الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

École Nationale Polytechnique

Département d'Automatique

Laboratoire de Commande des Processus

End of studies project thesis

Submitted in partial fulfillment of the requirements
for the State Engineer Degree in Automation Engineering

Visual SLAM on lie groups

Realized by:
Mr. Boudjoghra Med El Amine
Mr. Daimellah Sofiane Sid Ali

Supervised by:
Pr. Tadjine Mohamed
Pr. Tayebi Abdelhamid

Publicly presented and defended on the 25th of June, 2022.

Jury members:

President	M. CHAKIR Messaoud	Pr	ENP
Promoter	M. TADJINE Mohamed	Pr	ENP
	M. TAYEBI Abdelhamid	Pr	Lakehead University
Examiner	M. BENREZKI Riadh Rabie	Dr	CDTA

ENP 2022

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

École Nationale Polytechnique

Département d'Automatique

Laboratoire de Commande des Processus

End of studies project thesis

Submitted in partial fulfillment of the requirements
for the State Engineer Degree in Automation Engineering

Visual SLAM on lie groups

Realized by:
Mr. Boudjoghra Med El Amine
Mr. Daimellah Sofiane Sid Ali

Supervised by:
Pr. Tadjine Mohamed
Pr. Tayebi Abdelhamid

Publicly presented and defended on the 25th of June, 2022.

Jury members:

President	M. CHAKIR Messaoud	Pr	ENP
Promoter	M. TADJINE Mohamed	Pr	ENP
	M. TAYEBI Abdelhamid	Pr	Lakehead University
Examiner	M. BENREZKI Riadh Rabie	Dr	CDTA

ENP 2022

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

École Nationale Polytechnique

Département d'Automatique

Laboratoire de Commande des Processus

Mémoire de projet de fin d'études

pour l'obtention du diplôme d'Ingénieur d'État en Automatique

SLAM visuel sur les groupes de lie

Réalisé par :
M. Boudjoghra Med El Amine
M. Daimellah Sofiane Sid Ali

Supervisé par:
Pr. Tadjine Mohamed
Pr. Tayebi Abdelhamid

Présenté et soutenue publiquement le 25 Juin 2022.

Membres du jury :

Président	M. CHAKIR Messaoud	Pr	ENP
Promoteur	M. TADJINE Mohamed	Pr	ENP
	M. TAYEBI Abdelhamid	Pr	Lakehead University
Examinateur	M. BENREZKI Riadh Rabie	Dr	CDTA

ENP 2022

ملخص

يركز هذا العمل على الاستفادة من إمكانات نظرية لي في تقدير الحالة لاشتقاق مراقب غير خطي لحل مشكلة التموضع المتزامن ورسم الخرائط. في واقع الأمر ، أثبتت المجموعات SE(3) و SO(3) أنها مفيدة جدًا في تمثيل حركات الجسم في مساحة ثلاثية الأبعاد. لذلك ، يصبح من الممكن تصميم مراقبين غير خطيين لحل مشكلة SLAM باستخدام تحليل استقرار Lyapunov ، والذي نوضحه في أطروحتنا من خلال تقديم عمل[1]. تتمثل مساهمتنا في منح المراقب ميزتين عمليتين: ميزة تعزيز النظام ، لإعطاء السيارة القدرة على تغيير أبعاد مصفوفة الحالة ديناميكيًا ، وكتلة اكتشاف الأعطال والعزل التي تكتشف وتصحح القياسات الخاطئة من الكاميرا و IMU لتنفيذ المراقب المقترح.

كلمات مفتاحية : نظرية لي, SLAM ,مراقب غير خطي.

