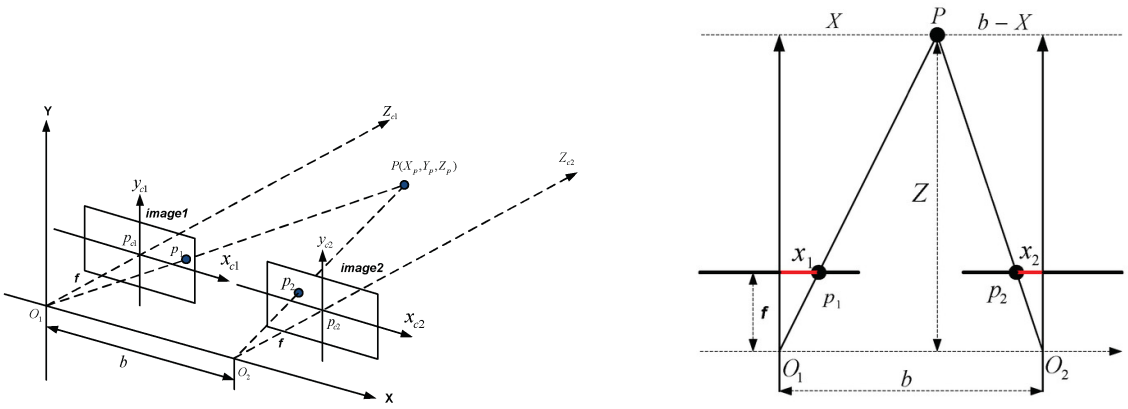


Figure 2-2: Inverse mapping of the pinhole camera model

2-2 Epipolar Geometry

The perception of depth from visual cues can be achieved by taking multiple viewpoints from different locations into consideration. The biological analogy to this is the fact that human beings are equipped with two "pinhole camera's", being our eyes. Inspired by this feat of biological evolution, the robotic domain mimics this principle by incorporating multiple viewpoints to estimate depth in our environment. The geometry across two different viewpoints is referred to as *epipolar geometry*, which is visualised in figure 2-3:



(a) Configuration of stereo vision **(b)** Triangulation for depth perception

Figure 2-3: Epipolar geometry for depth estimation by means of point triangulation

Figure 2-3a illustrates how the 3D location of point P is computed at the intersection of the lines O_1P and O_2P [4]. The parallax $d = x_1 + x_2$ is a measure for the apparent shift of an object due to observing this object from different viewpoints. Using this measure the location of P is found by:

$$\begin{aligned} Z_p &= b * f/d \\ X_p &= Z_p * x_1/f \\ Y_p &= Z_p * y_1/f \end{aligned} \tag{2-3}$$

where b is the baseline distance between the camera centres and f is the focal length of the camera's. This calculation requires knowledge about correspondences between features in the left and right image. Chapter 3 will elaborate on the extraction and recognition of features in both images.

2-3 Bundle Adjustment

Bundle Adjustment is a process used in feature-based visual SLAM methods to estimate both the camera motion between two sequential images, as well as the 3D location of photographed features [5]. It does so by minimising the sum of squared *reprojection errors*. The reprojection error is the distance between the location where feature f_i is measured in an image, and the location of the projection of the estimate of f_i. This is visualised in figure 2-4:

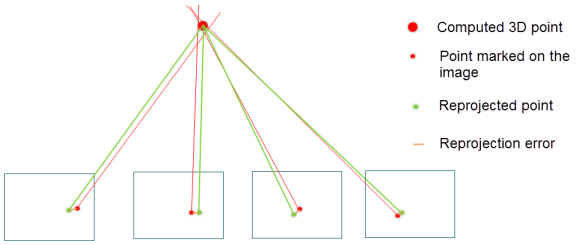


Figure 2-4: Bundle Adjustment through minimising the reprojection error

On the bottom of figure 2-4 a set of sequential images are shown, showing the measured projection of a 3D point P in red. If this point P is only captured in two images, the estimated location of this point in 3D is easily retrieved at the intersection of the two rays, computed by triangulation. If the point is then observed in a third image, it is extremely unlikely that the three projective rays (shown in red) from the three camera centres exactly line up in the current estimate of point P. This means that the current estimate for the location of P agrees with the measurements taken in the first two images, but not so much with the third image. This is quantified by reprojecting the current estimate of P back onto the image plane of the third image, as shown in green. The reprojection error is then defined as the distance between the reprojection of the estimate and the actual measurement, as shown in orange. By minimising the sum of the squares of all reprojection errors (four in the example above), an optimal solution can be found, which tries to satisfy the evidence from all four measurements as much as possible. It is clear that the estimated location of P can be refined by adding more observations into the minimisation problem.

Besides refining the estimate of the 3D structure of the environment, bundle adjustment also finds the optimal transformation between camera images by minimising the same reprojection error. The reprojection error can be mathematically defined as:

$$\epsilon_{ij}(\mathbf{x}_{ij}; \boldsymbol{\theta}_i, \boldsymbol{\xi}_j) = \| \mathbf{x}_{ij} - \pi(\mathbf{T}(\boldsymbol{\theta}_i), \mathbf{X}(\boldsymbol{\xi}_j)) \| \tag{2-4}$$

in which $\mathbf{x}_{ij}$ actual measured location of point j in image i, $\boldsymbol{\theta}_i$ contains the intrinsic and extrinsic viewing parameter of the camera for image i and $\boldsymbol{\xi}_j$ is the best estimate of the world coordinate of point j. $\pi(\cdot, \cdot)$ is the projection function.

The mathematical definition of bundle adjustment then becomes:

$$\boldsymbol{\theta}^*, \boldsymbol{\xi}^* = \underset{\boldsymbol{\theta}_i, \boldsymbol{\xi}_j}{argmin} \sum_{i=1} \sum_{j=1} \frac{1}{2} \delta_{ij} \epsilon_{ij}^2 (\mathbf{x}_{ij}; \boldsymbol{\theta}_i, \boldsymbol{\xi}_j) \tag{2-5}$$

It finds the set of optimal camera transformations $\boldsymbol{\theta}^*$ and 3D points $\boldsymbol{\xi}^*$ that are able to most accurately explain the observed measurements $\mathbf{x}_{ij}$. Bundle Adjustment can also be used to retrieve a whole set of camera transformations, i.e. finding the robot trajectory. The algorithm then serves as SLAM back-end, which is further elaborated on in chapter 5.

Bundle adjustment typically refers to 2D-2D image registration, meaning that the camera movement is estimated through comparing visual features in two consecutive 2D images. In

SLAM applications, as an estimate of the 3D world is generated, it is useful to be able to perform similar image registration, but based on the 3D world and the features recorded in the last 2D image. This can be done using the Perspective-n-Point (PnP) algorithm, which estimates the pose of the camera based on the correlations between features in the 2D image and a set of 3D world points. The PnP problem is defined as:

$$\underbrace{s\begin{bmatrix}u\\v\\1\end{bmatrix}}_{p_c} = \underbrace{\begin{bmatrix}f_x & \rho & c_x\\0 & f_y & c_y\\0 & 0 & 1\end{bmatrix}}_{K}\underbrace{\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_1\\r_{21} & r_{22} & r_{23} & t_2\\r_{31} & r_{32} & r_{33} & t_3\end{bmatrix}}_{[R|T]}\underbrace{\begin{bmatrix}x\\y\\z\\1\end{bmatrix}}_{p_w} \tag{2-6}$$

In this equation, p_c are the image coordinates of a 2D feature, p_w are the world coordinates of the corresponding 3D point, K is the calibration matrix of the camera and $[R|T]$ is the transformation matrix that describes the camera pose in the world frame. To find a solution, a minimum number of corresponding points is needed. For calibrated camera's, with known calibration matrix K, this amount of points is 5. For uncalibrated cameras, 8 points of reference are needed. The PnP problem is hence often referred to as the 5-point algorithm or the 8-point-algorithm.

2-4 Monocular and Stereo Camera

In order to make use of the epipolar geometric methods discussed before, images have to be captured from different viewpoints, for which two different methods exist. The first method uses a *monocular camera* setup, which is a single camera moving through space. It captures an image at timestamp t_n, then moves through space and captures the next image at timestamp t_{n+1}. The second method utilises a so-called *stereo camera* setup. In this setup two camera's are rigidly mounted to the moving platform with a known distance and orientation relative to each other. The first SLAM systems however were primarily based on monocular setups [6–8]. As they already were limited in computational power by the available computers at that time, utilising a stereo setup and analysing twice the amount of images was simply not feasible.

The main drawback for monocular camera setups is that absolute scale estimation is theoretically impossible. This is visualised in figure 2-5. It demonstrates that the camera transformation C_{c1} to capture the down-scaled object, results in the same image of performing transformation C_{c2} to capture the real object. Although the rotation between frame C_p and C_{ci} for $i = \{1, 2, 3...\}$ and the *direction* of the translation can be found, but the actual distance in this direction is impossible to determine [9]. A common workaround in monocular SLAM methods like [6, 10, 11] is to perform a specific initialisation procedure with the aim to solve the 5-point or 8-point algorithm, as covered in the previous section. This requires however a scene or object with known dimensions, from which the scale can be determined and used as reference later during operation.

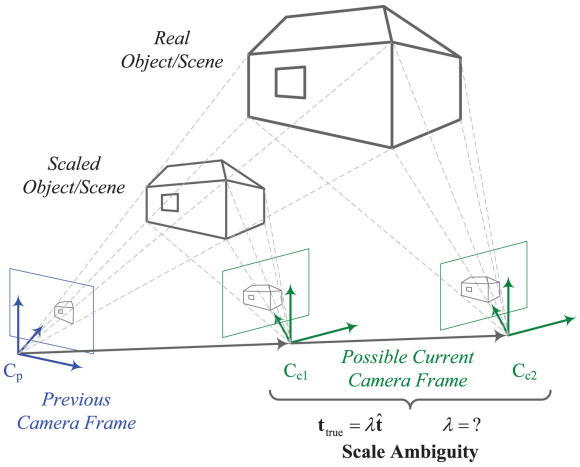


Figure 2-5: Demonstration of scale ambiguity for monocular camera setups

Once a successful initialisation is complete, the algorithm has a reference for the scale of the solution. If new points are observed with unknown 3D location, the algorithm can deduct the camera motion from the set of already estimated points in previous timestamps. Then, based on the optimised camera motion, the algorithm estimates the depth information of new points by performing local BA. This poses a new problem however: since the algorithm is now using an estimated camera translation based on the result of previous bundle adjustments, any error in this estimated camera translation propagates into the estimation of 3D point location, which in its turn again influences future estimations of camera translations. This is a cycle in which the error builds up, leading to a phenomenon referred to as *scale drift*.

For a stereo camera setup the transformation (the baseline distance) between the two camera centres is known, which eliminates the problem of direct scale estimation. Moreover, scale drift is not an issue because the depth estimation can be performed directly from the known baseline distance, instead of using an estimated transformation as baseline which is the cause of scale drift in monocular camera setups. This helps to more accurately create 3D reconstructions of the environment.

2-5 Filtering vs Optimisation based SLAM

The SLAM problem concerns the reconstruction of the environment, whilst localising the robot in this map at the same time. Within the problem framework, the following is given:

- **The robot's control inputs:** $u_{1:t} = \{u_1, u_2, u_3..., u_t\}$
- **The robot's observations:** $z_{1:t} = \{z_1, z_2, z_3..., z_t\}$

The core of the SLAM problem boils down to estimating the entire trajectory of the robot and the map of the environment, represented by a set of 3D feature locations. This can be mathematically denoted as estimating the following probability distribution:

$$p(x_{0:t}, m|z_{1:t}, u_{1:t}) \tag{2-7}$$

Available literature distinguishes two main forms of the SLAM problem: the *full SLAM* problem and the *online SLAM* problem. The full SLAM problem aims to estimate the posterior probability over the entire robot path together with the map, as it is defined by 2-7. The online SLAM problem though focuses on recovering the posterior probability distribution of only the current state x_t, instead of the full path $x0 : t$:

$$p(x_t, m|z_{1:t}, u_{1:t}) \tag{2-8}$$

By repeatedly solving the online SLAM problem for every timestamp, an solution to the full problem is obtained. This is the strategy employed by filter-based SLAM algorithms. The most filter-based approaches are based on Extended Kalman Filters and Particle Filters [12,13]. Solutions that solve the full SLAM problem at once are optimisation or batch-solving based solutions. Both strategies have advantages and drawbacks. Filter-based solutions tend to have a lower computational load since old states are marginalised out. Optimisation based algorithms on the other hand yield high accuracies, since the solution is estimated based on áll evidence, instead of only the last measurements. Moreover, they allow for loop closure detection, which effectively remove drift from the estimated trajectory. This is illustrated in figure 2-6 [14]:

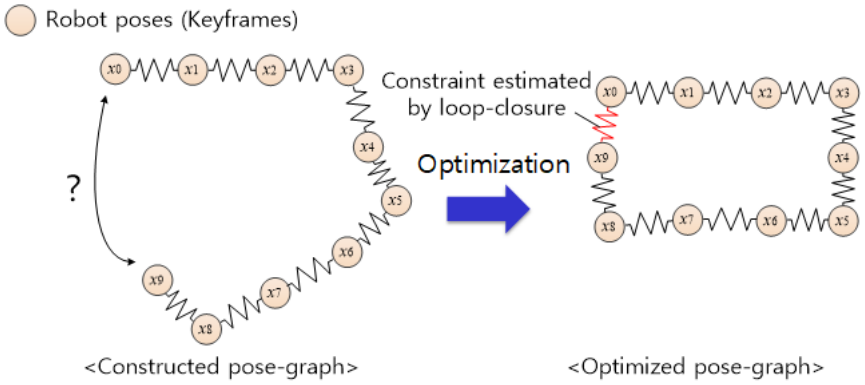
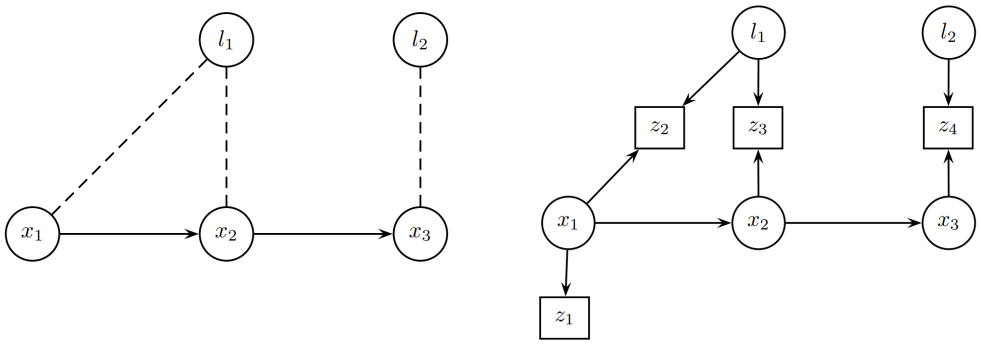


Figure 2-6: Example of a Loop Closure

In [15] is demonstrated that for an equal computational load, optimisation based approaches obtain a higher accuracy. For this reason, in combination with the powerful loop closure tool, this thesis work focuses on the implementation of optimisation-based SLAM algorithms.