Résumé

Ce travail se concentre sur l'exploitation de la théorie de Lie dans le domaine de l'estimation d'état afin de développer une solution non linéaire au problème de *localisation et cartographie simultanées* (SLAM). En effet, les groupes *SE(3)* et *SO(3)* se sont avérés très pratique pour représenter le mouvement d'un corps évoluant dans un espace 3D. Par conséquent, il est possible de concevoir des observateurs non linéaires afin de résoudre le problème SLAM à travers une analyse de stabilité de Lyapunov, ce qui est démontré dans notre thèse en présentant le travail de [1]. Notre contribution consiste à doter l'observateur de deux fonctionnalités pratiques : une fonctionnalité de redimensionnement de système, afin de permettre au véhicule de dynamiquement changer la dimension de la matrice d'état, ainsi qu'un bloc de Détection et d'Isolation de Faute afin de détecter et corriger les mesures faussées de la caméra et de l'IMU utilisées pour implémenter l'observateur proposé.

Mots clés : Théorie de Lie, SLAM, observateur non linéaire.

Abstract

This work focuses on leveraging the potential of Lie theory in state estimation to derive a nonlinear approach for solving the *Simultaneous Localization And Mapping* problem. As a matter of fact, the groups *SE(3)* and *SO(3)* have proven to be very convenient in representing body motions in 3D space. Therefore, it becomes possible to design nonlinear observers for solving the SLAM problem using Lyapunov stability analysis, which we demonstrate in our thesis by presenting the work of [1]. Our contribution consists of endowing the observer with two practical features: a *System re-dimensioning* feature, to give a vehicle the ability to dynamically change the dimension of the state matrix, and a *Fault Detection and Isolation* block that detects and corrects faulty measurements from the camera and the IMU used to implement the proposed observer.

Keywords : Lie theory, SLAM, nonlinear observer.

Dedication

“

To our dear parents, our families, and our friends.

”

- Sofiane & Amine

Acknowledgments

First of all, we thank God the Almighty for giving us the courage, the will and the patience to carry out this work.

We thank our parents, who supported us throughout our studies' journey.

We would like to express our gratitude to our supervisors, Pr. Abdelhamid TAYEBI and Pr. Mohamed TADJINE, for their consistent support, guidance and clarifications during the running of this thesis.

We would also like to thank in advance Pr. CHAKIR Messaoud and Dr. BENREZKI Riadh Rabie, for evaluating our project.

Each one of us would also like to thank the other, for having the courage to embark on a stranger field that has as expected seduced and benefited us a lot.

Contents

List of Figures

List of Abbreviations

GPS	*Global Positioning System*
SO(3)	*Special Orthogonal Group*
SE(3)	*Special Euclidean Group*
KF	*Kalman Filter*
PDF	*Probability Density Function*
EKF	*Extended Kalman Filter*
PGO	*Pose Graph Optimization*
SLAM	*Simultenious Localization And Mapping*
V-SLAM	*Visual Simultenious Localization And Mapping*
LIDAR	*Light Detection And Ranging*
UAV	*Unmanned Aerial Vehicles*
IMU	*Inertial Measurment Unit*
UGR	*Unmanned Ground Robots*
USV	*Unmanned Surface Vehicles*
HMD	*Handheld Mapping Devices*
FDI	*Fault Detector and Isolator*

Chapter 1

General Introduction

SLAM is a very popular topic in the mobile robotics community, and it has been an open problem among researchers during the last 30 years [2]. It stands for *Simultaneous Localization And Mapping*, and it basically consists of giving the ability to a mobile vehicle of simultaneously building a consistent map of its surrounding by estimating the position of some discriminant features[1] in the environment, and localizing itself within that very map by estimating its position and orientation relative to the features describing the estimated map. Over the years, SLAM has found applications in numerous fields, such as self-driving cars, augmented reality...etc. But it is generally used in indoor applications where there is no access to an absolute positioning system such as GPS [3]. Two of the most popular state-of-the-art techniques used to deal with the SLAM problem are the Extended Kalman Filter (EKF) and the Pose Graph Optimization (PGO). However, these techniques approach the SLAM problem through linearization, and do not deal very well with poor initial guess. Additionally, these probabilistic approaches are considerably computationally expensive and do not perform well in the long run [4]. Most recently, the nonlinear nature of the SLAM problem has caught the interest of many researchers from the nonlinear observer community. This is due to the fact that the pose dynamics of a vehicle moving in 3D space are highly nonlinear, and are best modeled using the *Special Euclidean group SE(3)*. Also, the dynamics of the features describing the map rely on the orientation of the vehicle, which is best represented using the *Special orthogonal group SO(3)*. These two groups are two of the most popular Lie groups used in solving state estimation problems in robotics. Therefore, it becomes legitimate to believe that Lie groups based approaches can fit the true nature of the SLAM problem and lead to globally stable observers. Consequently, many Lie groups based observers have been designed during the most recent years [1], [4]–[6].

In this work, we presented a Geometric Nonlinear Observer on Matrix Lie groups proposed by Tayebi *et al.* in [1], where they extended the SE(3) group and its Lie algebra $\mathfrak{se}(3)$ to encompass positions and velocities of the landmarks (measured using a camera and an IMU), which they defined as $SE_{n+1}(3)$ and $\mathfrak{se}_{n+1}(3)$, respectively. Additionally, they provided an analysis of the global convergence of the proposed observer, and finalized the paper by extending the observer to compensate angular and linear velocity biases. Finally, their work also deals with dynamic environments, as the landmarks can also take arbitrary velocities.

This report consists of four chapters, alongside the general introduction and conclusion. Chapter 2 consists of a literature review of the SLAM problem, where we provide a brief overview of the most consequent advancements towards solving the SLAM problem during the last three to four decades in a chronological manner. Chapter 3 provides a light introduction to Lie theory, which will be needed later on in the next chapter. Chapter 4 presents the Lie groups based nonlinear observer proposed in [1]. Finally, chapter 5 presents a robust SLAM on lie groups when facing landmarks uncertainties.

[1]These features can be landmarks, point clouds, planes, surfels...

Chapter 2

Literature review

2.1 History of SLAM

Autonomous navigation has always had a central place in the mobile robotics community and caught the interest of the most experienced researchers. And one of the main capabilities that a robot must have in order to aim for autonomy is the ability to localize itself relative to its environment, as this would enable it to achieve the subsequent building blocks of autonomous navigation, such as path planning and online decision-making (see Fig 2.1). In early applications, the location of a mobile robot evolving in a **known environment** was obtained by integrating wheel encoders, which is referred to as *odometry*, and which quickly drifts because of the accumulation of the measurement errors caused by the encoders. Afterwards, and with the emergence of new sensors of higher precision (cameras, inertial measurement units...), new odometry algorithms based on visual and inertial information arose. These algorithms have proven to be way more reliable and showed less significant drifts ($< 0.5\%$ of the whole trajectory [7]). However, localization using this kind of approaches do require either a manually built map of the environment, or access to an absolute positioning system such as GPS. What actually makes a robot truly autonomous is its ability to learn about its environment through exploration and navigate without the need of a pre-build map. In other words, the robot needs to consistently estimate its surroundings (by building a map of its environment) before estimating its state[1] relative to its environment.