2-6 Bayesian Networks and Factor Graphs

A complex probability density in problems such as optimisation-based SLAM can be described by so-called probabilistic graphical models by exploiting the structure in them. A very useful trick is that high-dimensional probability distributions often can be factorised into a product of multiple factors, which all describe the probability distribution of a smaller domain of the problem.



(a) SLAM example with three poses and two landmarks

(b) Bayesian Network of the SLAM example

In the example SLAM problem in figure 2-7a, two landmarks l_1 and l_2 are observed from three robot states x_1, x_2, and x_3, resulting in the state vector $X = \{x_1, x_2, x_3, l_1, l_2\}$. A **Bayesian Network** or Bayes Net helps to summarise the joint probability distribution over all random variables, i.e: $\Theta = \{X, Z\}$ in which Z are the measurements. The Bayes Net is a directed graph in which the nodes represent a set of variables θ_j and subsequently defines a joint probability density on the full set, $p(\Theta)$ as the factorisation of conditional densities associate with each variable θ_j and its parents π_j:

$$p(\Theta) \triangleq \prod_j p(\theta_j|\pi_j) \tag{2-9}$$

Figure 2-7b shows the Bayes Net for the example from figure 2-7a. Note that between the robot states x_1 and x_2, and between x_2 and x_3, no measurement nodes are included. In this example, the movement of the robot is not measured, but the rover movement follow a certain motion model providing information how the rover moves from one state to another. As the example consists of nine nodes θ_j, the joint probability density consists of nine factors:

$$\begin{aligned} p(X, Z) = & \, p(x_1)p(x_2|x_1)p(x_3|x_2) \\ & \times p(l_1)p(l_2) \\ & \times p(z_1|x_1) \\ & \times p(z2|x_1, l_1)p(z_3|x_2, l_1)p(z_4|x_3, l_2) \end{aligned} \tag{2-10}$$

In which $p(x_1)p(x_2|x_1)p(x_3|x_2)$ is the Markov chain, describing how the robot moves through the environment, $p(l_1)p(l_2)$ are the prior densities on the landmarks, $p(z_1|x_1)$ is the prior

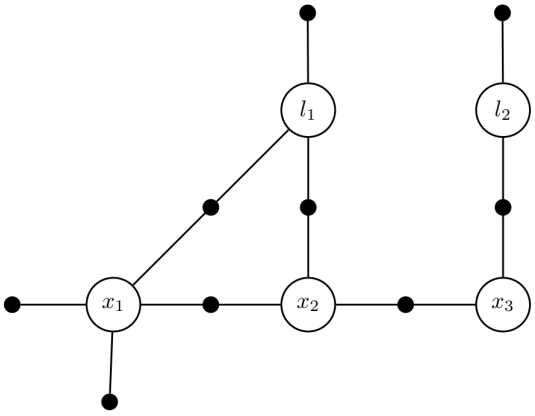


Figure 2-8: Factor graph example

density on the first pose, and $p(z2|x_1, l_1)p(z_3|x_2, l_1)p(z_4|x_3, l_2)$ corresponds to the conditional densities on the observations, given a robot position x_n and landmark position l_m.

In robotics, we are interested in the values of unknown state variables X, *given* the measurements Z, in other words: we are looking for the probability distribution $P(X|Z)$. Instead of modelling a complete scenario, for which a Bayes Net is useful, we are interested in performing inference: processing knowledge about the environment from their sensors. For this task, the **Factor Graph** is suitable. A Bayes Net is easily converted into a Factor Graph by removing the measurement variables, and instead of associating an unknown variable with a conditional density, unknowns are connected by a new node type: *factors*. A factor node ϕ_j describes how two unknown variables are related to each other. The complete factor graph then again describes the factorisation of a global function $\phi(X)$, which corresponds to the density $P(X|Z)$:

$$p(X|Z) \propto \prod_j \phi_i(X_i) \tag{2-11}$$

The factor graph from the earlier example is shown in figure 2-8.

The probability densities in the earlier Bayes Net are multivariate Gaussian distributions, with probability density:

$$\mathcal{N}(\theta|\mu, \Sigma) = \frac{1}{\sqrt{2\pi\Sigma}} \exp\Big\{-\frac{1}{2}||\theta - \mu||^2_\Sigma\Big\} \tag{2-12}$$

with:

$$||\theta - \mu||^2_\Sigma = (\theta - \mu)^T \Sigma^{-1} (\theta - \mu) \tag{2-13}$$

Therefore, the factors can be assumed to be of the form:

$$\phi_i(X_i) \propto \exp\Big\{-\frac{1}{2}||h_i(X_i) - z_i||^2_{\Sigma_i}\Big\} \tag{2-14}$$

Here, $h_i(X_i)$ is a measurement prediction function and z_i is the actual measurement of variable i. To estimate the values of these variables, the Maximum a Posterior (MAP) estimate is used, which maximises the posterior density $p(X|Z)$ of the states X given the measurements Z. MAP inference boils down to maximising the product of the factor graph potential, i.e. the factorisation of all factors:

$$X^{MAP} = \underset{X}{argmax}\, \phi(X) \tag{2-15}$$

The combination of equations 2-14 and 2-15 yields the **nonlinear least-squares problem**:

$$X^{MAP} = \underset{X}{\operatorname{argmin}} \sum_i ||h_i(X_i) - z_i||^2_{\Sigma_i} \tag{2-16}$$

Given that the measurement functions $h_i(X_i)$ are nonlinear, the nonlinear least squares problem can be solved by methods such as Gauss-Newton or Levenberg-Marquardt to converge to a global minimum using a succession of linearisations. Linearisation of the measurement functions h_i is done through a first order Taylor expansion:

$$h_i(X_i) = h_i(X_i^0 + \Delta_i) \approx h_i(X_i^0) + H_i\Delta_i \tag{2-17}$$

in which H_i is the **measurement Jacobian**, which is defined as:

$$H_i = \frac{\partial h_i(X_i)}{\partial X_i} \tag{2-18}$$

In equation 2-17, X_i^0 is the initial guess and $\Delta_i = X_i - X_i^0$ is the state update vector. The next step is to substitute the linearised Taylor expansion from equation 2-17 into the nonlinear least-squares problem as defined in 2-16 to obtain the following linear least-squares problem:

$$\Delta^* = \underset{\Delta}{\operatorname{argmin}} \sum_i ||H_i(\Delta_i) - \{z_i - h_i(X_i^0)\}||^2_{\Sigma_i} \tag{2-19}$$

After rearranging the Mahalanobis Distance term, the following standard least-squares problem is obtained:

$$\begin{aligned} \Delta^* &= \underset{\Delta}{\operatorname{argmin}} \sum_i ||A_i\Delta_i - b_i||^2 \\ &\text{with:} \\ A_i &= \Sigma_i^{-1/2} H_i \\ b_i &= \Sigma_i^{-1/2}(z_i - h_i(X_i^0)) \end{aligned} \tag{2-20}$$

The difference between Gauss-Newton and Levenberg-Marquardt is the way in which relinearisation occurs, and how getting stuck in local minima is prevented.

Since inference problems in the robotics domain are typically incremental, meaning that new measurements continuously extend the size of the problem, one can wonder if it is necessary to compute the full solution every time, or whether previous computations can be reused. This is the core of iSAM [16] (incremental Smoothing and Mapping). Instead of refactorising the problem for every new set of measurements, it updates the existing matrix A using Givens rotations.

Chapter 3

Problem Statement

The work in this thesis is based on a small robotic platform, the Lunar Rover Mini (LRM). Inspired by ESA's Rosalind Franklin Mars rover, the aim of the LRM-project is to establish a low-cost development and testing platform, for the purpose of learning and experimenting.

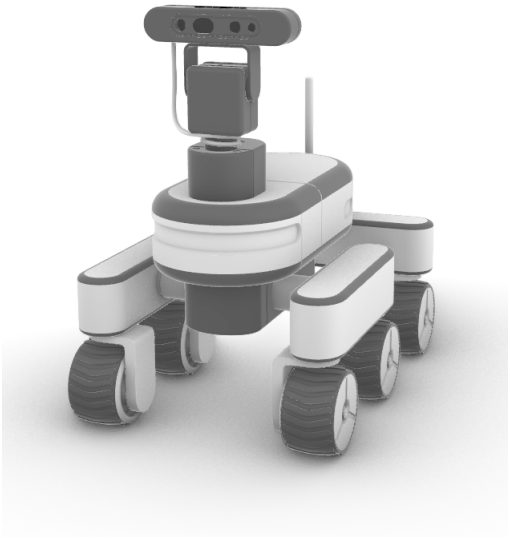


Figure 3-1: The Lunar Rover Mini

As a product of earlier student projects, the LRM is already able to perform various autonomous tasks such as locomotion, navigation and mapping. At the start of this thesis, the mapping thread used to consist of a plain visual odometry pipeline, which makes use of the onboard RealSense D435i sensor. This device provides the on-board computer (an Intel NUC) with Red Green Blue and Depth (RGB-D) images. The visual SLAM algorithm was developed using the open-source available RTAB-map (Real-Time Appearance-Based Mapping)

software [17]. It is a framework for RGB-D, Stereo and LiDAR mapping, which employs an incremental loop closure detector to obtain a good accuracy through harvesting the power of loop closures. By using the Bag-of-Words approach, it generates a loop closure hypothesis for every images. This hypothesis is based on how confident the algorithm is regarding a potentially closed loop. If this hypothesis is accepted, an additional constraint is added to the map, which closes the loop and reduces the drift error in that loop.

The reliability on a single sensor, being a camera in this case, poses a threat to the robustness of the system. Although it can be assumed that the camera itself works all the time, scenarios exist in which the visual odometry pipeline might fail to calculate motion estimates from the sequence of images. An example of such a scenario is the traversing of a feature-poor environment, as illustrated in figure 3-2. Other examples could be a change in lighting conditions, which occurs when directly looking into the sunlight, or quickly rotating around the vertical axis of the rover, causing motion blur. As the visual odometry pipeline relies heavily on the extraction of features, both the image quality and feature density of the environment directly influence the rover's capability to solve the SLAM problem. This introduces a major flaw in the current implementation of visual SLAM on the LRM, as the scenario's laid out above are certainly not uncommon. The quality of the captured images potentially quickly deteriorates, introducing a major weakness in pure visual SLAM systems. A failure in this crucial feature detection mechanism may cause the complete SLAM system to fail.

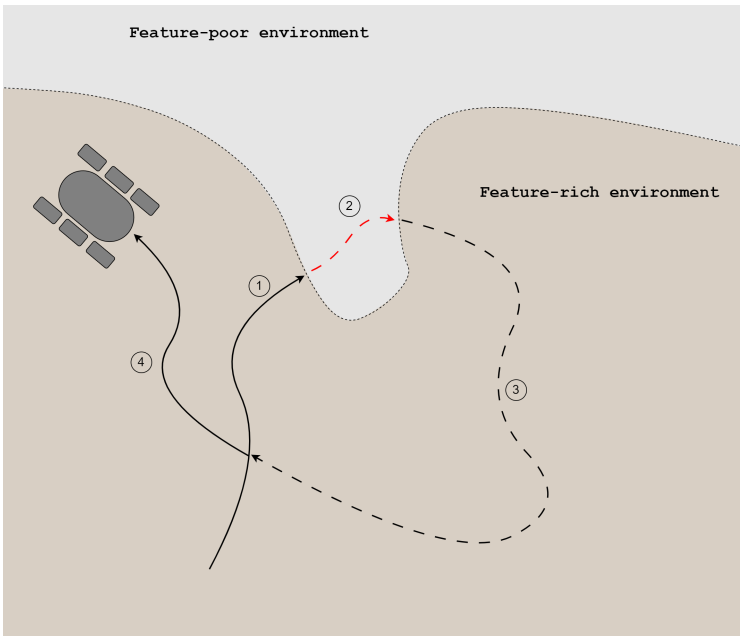


Figure 3-2: Relocalisation after loss of features

The figure above illustrates the problems of the current implementation of SLAM when it experiences a loss of features to track. In this figure, four sequential trajectories are indicated. In the first trajectory, the rover traverses the beige, feature-rich area. The rover successfully estimating its motion and trajectory through visual odometry and extends the global environment map with every observation. The second trajectory however traverses a very feature-poor area, which leaves the visual odometry module useless. No motion can be estimated in this period and no new features are added to the map. The major loss occurs in

the third trajectory, in which technically there are enough features to perform visual odometry and estimate the motion of the rover. The problem is that these motion constraints plus the environment information gathered at these timestamps can not be added to the global map, since the rover does not know where it was when it made those observations. For this reason, all of the features observed in the third trajectory are essentially lost. It is only after the rover visits a previously visited location at which it can relocalise itself, before the new observations can be inserted to the global map.

The incorporation of an additional sensor that registers the motion of the rover could assist, serving as a back-up sensor in case of a visual odometry failure and preventing the whole SLAM algorithm from failing. An inertial measurement unit (IMU)is such a sensor and provides a stable source of information regarding the accelerations and angular velocities of the sensor. More importantly, the IMU is not sensitive to the failure modes of the visual odometry pipeline as explained above. IMUs are well-known in the robotics community for their light weight and energy efficiency, two crucial characteristics for robotic applications in space. By combining visual data with inertial measurements, the system becomes more robust and reliable in environments where pure visual odometry would have difficulties. The following chapter goes into the specific details regarding the IMU measurement model, and challenges that are introduced by the incorporation of this sensor.

Chapter 4

Visual-Inertial SLAM

This chapter focuses on the theory behind visual-inertial SLAM algorithms. These algorithms are based on a fusion of visual measurements from a camera and inertial measurements from an Inertial Measurement Unit (IMU). This sensor consists of a set of gyroscopes, accelerometers and sometimes magnetometers and provides information regarding the angular velocity and the linear acceleration of the IMU. Visual-inertial SLAM algorithms can be categorised based on the fusion type of IMU measurement into the camera measurement processing. *Loosely coupled* methods process the IMU measurements separately from the image measurements and use both measurements to track the robot pose. After tracking is computed separately, the outputs are fused for both measurement types. *Tightly coupled* methods however find a way to fuse the measurements before they are used to perform SLAM which enables the algorithm to find the solution given these already fused measurements, which increases the accuracy and robustness compared to loosely coupled algorithms [18]. For this reason, this thesis focuses on tightly fusing the two datastreams.