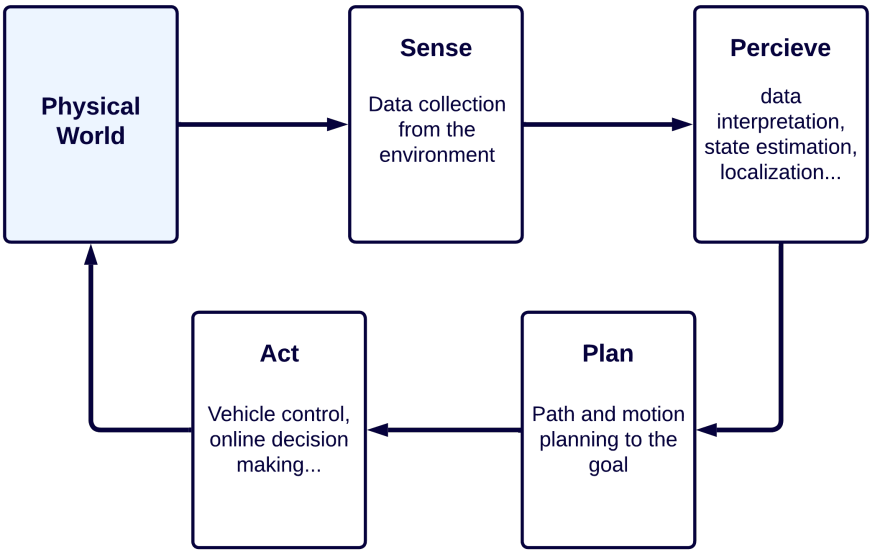


Figure 2.1: Building blocks of autonomous navigation

Seeking for true autonomy, researchers agreed in the *IEEE International Conference on Robotics and Automation*, which was held in San Francisco [8], that self-sufficient localization and mapping was an open problem that needed to be explored from various

[1]The state of the robot usually contains information about its pose, velocity, measurement biases...etc.

perspectives (machine vision, signal processing, optimization, graph theory, geometry, probabilistic estimation...)[9]. Ever since, many works have been proposed to lay down the foundations in the different fields needed to implement such a system ([10]–[16]), which finally resulted in the idea of joining the vehicle's and the environment's states in a larger vector and estimating them simultaneously, and the term SLAM (Simultaneous Localization And Mapping) was first introduced by Durrant-Whyte et al. in [17].

Nowadays, and after more than three decades from the emergence of SLAM, it is now a reliable solution for map and pose estimation in robotics, and it has found many applications in self-driving cars [18], augmented reality [19], and other indoor and outdoor applications that require the building of a consistent map of the environment [9], [20]. Solutions to the SLAM problem had been constantly proposed from the day of its emergence until now, and it has been solved on a theoretical level in various ways, and using different sensors (Lidar and Visual SLAM [21]) and approaches (smoothing and filtering approaches [3]), so that the developer now, has the ability to choose from a broad collection of algorithms based on his requirements (in terms of efficiency, cost...etc.).

In this chapter, and because it is not the main subject of our work, we will only provide a shallow overview of the SLAM problem, by presenting its components, its classifications, and some of its approaches. However, we redirect the reader to more comprehensive overviews of SLAM ([3], [9]), especially the work of Cadena et al. where they make a very interesting distinction between the eras of SLAM (the classical age and the algorithmic-analysis age) and they even argue that we are entering a third era which they called the robust-perception age, and they provided the main characteristics of each period that SLAM went through. We also redirect the reader to a deeper historical review of the first 20 years of SLAM ([8], [22]), where Durrant–Whyte and Bailey presented the main probabilistic formulations for SLAM [9].

2.2 Components of SLAM

The anatomy of a typical SLAM solution is made of two main components, namely, the frontend component, and the backend component (see Fig 2.2). Where the frontend component is responsible for processing and basically making sense out of sensor data. whereas, the backend component handles the optimization underlying the SLAM problem to both estimate the vehicle's and the environment's states.

2.2.1 The frontend component

Typically, SLAM uses either cameras or laser scanners to gather information from the environment. And that data in its raw state cannot be fed directly into an optimization algorithm because of its high complexity and level of abstraction. In other words, it is impossible to design a function that describes the environment in an enough general manner while not exceeding the currently-supported complexity by the chips that are used in most robotics applications. Instead, sensor data need to be preprocessed and well understood before used for estimation. And that is where the role of the frontend component lies, which generally includes a short term data association block, responsible for detecting the tracking recently seen features (in two consecutive measurements for example), and a

long term data association block, responsible for associating new measurements of older explored scenes of the environment (which is referred to as loop closure). In visual-SLAM, where the main sensor used is either a monocular or a stereoscopic camera, feature detection and tracking is performed using a very well known technique in machine vision, namely structure for motion (SfM) [23]. On the other hand, in Lidar-SLAM, iterative closets point (ICP) [24] is used to achieve feature matching.