(a) RealSense D435i **(b)** BMI044 IMU

Figure 4-1: RealSense D435i and the internal BMI055 IMU

4-1 IMU Measurement Model

The first step in the process of sensor fusion is understanding which information is contained withing the IMU measurements, i.e. comprehending the **IMU measurement model**. The IMU has a 3-axis accelerometer and a 3-axis gyroscope, and may contain a magnetometer to detect the Earth's magnetic field. This feature is not needed for the purpose of visual-inertial SLAM. The measured linear acceleration $\tilde{\mathbf{a}}(t)$ and angular velocity $\tilde{\boldsymbol{\omega}}(t)$ are affected by white noise $\boldsymbol{\eta}$ and a sensor bias $\mathbf{b}$:

$$\begin{aligned} \tilde{\mathbf{a}}(t) &= R^T(t)\Big(\mathbf{a}(t) - \mathbf{g}\Big) + \mathbf{b}^a(t) + \boldsymbol{\eta}^a \\ \tilde{\boldsymbol{\omega}}(t) &= \boldsymbol{\omega}(t) + \mathbf{b}^g(t) + \boldsymbol{\eta}^g \end{aligned} \tag{4-1}$$

The white noises $\boldsymbol{\eta}^a$ and $\boldsymbol{\eta}^g$ are assumed to be Gaussian white noise. Therefore, they can hardly be estimated. The sensors biases $\mathbf{b}^a$ and $\mathbf{b}^g$ for the accelerometer and the gyroscope, respectively, are influenced by external factors such as temperature and vary slowly through time. They are modelled as random walk, which means that the *derivatives* are modeles as Gaussian white noise.

$$\dot{\mathbf{b}}^a \sim \mathcal{N}(0, \sigma_{b_a}),\ \dot{\mathbf{b}}^g \sim \mathcal{N}(0, \sigma_{b_g}) \tag{4-2}$$

Due to this slowly varying character, the biases are almost constant over a small time interval. Therefore, if the bias from timestamp t_{k-1} is known, the IMU measurements from timestamp t_k can be cleaned up by removing this bias. This is exactly why tightly coupled algorithms perform better than loosely coupled systems: by jointly optimising over both the visual measurements and the inertial measurements, the IMU biases can be estimated and deducted from the next measurement to approximately filter out the biases. Bias estimation is performed by incorporating the as unknown parameters in the state vector (EKF-based algorithms) or the cost function (optimisation based smoothing algorithms), and then figuring out which bias best fits the newly observed measurements, given the estimated biases from the previous timestamp.

4-2 IMU Pre-integration

The frequency of incoming IMU measurements can reach 200Hz and is often much higher than the frequency at which a camera collects data, which is often around 24Hz. The high IMU frequency may lead to problems regarding the computational capacity of the computing device. For all SLAM applications, it is beneficial to summarise the sequence of IMU messages in between two camera frames to reduce the computational load. This process is called IMU pre-integration and yields one single IMU observation that summarises the information about the movement of the IMU in between two sequential states. To understand this method, some preliminary knowledge regarding two Riemannian manifolds is needed: the Special Orthogonal Group $SO(3)$ and the Special Euclidean Group $SE(3)$. A manifold is mathematically defined a topological space, that locally approximates a Euclidean ($\mathbb{R}^n$) space [19]. An example is the

round Earth, which was unthinkable in early days because of the scale of the Earth, which gave the illusion of flatness.

The $SO(3)$ group is the group of 3×3, 3D rotation matrices. This group forms a smooth manifold. The tangent space to this manifold is known as the Lie Algebra, denoted as $\mathfrak{so}(3)$, which coincides with the group of 3×3 skew symmetric matrices. Every vector in $\mathbb{R}^3$ can be mapped to a skew symmetric matrix using the *hat* operator:

$$\omega^\wedge = \begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix}^\wedge = \begin{bmatrix}0 & -\omega_3 & \omega_2\\ \omega_3 & 0 & -\omega_1\\ -\omega_2 & \omega_1 & 0\end{bmatrix} \in \mathfrak{so}(3) \tag{4-3}$$

The reverse operator of the *hat* operator is the *vee* operator, which maps a skew symmetric matrix to a vector in $\mathbb{R}^3$. Now two other maps will be introduced: the *exponential map* and the *logarithm map*. The *exponential map* associates a skew symmetric matrix from the Lie Algebra to a rotation matrix in $SO(3)$, following Rodrigues' formula [20]:

$$\begin{aligned}\exp(\phi^\wedge) &= \mathbf{I} + \frac{\sin(|\phi|)}{|\phi|}\phi^\wedge + \frac{-\cos(|\phi|)}{|\phi|^2}(\phi^\wedge)^2\\ \exp(\phi^\wedge) &\approx \mathbf{I} + (\phi^\wedge)\end{aligned} \tag{4-4}$$

The *logarithm map* is the reversed operator of the *exponential map* and associates a matrix in $SO(3)$ with a skew symmetric matrix:

$$log(R) = \frac{\psi(R - R^T)}{2\sin(\psi)} \quad \text{with } \psi = \cos^{-1}\left(\frac{\mathrm{tr}(R) - 1}{2}\right) \tag{4-5}$$

In the following equations, a more convenient notation will be used in which Exp and Log operate on vectors rather than on elements from the Lie Algebra:

$$\begin{aligned}\mathrm{Exp}(\phi) &= \exp(\phi^\wedge)\\ \mathrm{Log}(\phi) &= \log(R)^\vee\end{aligned} \tag{4-6}$$

During the derivation of IMU pre-integration equations, the following first-order approximation will turn out very useful, which helps to relate a small additive perturbation $\delta\phi$ in the Lie Algebra to a multiplicative perturbation on the manifold SO(3), which is visualised in figure 4-2:

$$\mathrm{Exp}(\phi + \delta\phi) \approx \mathrm{Exp}(\phi)\mathrm{Exp}(J_r(\phi)\delta\phi) \tag{4-7}$$

Similarly, for the *logarithm map* the following approximation holds:

$$\mathrm{Log}\left(\mathrm{Exp}(\phi)\mathrm{Exp}(\delta\phi)\right) \approx \phi + J_r^{-1}(\phi)\delta\phi \tag{4-8}$$

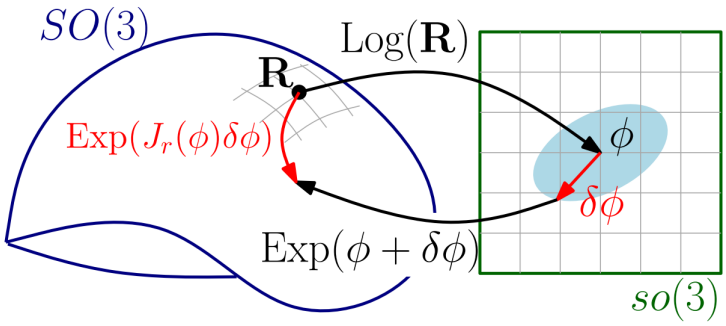


Figure 4-2: The exponential and logarithm map. The right Jacobian J_r is used to translate an additive perturbation $\delta\phi$ from $\mathfrak{so}(3)$ to the manifold $SO(3)$

In these equations, $J_r(\phi)$ and J_r^{-1} are the *right Jacobian* and its inverse of $SO(3)$ respectively. They are defined as:

$$\begin{aligned} J_r(\phi) &= \mathbf{I} - \frac{1-\cos(||\phi||)}{||\phi||^2}\phi^\wedge + \frac{||\phi|| - \sin(||\phi||)}{||\phi||^3}\left(\phi^\wedge\right)^2 \\ J_r^{-1}(\phi) &= \mathbf{I} + \frac{1}{2}\phi^\wedge + \left(\frac{1}{||\phi||^2} + \frac{1+\cos(||\phi||)}{2||\phi||\sin(|\phi|)}\right)\left(\phi^\wedge\right)^2 \end{aligned} \tag{4-9}$$

The following kinematic model is introduced from [21]:

$$\dot{R} = R\,\boldsymbol{\omega}^\wedge, \quad \dot{\mathbf{v}} = \mathbf{a}, \quad \dot{\mathbf{p}} = \mathbf{v} \tag{4-10}$$

By integrating the equations from this kinematic model, the state at time $t+\Delta t$ is obtained, which is purpose of IMU pre-integration. The method described above is utilised by the majority of visual-inertial SLAM systems that perform IMU pre-integration.

$$\begin{aligned} R(t+\Delta t) &= R(t)\mathrm{Exp}\left(\int_t^{t+\Delta T} \boldsymbol{\omega}(\tau)d\tau\right) \\ \mathbf{v}(t+\Delta t) &= \mathbf{v}(t) + \int_t^{t+\Delta t} \mathbf{a}(\tau)d\tau \\ \mathbf{p}(t+\Delta t) &= \mathbf{p}(t) + \int_t^{t+\Delta t} \mathbf{v}(\tau)d\tau + \iint_t^{t+\Delta t} \mathbf{a}(\tau)d\tau^2 \end{aligned} \tag{4-11}$$

4-3 RealSense Calibration

For any visual-inertial SLAM system that is designed for real-time missions, it is of great importance to determine the noise characteristics of the sensors. The hardware used in this thesis is a RealSense D435i, which carries multiple RGB and infrared camera's to provide RGB-D images to the on-board computer. The IMU incorporated in the RealSense is a BMI055. The calibration procedure consists of three main steps: calibration of the camera intrinsics, calibration of the IMU and finally a joint camera-IMU calibration.

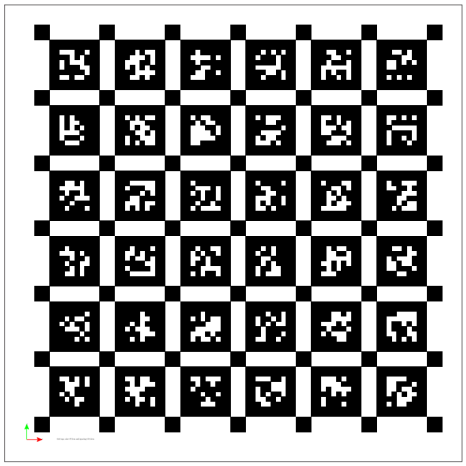


Figure 4-3: Calibration target utilised by the Kalibr tool

For the calibration of the intrinsic camera parameters, the open-source Kalibr[1] [22–26] toolkit was used. Using a calibration pattern of known dimensions, which is the input to the Kalibr tool, the intrinsic parameters of the camera can be determined. This calibration pattern is depicted in figure 4-3. Because the dimensions of the calibration target are known, the intrinsic camera parameters can be determined with great accuracy, which is needed for the camera-IMU extrinsic calibration.

Secondly, the noise parameters of the IMU inside the RealSense are retrieved using the open-source Allan Variance package[2]. This requires a ROS-bag containing IMU measurements from a long, static IMU session, to make sure that the noise characteristics of the sensor can be directly retrieved from the measured accelerations. In our case, a ROS-bag 13 hours was recorded. The resulting Allan Deviation (AD) plots are given below:

Parameter	Unit	Value
Accelerometer noise density	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$	0.009972179
Accelerometer random walk	$\frac{m}{s^3}\frac{1}{\sqrt{Hz}}$	0.0015407493
Gyroscope noise density	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$	0.00639172965
Gyroscope random walk	$\frac{rad}{s^2}\frac{1}{\sqrt{Hz}}$	0.0000847221059

Table 4-1: IMU noise model parameters

[1] https://github.com/ethz-asl/kalibr
[2] https://github.com/ori-drs/allan_variance_ros

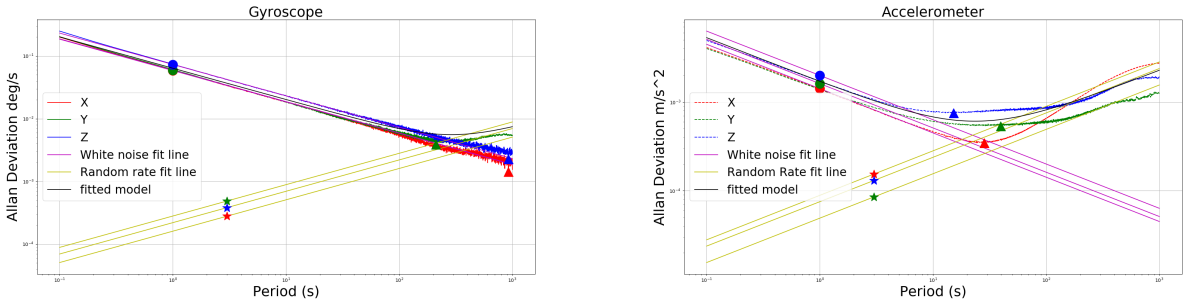


Figure 4-4: Allan Deviation plots for IMU accelerometers and gyroscopes

The third calibration is the camera-IMU calibration, for which again the Kalibr tool was used. The results of this calibration are included in Appendix A.

Chapter 5

Methodology

As introduced in chapter 3, the design objective for this thesis to establish a more robust and failure-resilient SLAM pipeline, through the incorporation of IMU measurements into the solution. As this idea is not necessarily unique, several frameworks and solvers have been published in literature. To justify to choice for the chosen strategy of building a non-linear factor graph, several other frameworks are discussed below.

Popular visual-inertial SLAM frameworks are VINS-Mono [27] and ORB-SLAM3 [28]. Although these methods both demonstrate impressive results regarding accuracy and efficiency, they are very rigid in a sense that they have little room for adjustments, if needed. RTAB-map on the other hand is a much more flexible framework that adapts to incorporate a wide range of combinations of different sensors, such as RGB-D, stereo or monocular camera setups, as well as inertial information and even LiDAR measurements. As it is designed to be modular, it is a suitable framework for experimenting with various sensor setups. A major advantage is that it solves the SLAM problem using the factor graph strategy.

Regarding the choice for the solver, options such as the Ceres [29] and G2O [30] exist. While these are very powerful tools to solve large optimisation problems, GTSAM (Georgia Tech Smoothing and Mapping) [31] distinguishes itself because it is tailored towards SLAM applications. It is a C++ library that incrementally solves the SLAM problem, which yields a high computational efficiency through re-using previous calculations as explained in chapter 2.

Section 5-1 elaborates on the design of the non-linear factor graph. Section 5-2 discusses which parameters play a significant role during the optimisation. Section 5-3 elaborates on the general layout of the novel software infrastructure and section 5-4 discusses the experimental data that was gathered in order to demonstrate the behaviour of the novel visual inertial SLAM algorithm.

5-1 Nonlinear Factor Graph

As explained in section 2-6, the factor graph consists of *values* that are connected by *factors*. Note that a *value* is a vector of variable size, corresponding to the dimension of the optimised variable. The values in the graph are the unknown solution variables which are optimised for. In the implementation of this thesis, three different types of values exist: pose x, velocity y and bias b. The factors connect the values in the graph and contain information regarding the relationships between the individual values. The design of this graph is inspired on the graph presented in [32]:

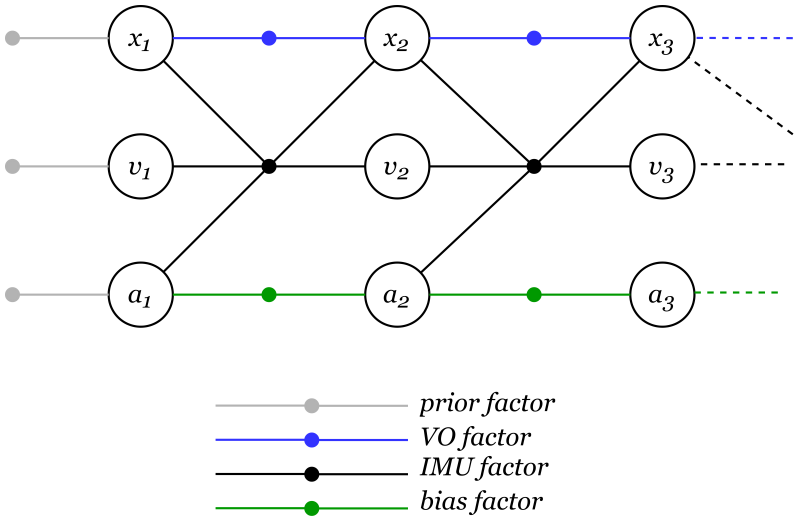


Figure 5-1: Schematic overview of the constructed factor graph

The factors are all associated with a noise model represented by a covariance matrix, the size of which depends on the dimensions of the value. This noise model captures the probabilistic information about the error in the measurements. This directly reflects on the level of confidence the optimiser will have in the data summarised by the corresponding factor.

The visual odometry factor is the most straightforward factor in the graph, providing information on how the rover moved from the previous pose x_{t-1} to the current pose x_t, *according to the visual odometry module*. As the pose of the rover is a 6-dimensional vector, the noise model is a 6 by 6 diagonal matrix, as shown in equation 5-1:

$$\eta_{VO} = \begin{bmatrix} \left[\sigma_{rot_{VO}}\right]_{3\times3} & 0 \\ 0 & \left[\sigma_{trans_{VO}}\right]_{3\times3} \end{bmatrix} \tag{5-1}$$

The values for these parameters $\sigma_{rot_{VO}}$ and $\sigma_{trans_{VO}}$ can be set manually, which will be elaborated on in section 5-2.

The second factor type in the factor graph is the IMU factor. The GTSAM-library is leveraged to perform IMU measurement pre-integration. This generates a single pre-integrated IMU measurement which contains the information how the rover has moved *according to the IMU*. The IMU factor is a five-ways factor connecting the previous pose, velocity and bias with

the current pose and velocity. The covariance matrix on the IMU factors is a non-diagonal 9 by 9 matrix, calculated by the GTSAM toolbox, which describes the covariance between pre-integrated rotation, position and velocity. This noise model can be influenced however on the diagonal by setting the values for the covariances $\sigma_{rot_{IMU}}$, $\sigma_{trans_{IMU}}$ and $\sigma_{vel_{IMU}}$. This factor is then incorporated into the factor graph as well.

$$\eta_{IMU} = \begin{bmatrix} \left[\sigma_{rot_{IMU}}\right]_{3\times3} & & \\ & \left[\sigma_{trans_{IMU}}\right]_{3\times3} & \\ & & \left[\sigma_{vel_{IMU}}\right]_{3\times3} \end{bmatrix} \tag{5-2}$$

The third factor type is the bias factor. This bias factor is a two-way factor connecting the previous and the current bias. When creating the bias factor, we set the inter-bias change to zero, implementing the suggestion that the bias does not change. A crucial principle of the optimisation scheme is that the optimiser however will apply some Δ_b between two consecutive biases, íf this minimises the residual optimisation error by reducing the difference between VO- and IMU-factors. This will result in a slowly varying bias over time, which is exactly the goal. By varying the covariance on the bias constraint, which is a 6 by 6 diagonal matrix as shown in equation, the rate of change can be controlled: a low covariance on the constant bias yields a very slowly varying bias, as the constraint indicates that the bias should be kept constant, and vice versa. Recall that the 'random walk' of the biases is sensor specific. The values for this bias stability are determined by the Allen Deviation plots, as explained in chapter 4.

$$\eta = \begin{bmatrix} \left[\sigma_{b_a}\right]_{3\times3} & 0 \\ 0 & \left[\sigma_{b_g}\right]_{3\times3} \end{bmatrix} \tag{5-3}$$

Parameter	Symbol	Default value
Variance on prior pose	$\sigma_{p_{pos}}$	0.001
Variance on prior velocity	$\sigma_{p_{vel}}$	0.001
Variance on prior bias	$\sigma_{p_{bias}}$	0.001
Accelerometer noise density	σ_a	0.00997
Gyroscope noise density	σ_g	0.00639
Integration uncertainty	σ_i	0.001
Accelerometer random walk	σ_{b_a}	0.001541
Gyroscope random walk	σ_{b_g}	0.00008472
Visual odometry rotational variance	$\sigma_{rot_{VO}}$	0.00030625
Visual odometry translational variance	$\sigma_{trans_{VO}}$	0.0001
Loop closure rotational variance	$\sigma_{rot_{LC}}$	0.00030625
Loop closure translational variance	$\sigma_{trans_{LC}}$	0.0001

Table 5-1: Optimisation parameters with default values

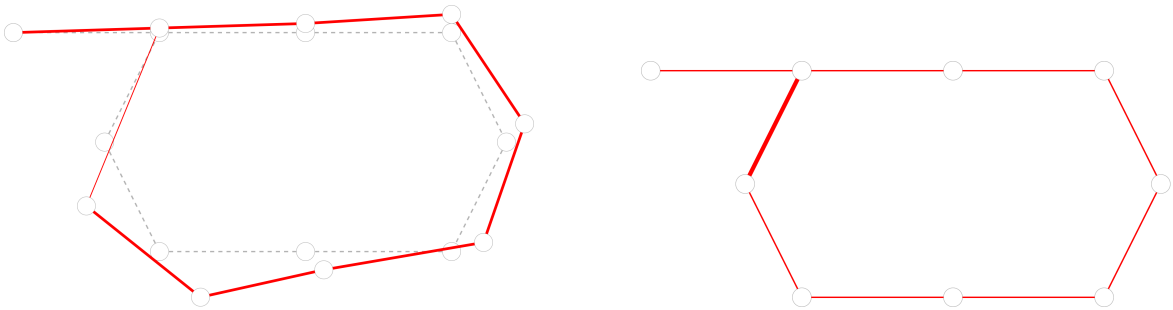
5-2 Optimisation Parameters

As mentioned before, the noise models on the individual factors determine how much the optimiser relies on the information provided by that specific factor. Hence, the behaviour of the final solution is influenced by the values of these parameters. An overview of the most important parameters are given in table 5-1. How these parameters affect the solution is discussed below.

(a) Low IMU covariance **(b)** Low VO covariance

The estimated motion represented by the VO and IMU factors is never perfectly the same due to error and noise in the measurement models of these sensors. To unify these motion constraints, the optimiser takes the noise models on both factor types into consideration, resulting with a weighted estimate somewhere in between these two factors. If a factor has a noise model with low covariances, deviation from this factor will contribute a lot to the residual optimisation cost. Hence, the optimiser prefers not to deviate as much from low-covariance factors rather than factors with a higher covariance, which contribute less to the optimisation cost. The resulting effect is that factors with a lower covariance noise model are trusted more. Figure 5-2a shows an example for an optimiser with low covariance in the IMU noise model, trusting the IMU factors more. Figure 5-2b shows the opposite effect.

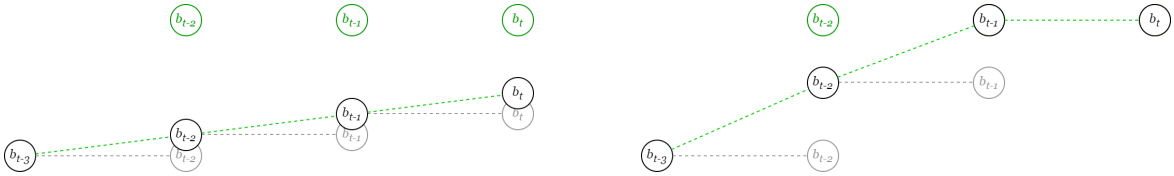
A similar strategy is used for loop closure constraints, as illustrated in the figures 5-3a and 5-3b below. It might seem tempting to set a very low covariance for the noise model on the loop closure constraints. Recall however that the mechanism behind a loop closure is exactly the same as regular visual odometry: from two images, features are extracted and used for in a bundle adjustment scheme to determine the motion of the camera from one image to the other. The only actual difference between visual odometry constraints and loop closure



(a) High loop closure covariance **(b)** Low loop closure covariance

constraints is that one image is the direct successor of the other one, while in detecting a loop closure constraint any two images are compared. For this reason, the noise models for loop closure constraints and visual odometry factors are equal.

Other important parameter that affect the solution behaviour are related to the IMU bias estimation. As explained in section 4-1, these biases should be modelled as Brownian motion, meaning that they randomly "walk" up or down. This is achieved through constraining two consecutive bias values to be constant. The optimiser will definitely deviate from this constant constraint, if it reduces the error between pre-integrated IMU poses and VO poses. The covariance on the bias constraints determines how much the optimiser is allowed to deviate from keeping the biases constant. The figures 5-4a and 5-4b below visualise this:



(a) Low bias constraint covariance **(b)** High bias constraint covariance

In these figures, the grey bias values represent the constraint, i.e. keeping the bias constant over time. The green values represent the ideal bias, i.e. the bias that if corrected for in the IMU measurements would lead to the perfect fit between the visual odometry and inertial navigation. The black values are the final estimated bias values, balancing the increase of the optimisation cost due to the deviation from constant-bias-constraint and the optimisation cost due to the VO-IMU discrepancy.

The last parameters to be discussed are the noise models on the priors for rover pose, velocity and initial bias. The prior on rover pose determines where in the global space the first pose is located. For a new SLAM session, this prior is set to zero. If the rover starts in a stationary position (as is always the case in our experiment), the prior on the velocity is also set to zero. The prior on the biases is not so easily determined, since we had to experiment with the initialisation of the estimated IMU biases.

Figure 5-5 illustrates that if the RealSense device is kept perfectly level, the gravity vector coincides perfectly with the y-axis of the IMU. Important to note here is that the RealSense device does not measure forces, but accelerations: since the gravity force results in the same forces on the IMU as an upwards acceleration of $9.81 m/s^2$, in the opposite direction of the IMU y-axis, the IMU would register gravity as an acceleration in negative y-direction. During

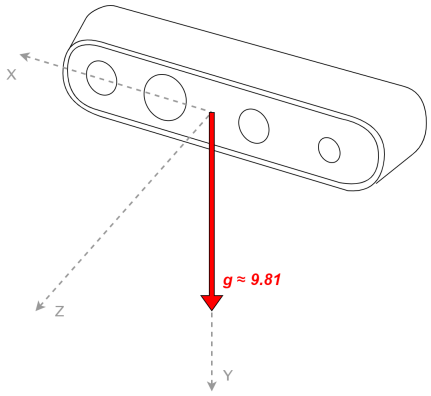


Figure 5-5: Gravity vector coinciding with IMU y-axis in level orientation

the construction of the GTSAM pre-integration pipeline, the direction of the gravity vector is needed, to enable GTSAM to distinguish the actual accelerations from the gravity-induced forces in the raw IMU measurements. As seen in the raw IMU measurements in figure 5-12, during the stationary period in the beginning of the recording the gravity vector is reflected on all three IMU axis, which corresponds to the fact that the RealSense was slightly tilted to the back (positive rotation around x-axis) and to the side (negative rotation around y-axis).

Although GTSAM maintains and updates this gravity direction according to the measured angular velocities, no stable solutions could be obtained. Upon inspection of the estimated IMU biases, it was clear that the solver could not correctly estimate these biases. An error in the bias estimation inevitably results in IMU drift. Running different datasets all showed the same instability in bias estimation. In the subsequent attempts, the dataset's first 20 seconds were cut to ensure that at the start of the recording the IMU is in the upright position, but to no avail.

A creative solution was found which confirms the understanding into the mechanism of bias estimation, but it requires the RealSense to maintain an upright orientation throughout the whole experiment. Since the gravity vector has to be removed from the raw IMU biases to obtain the actual IMU accelerations, we decided to set a gravity vector of $[0, 0, 0]^T$. As the optimiser now registers a high acceleration on the y-axis whilst observing from visual measurements that no acceleration in this direction takes place, the gravity vector will be processed as acceleration bias. In this way, the algorithm is still able to distillate the actual accelerations from the raw IMU measurements, which is ultimately the goal of removing the gravity component from the measurements. The estimated IMU biases showed much more plausible results now, which are discussed in chapter 6.

5-3 Software Architecture

The RTAB-Map software provides a convenient rtabmap_ros wrapper for usage with ROS (Robot Operating Software). Figure 5-6 gives an overview of the ROS infrastructure. The SLAM pipeline relies on the interaction between two ROS nodes: *rgbd_odometry* and *rtabmap*. The *rgbd_odometry* node is responsible for collecting the visual measurements. Subsequently, it calculates the rover's motions using visual odometry, based on the sequence of depth images from the RealSense. At a frequency of 1Hz, these visual odometry constraints, as well as the cached raw IMU measurements received since in the last second, are communicated to the *rtabmap* node. This node is the core of the SLAM solution, implementing the optimisation strategy of building a non-linear factor graph and solving it.