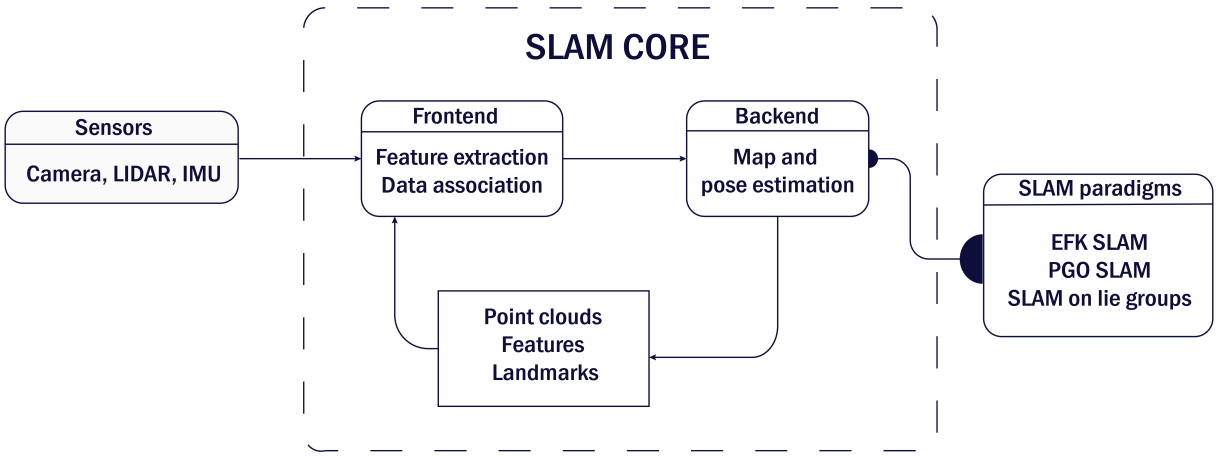


Figure 2.2: The architecture of modern SLAM solutions

2.2.2 The backend component

After transforming the raw sensor data into meaningful information that are amenable for optimization in the frontend component, the backend component estimates the map and the pose of the vehicle by optimizing a cost function whose formulation is chosen in advance (one fundamental formulation of the SLAM problem is the maximum a posteriori). The backend component also feeds back information to the frontend in order to detect, optimize and validate loop closures.
N.B. We will not provide any mathematical development in this section. However, two of the most popular SLAM solvers are presented later in this chapter.

2.3 Classifications of SLAM

Over the years, a tremendous amount of SLAM solvers had been proposed, and each one of them focussed on one aspect of SLAM and had, at a certain level, a target application in mind. Therefore, there exist nowadays, various solutions of the SLAM problem that one can choose from based on the application requirements. And as a result, these implementations of SLAM can be classified in various manners, based on the sensors used, or the representation of the environment. As an example, if a SLAM algorithm considers images as input data, then we call it Visual-SLAM. On the other hand, if it considers point clouds as its inputs, then we call it Lidar-SLAM. In this section, we will introduce the reader to two main ways of classifying SLAM solvers, and we will illustrate the advantage of each category.

2.3.1 LIDAR-SLAM vs Visual-SLAM

The main difference between LIDAR-SLAM and Visual-SLAM, is the type of sensor used to provide feedback. Clearly, the chosen sensor determines the type of acquired information about its surroundings (e.g. a camera provides images, while LIDAR provides a point cloud representing nearby objects), so input data can take different forms, and it's up to the researcher to make use of the acquired data to solve the SLAM problem.