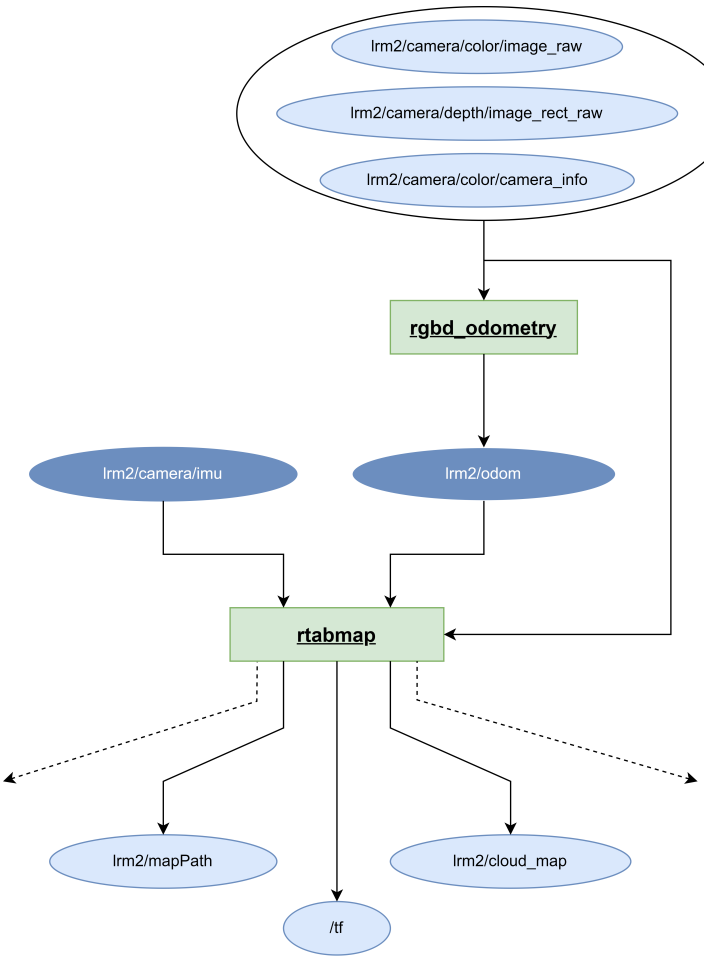


Figure 5-6: ROS architecture

5-3-1 ROS Node: *rgbd_odometry*

The *rgbd_odometry* node calculates the rover's motions based on the sequence of received images. Upon initialisation, the node creates a global coordinate frame *'odom'* at the same position and orientation as the rover's own body coordinate frame, *base_link*. This is now the global reference frame with respect to which all future robot poses will be calculated. By subscribing to the ROS-topic *camera/color/image_raw*, the latest RGB-D image is fetched. A GFTT-feature (Good Features to Track) [33] detector extracts keypoints from this RGB-D image, as visualised in figure 5-7. The local environment around the keypoints are described using BRIEF-descriptors [34]. Using this description, a feature can be across two different images. This is how a single feature can be matched in multiple images, which is detrimental to perform visual odometry.

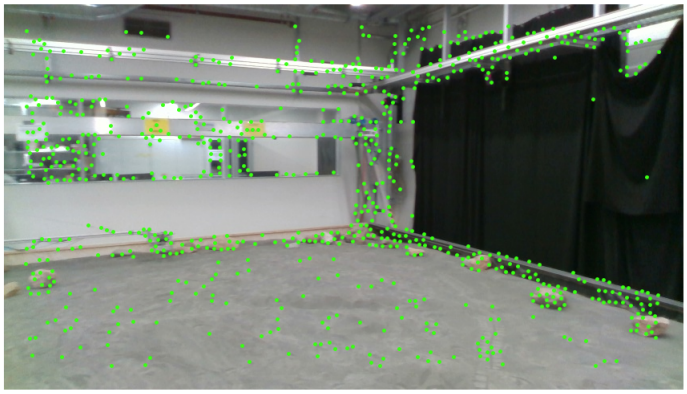
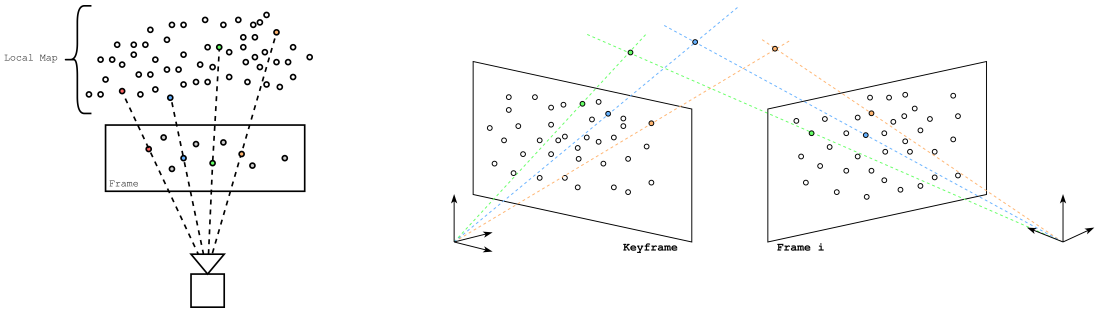


Figure 5-7: Extracted features from a scene in the experiment

In the previous implementation of the SLAM algorithm, a Frame-to-Map (F2M) strategy is used to perform visual odometry, as visualised in figure 5-8b. It keeps track of a small pointcloud of 3D features, consisting of the 3D keypoints for the last couple of depth images. Based on this local map and the 2D keypoints in the image, 3D-2D image registration is performed using a PnP (Perspective-n-Point) approach, which returns the estimated motion of the camera since the previous image. This strategy however does not work in the event of a loss of visual odometry, which is exactly the reason why this novel SLAM algorithm is needed. The reason why it does not work is that if the visual odometry fails due to a lack of recognised features, there will be no new features added to the local map. When visual odometry is regained after a while and new features are recognised, none of these new features will have correspondences to the local map because the new features are all yet unobserved. Since the 3D-2D image registration relies on these correspondences, the visual odometry pipeline will fail to extract the rover's motions, even when located in a feature-rich area.

To solve this issue, the Frame-to-Frame (F2F) strategy is adopted. In this solution, instead of performing 3D-2D image registration, 2D-2D registration is performed. The latest image frame is compared to the last saved *keyframe*. Through bundle adjustment, the motion between the two frames can be estimated. By constantly setting the keyframe to the second-to-last image, the visual odometry node is able to still perform its task, even after it has gone



(a) Frame-to-Map odometry **(b)** Frame-to-Frame odometry

Figure 5-8: Difference between F2M and F2F strategy

through a feature-poor and a new feature-rich area. Although it is unknown how these new odometry constraints are connected to the rest of the map, they can be connected to the global map using the found IMU factors during the period of VO-absence.

5-3-2 ROS Node: *rtabmap*

The second running node is the *rtabmap* node, which's core responsibility is the initialisation, expanding and solving of the non-linear factor graph. Upon initialisation, the *NonlinearFactorGraph* object is created. The first three values for pose, velocity and bias are created and inserted into the factor graph, along with the corresponding prior values.

After initialisation, the same two-step cycle of creating and appending VO- and IMU-factors to the factor graph is executed. The first step is to pre-integrate the batch of approximately 200 IMU-measurements. The GTSAM toolbox includes the software logic to perform this pre-integration, and returns the fresh IMU factor. For the VO-factor, the relative motion estimate between the previous and the current pose is needed. This is obtained by fetching the previous and current *unoptimised* poses, as they are provided by the *rgbd_odometry* node. The mathematics behind this operation are described by equation 5-4 and figure 5-9:

$$\begin{aligned} p_2^G &= v_{12} * p_1^G \\ v_{12} &= p_2^G (p_1^G)^{-1} \end{aligned} \tag{5-4}$$

Not only does the *rtabmap* node receive unoptimised rover poses, the *rgbd_odometry* node also keeps track of a quality score for the odometry. This score is determined by the amount of matched features in the PnP registration. If this value drops below a certain threshold, the feature extraction works poorly and switching to pure inertial odometry should be considered. This score is a good measure of how accurate a motion constraint is, and thus how much the optimiser can rely on it. By dynamically setting and changing the values of the covariances in the noise model, this layer of awareness can be incorporated in the factor graph.

The expected behaviour of the algorithm in a scenario of complete loss of visual odometry is that the estimated biases will not change during the period where only inertial measurements are received by *rtabmap*. This makes sense since there is no visual evidence that can be used

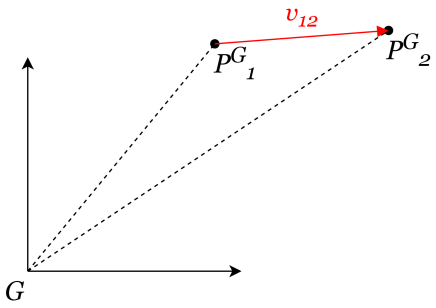


Figure 5-9: Derivation of relative transform based on two consecutive global poses

to update the biases. However, when the VO module is working again, the old, constant bias values are automatically updated by GTSAM, to minimise the residual error. In theory, this should even further reduce the the buildup of error due to the constant biases, which are in reality never constant.

Figure 5-10: Test environment at DLR-RMC

5-4 Experimental Data

The visual-inertial algorithm was tested in a realistic setting in the testbed at the DLR-RMC institute, as seen in figure 5-10. This testbed is a reconstructed Moon environment, including the fine soil and recognisable rocks. The main experiment that was conducted was a trajectory in the shape of an '8', hence the session name 'Figure_8'. During the test, the trajectory was repeatedly travelled in order to enforce some detected loop closures. Necessary data to perform SLAM was recorded using a ROS-bag to replay the experiment afterwards. The recorded ROS-topics are:

```
/lrm2/camera/color/image_raw
/lrm2/camera/color/camera_info
/lrm2/camera/depth/image_rect_raw
/lrm2/camera/imu
/lrm2/odom
/tf
```

The majority of developments was done on the *rtabmap* node. To work efficiently, the function calls to the main processor in *rtabmap* were stored in text- and imagefiles. The content of these files was then directly parsed to *rtabmap*, eliminating the need to constantly replay ROS-bags, which would occupy much more time as they are replayed at real-time speed.

To evaluate the performance and accuracy of the visual-inertial odometry, some knowledge about the true trajectory is needed. This ground truth data can be acquired by tracking the rover from an external reference frame, since it directly measures robot's position and hence does not accumulate errors which leads to drift. For this task, the Advanced Realtime Tracking (ART) system is used, available at the DLR-RMC institute. This system is installed on the ceiling of the testbed. The ART system consists of 10 infrared camera's tracking

the position of silver-coloured reflective markers. By observing the position of a marker by multiple camera's, the absolute position of the tracker can be accurately calculated through triangulation. Attaching these trackers to the rover then gives a good ground truth trajectory to evaluate the performance of new SLAM solutions. The ground truth trajectory is visualised in figure 5-11.

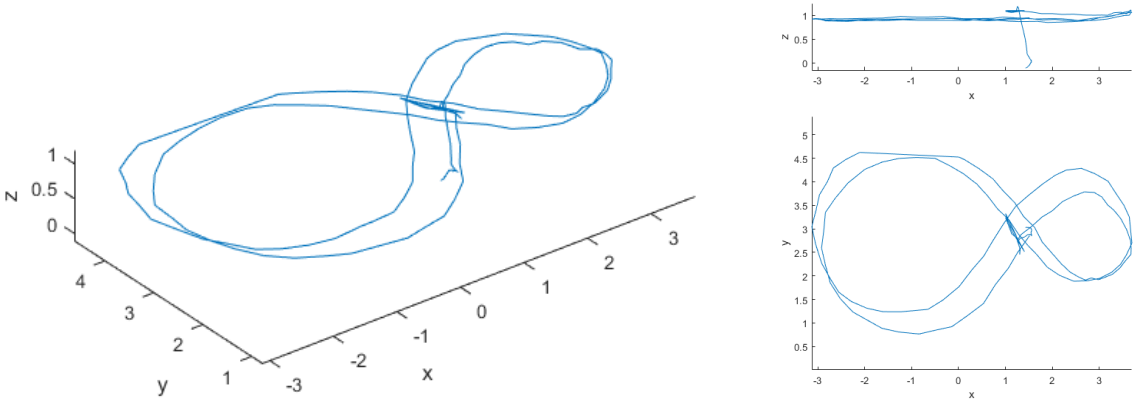


Figure 5-11: The ground truth trajectory measured by the ART tracking system

The raw IMU measurements from this experiment are plotted in figure 5-12. This plot shows the starting position in rest, in which only the gravity vector leads to a measured force.

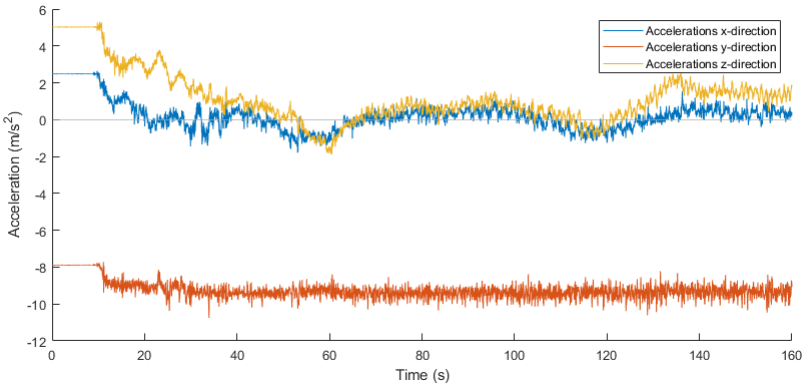


Figure 5-12: Raw IMU measurements during first 160 seconds of the experiment

Chapter 6

Results

Generally, in order to be able to calculate any metric on two different trajectories and evaluate the performance of a novel odometry system, these trajectories have to be expressed in the same coordinate frame. Since the visual-inertial estimate is generated in the camera frame of the first pose and the ground truth is generated in a global coordinate frame, it is necessary to align the trajectories of the estimate and the ground truth. Although the orientation of the marker in the global frame easily can be extracted from the tracking data, alignment is not possible since there is no accurate transformation from the tracker itself to the camera frame, as the tracker was taped to the camera during the experiments. Therefore, direct alignment through a concatenation of known transformations is not possible. For uncalibrated experiments such as the one performed in this thesis, the trajectories have to manually aligned for performance evaluation. This is done using Horn's method [35]:

$$\mathbf{R}, \mathbf{t} = \underset{\mathbf{R},\mathbf{t}}{argmin} \sum_{i=1}^{N} \|(\mathbf{R} * p_{slam,i} + \mathbf{t}) - p_{gt,i}\|^2 \tag{6-1}$$

Horn's method finds the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ to manipulate the estimated trajectory, which minimises the residual error. The downside however is that the absolute drift can not be estimated using this method, as part of the drift in a trajectory will be compensated for during an alignment based on Horn's method. Nevertheless, an estimator with low drift will generate a trajectory that more closely fits the ground truth than an estimator with high drift does. Therefore, calculating the deviation of each pose from its corresponding pose in the ground truth will provide useful information about the accuracy of the algorithm. The Root Mean Squared Error is a a metric that does this, which is defined by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|p_{slam,i} - p_{gt,i}\|^2}{N}} \tag{6-2}$$

6-1 Main Experiment

To evaluate the behaviour of the novel visual-inertial odometry, and especially the resiliency towards a camera failure, a scenario was simulated based on the gathered experimental data. As explained in the problem statement in chapter 3, a (short) period of absence of camera measurements causes the mapping and localisation activities to be disrupted. By manually disabling the visual odometry module, such a scenario can be simulated. This was done at specific moments during the simulation, as visualised in figure 6-1 and explained below:

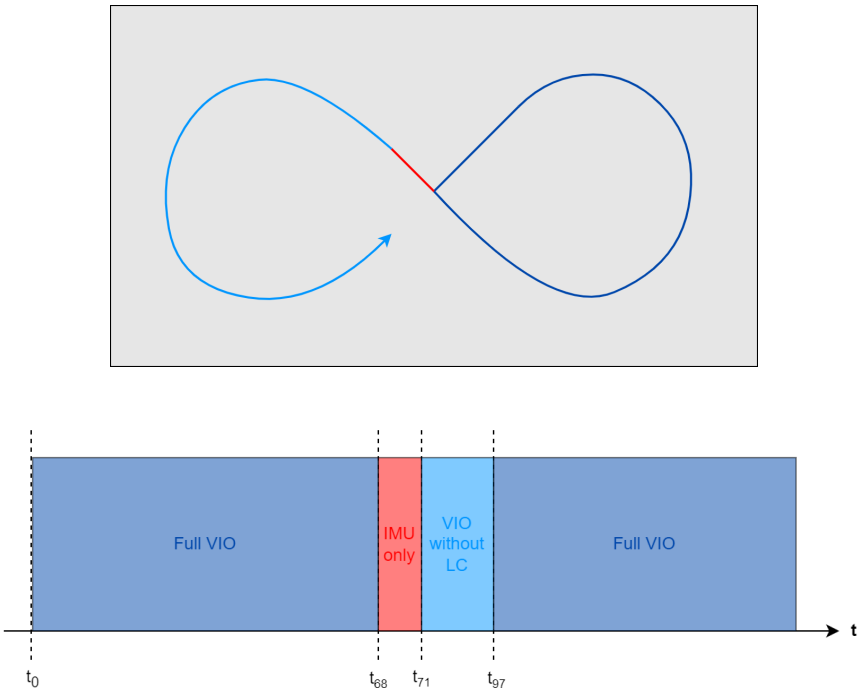


Figure 6-1: Setup for the testcase

From the start until timestamp 68, the full visual-inertial system with all necessary sensors are functional. This period is needed to allow the optimiser to correctly estimate the values for the IMU biases. At timestamp 68 the visual odometry pipeline is manually disabled, to simulate the event of loss of feature detection. At t=71, the visual odometry module is providing VO-constraints again. In theory, the full system is now operational, and also the loop closure detector is enabled. However, as this is an area that has not been visited before, no loop closures will be detected. To clarify this in the results, the trajectory up to pose 97 (when the first loop closure will be detected) is coloured in a lighter shade of blue, to illustrate that this part of the trajectory consists of visual-inertial odometry without the correction of a loop closure.

This section describes the test results of the main experiment as explained above. In figure 6-2, the ground truth data is plotted with a grey dashed line, and the estimated trajectory is plotted in various colours corresponding to the experiment setup as described above.

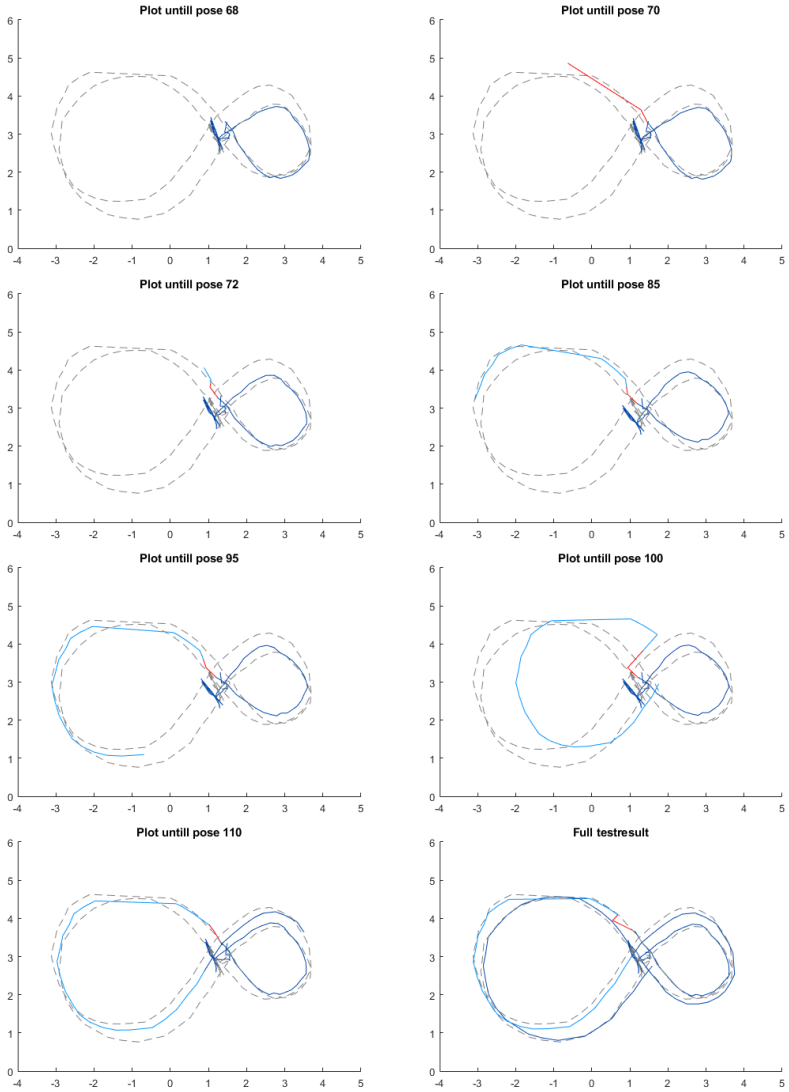


Figure 6-2: Results

The calculated RMSE for this testcase is 0.1361 meter. In the first semi-loop, both the visual odometry module and the IMU are providing motion constraints, establishing plain visual-inertial odometry. At pose 68, the visual odometry is manually disabled, which leaves the algorithm with just inertial measurements to navigate on. As visible, the estimated location of the rover quickly drifts off. However, as soon as the visual odometry module is functional again at pose 72, it is clear that also the previous poses are updated. This is fully in accordance to the expected behaviour, as explained in section 5-3.

In the subsequent poses, from pose 72 until 95, it is clear that the visual odometry module is working as desired again, since the shape of the light-blue trajectory seems to match the

ground truth. At pose 100, something remarkable takes place: the light-blue loop seems to have shifted away from the ground truth. This is caused by an detected loop closure between poses 97 and 35. In this scenario, the covariance on IMU factors was set relatively high, meaning that we allow the optimiser to deviate from the IMU constraints relatively much. Any error in the detected loop closure at the end of the light-blue loop, will be reflected on all poses in this loop, with an increasing effect as the distance to these poses increases. The problem however solves itself when additional loop closures are detected, correcting for the single erroneous loop closure at pose 97. Section 6-3 goes into more detail regarding this behaviour.

The major contribution of this thesis to the LRM is the resiliency to the failing visual odometry module. In figure 6-2, the third figure on the left concerning the trajectory until pose 95, shows that after a disruption of visual measurements the light-blue trajectory can be inserted with decent accuracy into the complete trajectory, without the need to perform relocalisation. In the old visual SLAM algorithm, none of the visual measurements in this light-blue trajectory could have been appended to the global map. Therefore, the novel visual-inertial SLAM solution offers a higher level of robustness to generate more complete maps of the environment.

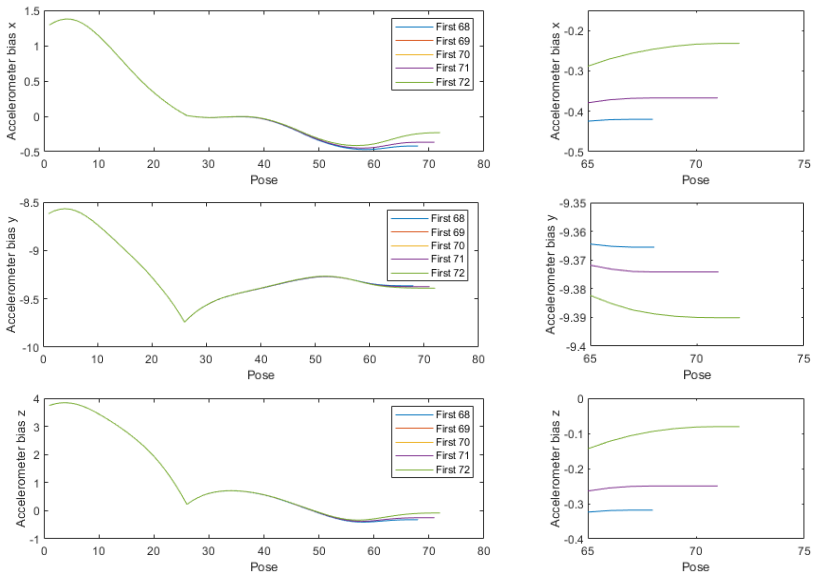


Figure 6-3: Bias estimated over time

Using to the graphs in figure 6-3, the behaviour of the bias estimation is discussed. The sharp point in the graphs of the y- and z-axis happening at pose 25 are caused by the prior bias value set to constrain the biases towards the expected values of [0, -9.81, 0], as explained in chapter 5. The most interesting result can be observed in the three detailed figures on the right. These plots show that the estimated biases do not change (as expected) in the period between t-68 and 71, where visual information is lacking. The more exciting observation is that at pose 72, as soon as the visual odometry works again, the optimiser realises that there is a new optimal bias estimate. Hence, all of the previous biases are updated, which influences the pre-integrated IMU measurements, which in its turn updates the red trajectory in figure 6-2.

Since the goal for the DLR is to have a working algorithm able to execute the novel visual-inertial SLAM algorithm on the LRM during real-time missions, it is important to investigate the computational effort of the novel algorithm. In the current implementation, the non-linear factor graph is updated at a frequency of 1 Hz. Logically, the optimiser then has 1 second to solve the problem, before the new measurements are received. The graph in figure 6-4 illustrates that the computational time slowly increases as the size of the nonlinear factor graph increases over time, but that the computational time per round is still well below 1 second. The spikes in this graph are caused by detected *potential* loop closures: in order to confirm one, the current image is compared to a range of potential matches. The amount of potential matches widely varies and is based on the amount of previously visited states in the vicinity of the current pose, that are at a distance within a certain threshold. In the case that the time-limit of 1 second is neared, due to the ever growing size of the problem, several actions can be undertaken to reduce the computational load. For starters, to reduce the height of the spikes, reduce the threshold for loop closure hypotheses, to reduce the amount of comparisons to previous states that have to be done. If the actual computational time of solving the non-linear factor graph becomes too long, a solution is to incorporate a sliding window approach, that optimises only the last subset of states, rather than the full problem. As the factor graph grows, it is very likely that the first couple of states and the information contained within the corresponding factors, does not change anymore. Removing them from the optimised window logically reduces the computational load.

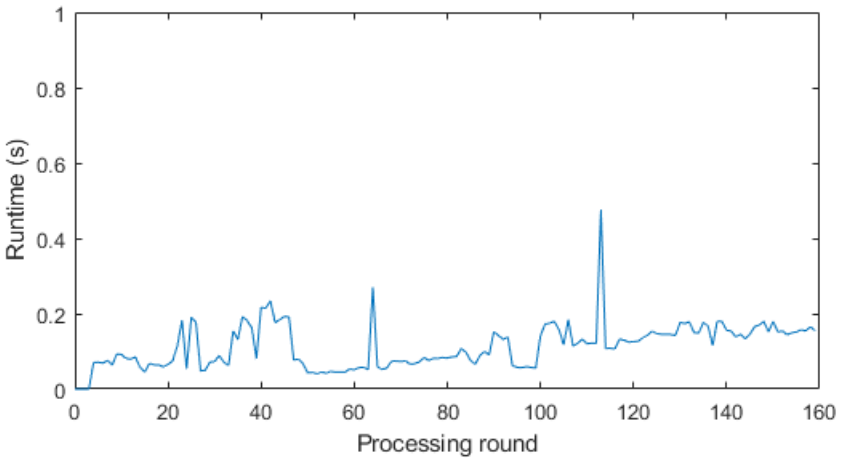


Figure 6-4: Duration of the computations for every optimisation round

6-2 Increased duration of VO absence

The main experiment illustrated that the novel visual-inertial SLAM algorithm is able to handle a complete absence of visual measurements. This was demonstrated on a relatively short period of 3 seconds. In the following experiment the duration of absence of visual measurements is increased to 8 seconds, in order to determine the boundaries of this novel algorithm's potential.

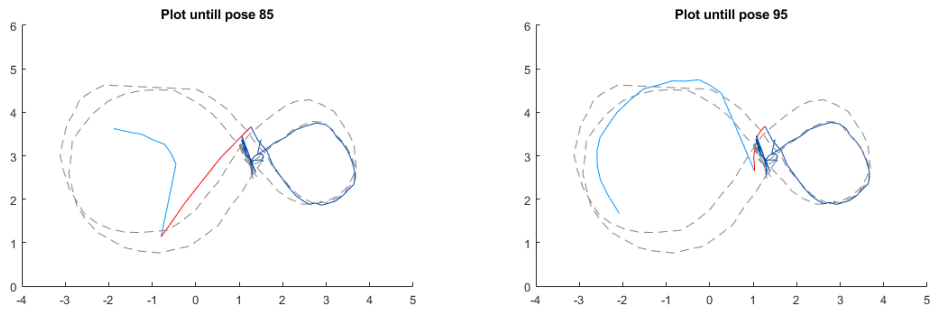


Figure 6-5: IMU drift for longer period of VO absence

The graphs in figure 6-5 show that the IMU drift is substantial, at least enough to completely displace the subsequent light-blue trajectory from the ground truth. Similarly to the main experiment, in the right figure is observable that over time this drift decreases as the biases are re-optimised at every computation. In figure 6-6 is illustrated that the loop closure between pose 97 and 35 again very heavily translates the complete light-blue trajectory to a wrong location. However, as more loop closures are detected afterwards, this problematic effect is minimised. Eventually, the final trajectory estimate is still quite accurate with an RMSE of 0.2635 meter.