In the case of Visual-SLAM, a camera is used to localize itself while simultaneously mapping the environment, using a sparse representation of many 3D points to describe distinguishable features within the environment; the front-end algorithm processes successive frames of the visualized scene in order to compute the pose of a detected feature within the scene. Even though the word *visual* refers to cameras, still there are different types of them, and depending on the type, different algorithms have been proposed. Among them, we find the *MonoSLAM*, which was developed in 2003 by *Davison et al.* ([25], [26]), and which makes use of a single monocular camera. Another algorithm that uses an RGB-D camera (e.g. kinect camera) is proposed in [27], and is called *RGB-D SLAM*. The latter is more accurate than the MonoSLAM, as it relies on a more developed type of cameras, which provides measurements containing depth information.

The LIDAR-SLAM, on the other hand, uses a LIDAR to extract dense information about the surrounding environment. A LIDAR emits an eye-safe laser and detects the light reflected on the surrounding objects, then it calculates the distance using the measured time between light emission and reception multiplied by half the speed of light. Unlike the Visual-SLAM which represents the environment using a sparse representation, the LIDAR-SLAM creates a more precise dense representation that shows all the geometrical properties of the objects around the LIDAR, and this is better than the sparse representation when avoiding obstacles.

The main advantages of cameras over Lidars is that they are less expensive and less cumbersome, as we know that Lidars usually cost more and take more space in the vehicle. In terms of data processing, the images taken with cameras do require heavy data preprocessing using computationally expensive computer vision algorithms in order to extract useful data. Whereas, the scans of a LIDAR can be used by the SLAM algorithm after light processing. Therefore, even though cameras are usually less expensive than Lidars, their high computational need requires more sophisticated calculators, which can also prove to be costly.

2.3.2 2D vs 3D SLAM

Another way to classify SLAM techniques, is their way of representing the map. Certainly, environment modeling plays a critical role to solve computational problems like path planning, where the optimal path of a drone, for example, can be a 3 dimensional trajectory, that requires a 3D construction of the map which calls for 3D SLAM. On the other hand, path planning on UGR can be accomplished without any need for 3D SLAM, thus 2D SLAM suffices.

When a SLAM method takes only 2 dimensions into account, it becomes classified as a 2D SLAM. Although this class seems to be incomplete, as it ignores the third dimension, it was studied enormously throughout the years, where several 2D SLAM methods had

proved to be extremely efficient in practice for navigation, path planning, and obstacle avoidance ([28]–[32]). Among the commonly known 2D SLAM algorithms, we find the *HectorSLAM*, which is LIDAR based. This latter, proved to be extremely efficient on UGR, USV, and HMD ([33], [34]).

Even though, 2D SLAM methods are utilized mostly in robotics, they remain relatively limited compared to their counterparts 3D SLAM methods. In the same way 2D SLAM was defined, 3D SLAM is a problem where we attempt to construct a 3D map of the environment while localizing the measurement sensor relative to this map. Thus, we can deduce that 3D SLAM methods offer a more clear observability of the surroundings, and allows the robot to navigate in a 3D space more freely. A method of 3D map reconstruction using an RGB-D camera is presented in [35], while a non-linear observer which helps in estimating the 3D map and localizing the robot relative to this map is presented in [1]. Several researches were also conducted to solve the 3D SLAM problem using a LIDAR sensor, including ([36], [37]).

2.4 Different approaches of SLAM

2.4.1 The Extended Kalman Filter based SLAM

In this section, we will first introduce some probabilistic laws that are required for a clear understanding of the estimation process. Then, we will define the *Bayes Filter* which is derived from the *Bayes theorem*, along with some mathematical definitions. Next, we will present a practical application of the Bayes Filter, namely, the *Kalman Filter*, and we will expose its pros and cons. Finally, we will introduce a more flexible version of the Kalman filter and which is referred to as the *Extended Kalman filter*.