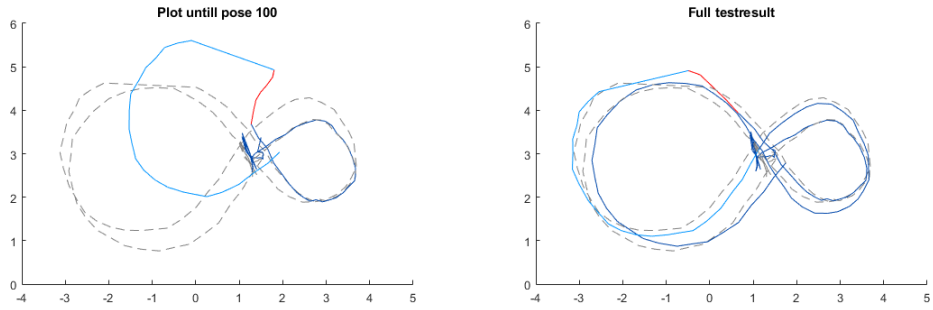


Figure 6-6: IMU drift for longer period of VO absence

The observable difference from this result in comparison to the results discussed in the previous section, is that after the visual odometry is regained (at the beginning of the light-blue trajectory), the red trajectory is not completely corrected. As the duration of pure inertial navigation was much longer in the second experiment, the damage done to the estimated

trajectory due to the inability to update the estimated biases is larger. An important insight from this experiment however is that through setting a high enough covariance on IMU factors, the problematic IMU poses in the red trajectory can be fully accounted for by loop closure constraints that would be detected in the future. This is yet another key takeaway from the work done in this thesis: through varying the covariances in the noise models on the individual factors, the optimiser can incorporate a level of awareness regarding how certain it is of specific areas in a map. The novel visual-inertial SLAM algorithm very conveniently allows for updating an area of the map which the rover is uncertain about.

During a normal mission it is of course impossible to know how long the period of pure inertial navigation will be. As an overall strategy, it seems to be wise to generate the factor graph with high variances on IMU factors and low variances on VO and LC factors.

6-3 Experimental Testcases

In the previous section, the results of the main testcase are discussed. During the execution of this experiment, the default/optimal values for all parameters are used, as listed in table 5-1. The following section illustrates how varying the parameter values affects the final solution, demonstrating the importance of setting the correct values for the optimisation parameters.

Figure 6-7 shows the estimated trajectory for the same experiment, with a much higher covariance on the loop closure constraints. In comparison to figure 6-2, it is clear that at pose 100 the loop closure constraint (97-35) now has a much smaller impact on the light-blue trajectory. Although this initially seems positive for the trajectory accuracy, especially since a erroneous loop closure constraint potentially ruins the accuracy of map sections, it soon becomes evident that the trajectory on the right side of the figure 8 deviates a lot from the ground truth. This demonstrates the power of these loop closure constraints and why it is typically not beneficial to set a high variance on these constraints, which lowers the trust the optimiser has in these constraints. With an RMSE of 0.2460 meter, this strategy clearly performs worse than a strategy where loop closure constraints are weighed more heavily in the optimisation.

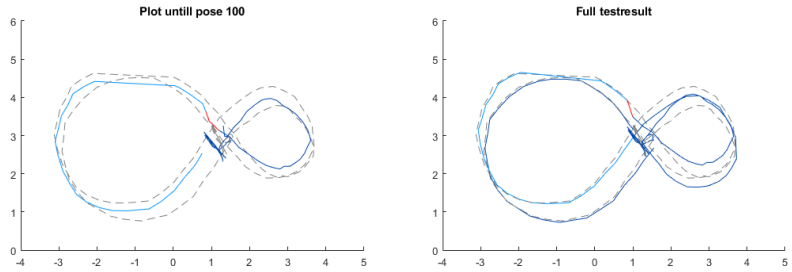


Figure 6-7: Results with a higher covariance on loop closures

Setting a very low covariance on these loop closure constraints is sub-optimal as well, as shown in figure 6-8. Similar to the results of the main experiment, one can see that the first loop closure on pose 97 has a detrimental effect on the light-blue trajectory. This makes sense as the covariance in the noise model are even lower, making the optimiser trust this single loop closure constraint even more. Similarly to the main experiment, this error is compensated for later in the trajectory. However, the calculated RMSE = 0.5264 meter, which is definitely higher than in the main experiment. The most probable explanation for this behaviour is the fact that VO-factors and the loop-closure-factors are computed using the exact same mathematical algorithm, being 2D-2D image registration. The noise models on them should therefore be equal, as it is the same measurement model estimating a certain value. If the noise models are not equal, as is the case in this example, it shifts the estimation into one direction, which clearly does not benefit the final solution.

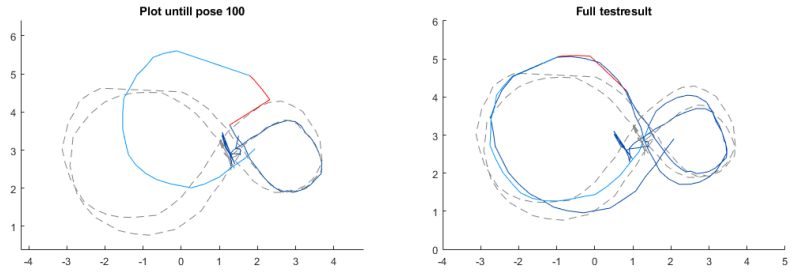


Figure 6-8: Results with a lower covariance on loop closures

Other parameters that should influence the final solution are the covariances on the noise models of the IMU-factors and the VO-factors, or at least the ratio between these two. The theoretical influence on the estimate is already illustrated in section 5-2. In this experiment however, there was no clear difference between the two parameter setups visible. This is caused by the loop closure constraints, which is yet another indication of how powerful these loop closures are. Imagine the IMU factors tend to drift in one direction, and VO factors drift in the opposite direction. It does not matter whether the optimiser relies more on IMU measurements or visual measurements, since a loop closure will always eliminate this drift. This idea is illustrated in figure 6-9:

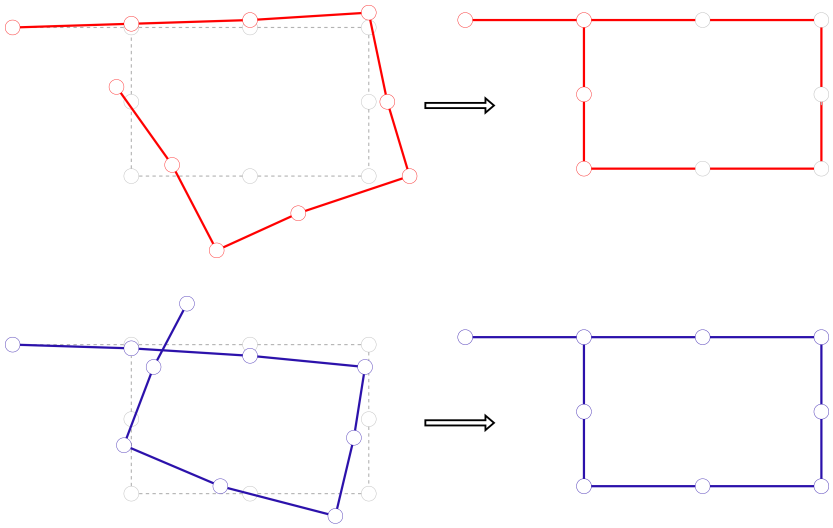


Figure 6-9: Regardless of drift direction, a loop closure will yeild similar trajectories

6-4 Improvements on accuracy

The novel visual-inertial SLAM algorithm has demonstrated an increased robustness, as it is able to handle a temporary loss of visual measurements, temporarily relying on pure inertial navigation. An additional potential benefit of fusing two sensors is that the accuracy of the algorithm increases. To investigate whether this is the case, a new experiment should be conducted in which loop closures are disabled, for the same reason as discussed in the previous section. A loop closure will always overrule the effects of estimation drift. However, it is not always possible to enforce a loop closure. Besides, having an algorithm that does not need to rely on loop closures to yield a high accuracy is desirable.

The same dataset as in the main experiment of this chapter is used, but a different scenario is simulated. After one full loop of the figure-8 trajectory, the loop closure detection module is disabled. This simulates an scenario in which the rover first drives around for a while in the same area, detecting loop closures to reduce the trajectory drift. This allows the algorithm to correctly estimate the IMU biases. Then, the rover enters a new area in which no loop closures can be detected. The point of this experiment is to investigate whether the novel visual-inertial SLAM algorithm shows a higher accuracy than the pure visual odometry, under the absence of loop closure constraints.

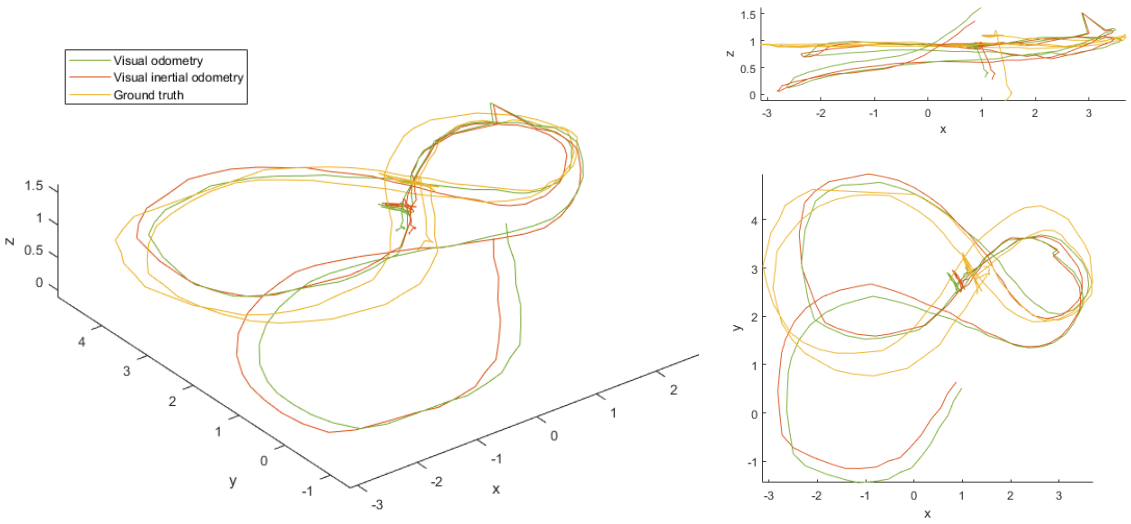


Figure 6-10: Estimated trajectories of visual and visual-inertial odometry without loop closures

Estimator	RMSE (m)
Visual odometry	1.1503
Visual-inertial odometry	1.0522

Table 6-1: RMSE scores for both estimators when mapping an unknown area

The calculated RMSE scores for both estimators are given in table 6-1. These metrics point out that the visual-inertial pipeline generates a trajectory that more closely fits the ground truth than the plain visual odometry. From this result, a conclusion could be that the drift

is therefore lower, which means that the incorporation of IMU measurements has improved the performance of the odometry. However, it is still obvious that the estimated trajectory drifts off, also for the visual-inertial odometry module. Upon visual inspection it is clear that it indeed shows less drift than the visual odometry, but the improvement is minimal. An explanation for this observation is the following: Even if the IMU factors are very close to the truth, as the visual odometry starts to drift off after pose 106, the visual-inertial pipeline 'notices' that the disagreement between IMU factors and VO factors increases. The way the algorithm deals with this discrepancy is to update the IMU biases, in order to minimise the error between the IMU- and VO-factors. This illustrates an inherent flaw of this type of tight sensor fusion: the estimation of the biases is performed by minimising the difference between IMU- and VO-factors. So as soon as the visual odometry module starts to drift off, the inertial measurements might offer some resistance and reduce this drift (assuming that the bias estimation until that point in time is done correctly). But as the VO further drifts off, it will influence the values of the estimated biases and hence influence the pre-integrated IMU measurements, deteriorating the accuracy of the IMU-factors. The main insight that can be gained from these results is the fact that the mathematical process of estimating these bias values is not correlated to the underlying physical process. The bias is just a measure of minimising the residual errors in the optimisation problem as best as possible, instead of reflecting the true bias values in the physical sensor.

Chapter 7

Conclusions

The generation of a 3D map of an unseen environment, obtained through solving the SLAM problem, is a popular topic currently in the robotics domain. In space applications, the hardware options are limited due to restrictions regarding weight and energy consumption, which is the reason why camera systems are favourable. Performing SLAM based on the visual measurements from a camera relies on visual odometry: the science of estimating the rover's motion through a sequence of visual measurements. The reliability on a single sensor however poses a threat to the robustness of the SLAM algorithm. The image quality can quickly degenerate in scenario's with varying lighting conditions, feature-poor environments or quick rotations around the rover's vertical axis. The visual odometry pipeline then loses track of the rovers current location, leaving the mapping thread unable to add any new visual observations to the global map until a relocalisation has taken place. Although such relocalisations are a reliable solution to the problem, it discards the information gathered between the moment tracking is lost and the relocalisation has taken place, which results in gaps in the global map of the environment.

The work done in this thesis aims to investigate how adding a second motion sensor can address this problem and establish a more robust and failure-resilient odometry pipeline. In the domain of space robotics, an inertial measurement unit (IMU) is a popular sensor due to its energy-efficiency and light weight. However, as IMU measurements tend to be contaminated with measurement biases, directly integrating the measured angular velocities and linear accelerations would quickly lead to the build up of errors in the estimated poses. A method to estimate these biases, in order to remove them from the raw IMU measurements, is called *tightly coupled sensor fusion*. This is done by comparing the motions found through visual odometry with the motions found through integration of IMU measurements, in which the bias is an yet unknown parameter. Theoretically, by optimising for the unknown biases over the full SLAM problem, these biases can be estimated by minimising the difference between the resulting motion from the two sensors.

To answer the research question, an IMU can serve as a temporary primary sensor to perform navigation in the event of a loss of visual measurements. A solution is proposed that involves constructing a non-linear factor graph, a graphical tool to model complex probabilistic distributions. This solution allows for the incorporation of various sensor types through summarising their measurements in so-called factors. These factors connect the unknown variables which are optimised for and contain information on how these variables are related. By building on the open-source GTSAM framework, leveraging an incremental smoothing algorithm, an efficient visual-inertial SLAM algorithm is developed that is able to run in real-time on the Lunar Rover Mini.

We were able to demonstrate that the novel algorithm is more failure-resilient than the pure visual algorithm, by simulating a scenario in which the visual odometry module temporarily fails to provide motion estimates to the optimiser. The visual-inertial odometry successfully handles a duration of absence of visual measurements of 3 seconds, showing the expected behaviour that during pure inertial navigation, the values of the estimated biases does not change. When visual odometry is regained, the optimiser is able to update all of the previous biases as well, yielding an even higher accuracy during the period where only inertial measurements are available. For a longer period of absence of visual measurements of 8 seconds, the plotted trajectories show more drift. However, by enforcing a loop closure this drift can be eliminated and sections of the estimated trajectory can be updated.

We were also able to demonstrate the sensitivity of the solution to variance parameters. A key parameter in updating the map is the noise models on the individual factors. As high covariance values in the noise models indicate a high uncertainty regarding that specific factor, the optimiser will trust this specific factor less, and vice versa. This dynamic covariance setting can be leveraged to include a certain level of certainty to the map. This valuable insight results in a much more useful and deeper understanding of the environment, which can be easily updated if new measurements provide additional evidence regarding this map. The generated map is now not merely a simple collection of points in a pointcloud anymore, but rather a reflection of a rover's understanding of his environment. This similarity to how human beings would behave if tasked to draw a map of an unknown environment is beautiful, indicating that computers are capable of imitating human beings in some aspects.

The developed SLAM algorithm based on tightly coupled sensor fusion, theoretically allows for more applications than only increasing the robustness of a single-sensor system. Adding another sensor to the estimation could increase the accuracy of the algorithm. Assuming Gaussian noise models for both measurement models, averaging the two can filter out potential outliers and errors that would have influenced the trajectory if the estimation relied on just one of the two sensors. This theory was tested in another experiment, in which loop closure detections were disabled. This experiment illustrated that the novel visual-inertial algorithm shows slightly lower drift than the pure visual odometry, but after a couple of seconds also drifts off. Recall that the main mechanism behind bias estimation is to figure out which biases causes the inertial navigation to resemble the visual navigation as much as possible. It therefore makes sense that as soon as the visual odometry starts to drift off, cause by the lack of loop closures, the biases will quickly be erroneously estimated in order to reduce the residual error in the optimisation as much as possible. This illustrates the main flaw of this type of visual-inertial navigation, as the bias estimation is not correlated to the underlying physical process, but is merely a variable in the optimisation problem which can be used by the optimiser to reduce the residual error.

The added value of this thesis to the DLR first and foremost is that the novel algorithm can serve as a framework or base to continue performing research on. The increased robustness of the algorithm, the primary result of this thesis, is only just a result of what is actually interesting to them: a novel method to perform visual-inertial SLAM. This perfectly fulfils the purpose of the Lunar Rover Mini, which is to serve as a experimental testing platform for new ideas and software designs. Although the idea of visual-inertial SLAM is certainly not new to them, the strategy of tightly coupled sensor fusion based on a non-linear factor graph has not been studied and implemented to this extend yet, as other rovers at DLR perform visual-inertial SLAM using a filtering approach.

A suggestion for future research concerns the further investigation into the estimation of biases. As the true biases are always unknown, it is very difficult to verify the degree to which they can successfully be estimated. By performing a simulated experiment with known values for the biases, a lot of insight can be gained into the estimation of these volatile parameters. In [21], researchers simulated a trajectory and the corresponding visual and inertial measurements, including the IMU biases and noise. A follow-up research involving such simulated experiments can provide a lot of understanding into the true capabilities of estimating them through non-linear factor graphs.

Appendix A

IMU-camera calibration report

Calibration results
==========
Normalized Residuals

Reprojection error (cam0): mean 0.5700995527100616, median 0.44635191784407013, std: 0.45088455346182316
Gyroscope error (imu0): mean 0.5735208904434693, median 0.4982112135433209 3, std: 0.3951283126526848 3
Accelerometer error (imu0): mean 0.7713668273472583, median 0.612789057751971, std: 0.6565329301424432

Residuals

Reprojection error (cam0) [px]: mean 0.5700995527100616, median 0.44635191784407013, std: 0.45088455346182316
Gyroscope error (imu0) [rad/s]: mean 0.05184210614117732, median 0.04503466053915858, std: 0.03571671802239935
Accelerometer error (imu0) [m/s^2]: mean 0.10878424987045357, median 0.08642035878781709, std: 0.0925894655937079

Transformation (cam0):

T_ci: (imu0 to cam0):
[[0.99985069 -0.00514407 0.0164964 0.0484304]
[0.00551334 0.99973348 -0.022418 -0.0667401]
[-0.01637669 0.02250561 0.99961258 -0.08896317]
[0. 0. 0. 1.]]

T_ic: (cam0 to imu0):
[[0.99985069 0.00551334 -0.01637669 -0.04951213]
[-0.00514407 0.99973348 0.02250561 0.06897361]
[0.0164964 -0.022418 0.99961258 0.0866336]
[0. 0. 0. 1.]]

timeshift cam0 to imu0: [s] (t_imu = t_cam + shift)
0.0063394190962141 81

Gravity vector in target coords: [m/s^2]
[-0.01601167 -9.79259292 -0.52277186]

Calibration configuration
==========

cam0

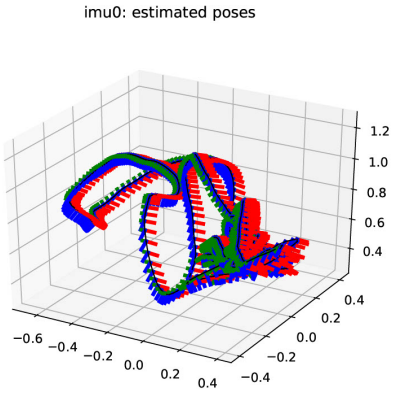
Camera model: pinhole
Focal length: [547.9864992476225, 545.0565501880732]
Principal point: [419.2175281448936, 228.15024235079835]
Distortion model: radtan
Distortion coefficients: [0.07322283348225525, -0.11421706243722514, 0.0001516519671137165, 0.00030571106477078353]
Type: aprilgrid
Tags:
Rows: 6
Cols: 6
Size: 0.022 [m]
Spacing 0.006599999999999999 [m]

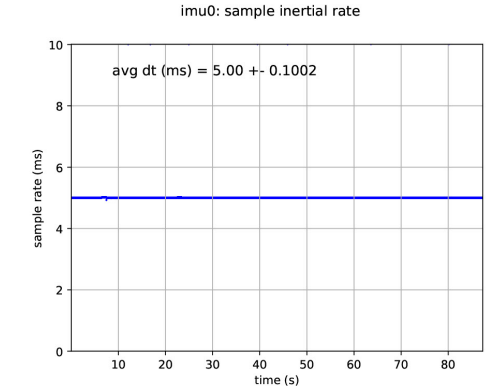
IMU configuration
==========

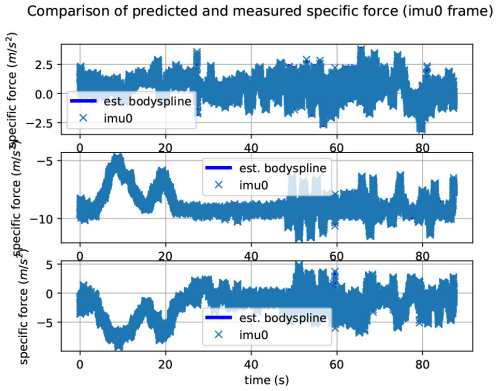
IMU0:

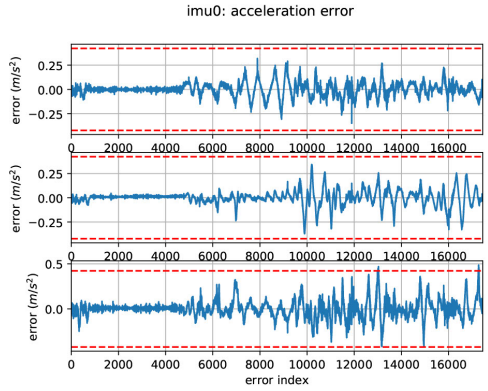
Model: calibrated
Update rate: 200.0
Accelerometer:
Noise density: 0.009972179
Noise density (discrete): 0.1410279078821217
Random walk: 0.001540749310997001
Gyroscope:
Noise density: 0.00639172965
Noise density (discrete): 0.09039270758052236
Random walk: 8.472210598431473e-05

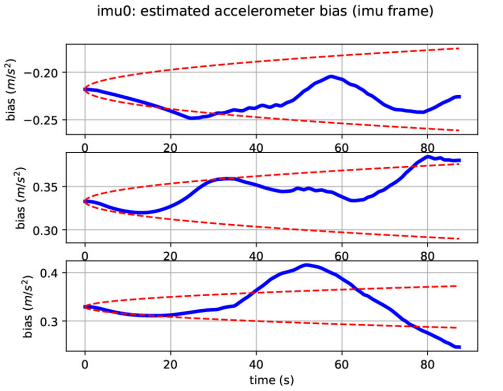
T_ib (imu0 to imu0)
[[1. 0. 0. 0.]
[0. 1. 0. 0.]
[0. 0. 1. 0.]
[0. 0. 0. 1.]]
time offset with respect to IMU0: 0.0 [s]

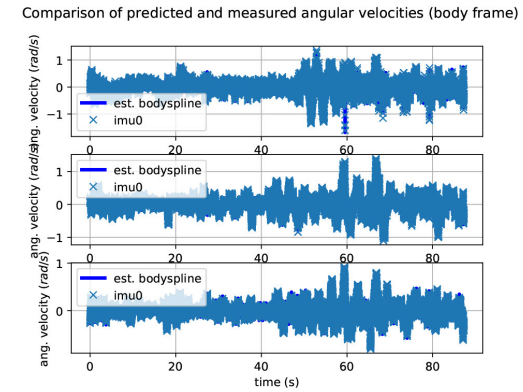


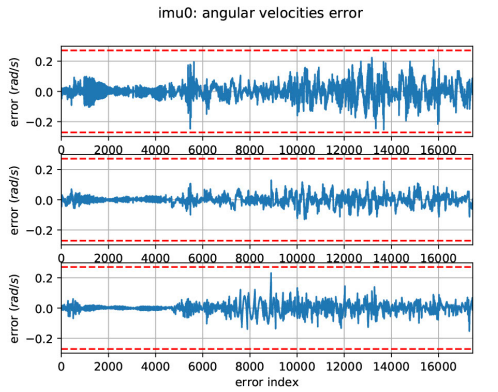

imu0: sample inertial rate
avg dt (ms) = 5.00 +- 0.1002
sample rate (ms)
time (s)

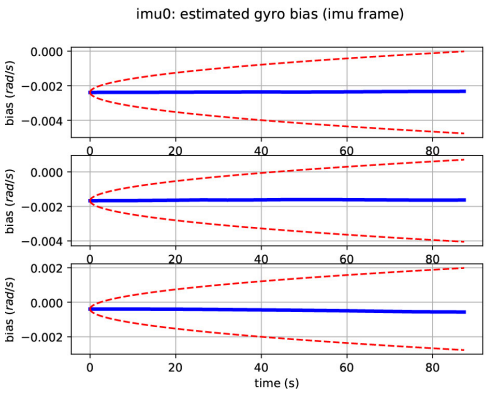

Comparison of predicted and measured specific force (imu0 frame)
est. bodyspline
imu0
specific force (m/s²)
time (s)

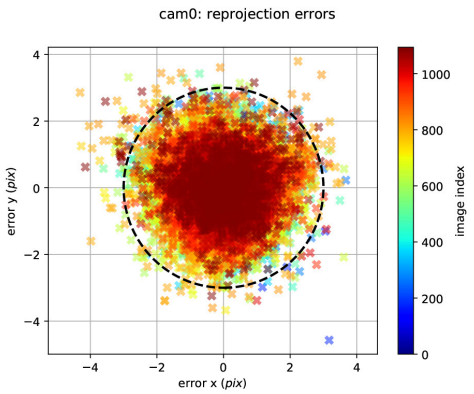

imu0: acceleration error
error (m/s²)
error index


imu0: estimated accelerometer bias (imu frame)
bias (m/s²)
time (s)


Comparison of predicted and measured angular velocities (body frame)
est. bodyspline
imu0
ang. velocity (rad/s)
time (s)


imu0: angular velocities error
error (rad/s)
error index


imu0: estimated gyro bias (imu frame)
bias (rad/s)
time (s)


cam0: reprojection errors
error y (pix)
error x (pix)
image index

Bibliography

[1] C. Stachniss, "Ekf slam," 2021.

[2] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 ed., 2004.

[3] D. M. Gavrila, "Lecture: 3d machinie vision," November 2021.

[4] Y.-C. Du, M. Muslikhin, T.-H. Hsieh, and M.-S. Wang, "Stereo vision-based object recognition and manipulation by regions with convolutional neural network," *Electronics*, vol. 9, no. 2, 2020.

[5] Y. S. Hung and W. K. Tang, "Projective reconstruction from multiple views with minimization of 2d reprojection error," *Int. J. Comput. Vis.*, vol. 66, no. 3, pp. 305–317, 2006.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[7] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234, 2007.

[8] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 International Conference on Computer Vision*, pp. 2320–2327, 2011.

[9] D. Yang, H. Ai, J. Liu, and B. He, "Absolute scale estimation for underwater monocular visual odometry based on 2-d imaging sonar," *Measurement*, vol. 190, p. 110665, 2022.

[10] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: a versatile and accurate monocular slam system.," *CoRR*, vol. abs/1502.00956, 2015.

[11] M. M. Butt, H. Zhang, X. Qiu, and B. Ge, "Monocular slam initialization using epipolar and homography model," in *2020 5th International Conference on Control and Robotics Engineering (ICCRE)*, pp. 177–182, 2020.

[12] S. Thrun and J. J. Leonard, *Simultaneous Localization and Mapping*, pp. 871–889. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[13] V. Imani, K. Haataja, and P. Toivanen, "Three main paradigms of simultaneous localization and mapping (slam) problem," p. 74, 04 2018.

[14] I. R. Laboratory, "Visual slam."

[15] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Real-time monocular slam: Why filter?," in *2010 IEEE International Conference on Robotics and Automation*, pp. 2657–2664, 2010.

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

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

[18] C. Chen, H. Zhu, M. Li, and S. You, "A review of visual-inertial simultaneous localization and mapping from filtering-based and optimization-based perspectives," *Robotics*, vol. 7, no. 3, 2018.

[19] T. Rowland, "Manifold."

[20] G. S. Chirikjian, *Stochastic models, information theory, and Lie groups, volume 2: Analytic methods and modern applications*, vol. 2. Springer Science & Business Media, 2011.

[21] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, pp. 1–21, feb 2017.

[22] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, "Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4304–4311, 2016.

[23] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1280–1286, 2013.

[24] P. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *2012 IEEE International Conference on Robotics and Automation*, pp. 2088–2095, 2012.

[25] J. Maye, P. Furgale, and R. Siegwart, "Self-supervised calibration for robotic systems," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 473–480, 2013.

[26] L. Oth, P. Furgale, L. Kneip, and R. Siegwart, "Rolling shutter camera calibration," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1360–1367, 2013.

[27] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[28] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Transactions on Robotics*, vol. 37, pp. 1874–1890, dec 2021.

[29] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271–1278, 2016.

[30] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, 2011.

[31] F. Dellaert and G. Contributors, "borglab/gtsam," May 2022.

[32] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Factor graph based incremental smoothing in inertial navigation systems," in *2012 15th International Conference on Information Fusion*, pp. 2154–2161, 2012.

[33] J. Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

[34] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," vol. 6314, pp. 778–792, 09 2010.

[35] M. J, "Absolute orientation - horns method,"