2.4.1.1 The Bayes Filter

The definition of a state

We can define a state as a collection of internal and external variables of a system that can have an influence on it. It can be either *dynamic*, where the variables change over time, e.g. the pose of a robot, or *static*, where the variables remain unchanged over time, e.g. the location of some landmarks like trees or walls. We denote a state vector X_t, where the t index refers to its dynamic nature.

Probabilistic laws

We denote the probability distribution over a state X_t conditioned on another event A, $P(X_t|A)$. Since we know that X_t is generated conditionally on the past states $X_{0:t-1} = \begin{bmatrix} X_0 & X_1 & ... & X_{t-1} \end{bmatrix}$, all past input data $U_{1:t} = \begin{bmatrix} U_1 & U_2 & ... & U_t \end{bmatrix}$, and all past measurements $z_{1:t-1} = \begin{bmatrix} z_1 & z_2 & ... & z_{t-1} \end{bmatrix}$, We conclude that $A = \{X_{0:t-1}, z_{1:t-1}, U_{1:t}\}$. Thus, we can write the probability distribution of X_t as :

$$p(X_t|X_{0:t-1}, z_{1:t-1}, U_{1:t}) \tag{2.1}$$

We can further simplify the formula of the probability distribution of X_t knowing that a state X_{t-1} is generated conditionally to $\{X_{0:t-2}, z_{1:t-2}, U_{1:t-1}\}$ along with an effectuated measurement z_{t-1}, which indicates the sufficiency of ${X_{t-1}}^2$ to replace $\{X_{0:t-2}, z_{1:t-1}, U_{1:t-1}\}$. Thus, we get :

$$p(X_t|X_{0:t-1}, z_{1:t-1}, U_{1:t}) = p(X_t|X_{t-1}, U_t) \tag{2.2}$$

Similarly, we can write the probability distribution of a measurement z_t as :

$$p(z_t|X_{0:t}, z_{1:t-1}, U_{1:t}) \tag{2.3}$$

and since we know that X_{t-1} is generated conditionally to $\{X_{0:t-2}, z_{1:t-2}, U_{1:t-1}\}$, along with an effectuated measurement z_{t-1}, we can bring (2.3) down to :

$$p(z_t|X_t) \tag{2.4}$$

Belief distributions

Knowing that state estimation is always subject to noise, a robot can never know perfectly its true state, as it relies on measurements and input data to generate an approximation of the real state. A way to describe this imperfect knowledge of the environment is the *belief*, where the belief simply is the probability distribution of a state X_t given all measurements and input data up to the instant of estimation. We mathematically present it as :

$$bel(X_t) = p(X_t|z_{1:t}, U_{1:t}) \tag{2.5}$$

Bayes Filter algorithm

The Bayes theorem is a mathematical formula that describes the relationship between the probability distributions of two events A and B, and their conditioned probability relative to each other. It is mathematically described as

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.6}$$

where $P(A|B)$ defines the probability distribution of A given B and $P(B|A)$ is the probability distribution of B given A. We will next use this theorem and the previous probabilistic laws (2.2) and (2.4), to derive the Bayes filter. This filter allows us to recursively estimate the belief of a state X_t using the belief of the past state X_{t-1} and some partial knowledge of the true state obtained from measurements.
We know from (2.5) that

$$P(X_t|z_{1:t}, U_{1:t}) = \frac{P(z_t|X_t, z_{1:t-1}, U_{1:t}) \cdot P(X_t|z_{1:t-1}, U_{1:t})}{P(z_t|z_{1:t-1}, U_{1:t})} \tag{2.7}$$

[2]When a state is sufficient to replace all past events, it is called *complete* under the *Makov assumption* [38], and for it to be complete all variables that can influence the state of the robot must be included in the vector state X_t, while assuming that the used probability distribution model is accurate enough.

from the Markov assumption, which assumes the completeness of X_t , we can safely say that past measurements and past input data will not provide additional information when measuring the state at an instant t, mathematically speaking we write

$$P(z_t|X_t, z_{1:t-1}, U_{1:t}) = P(z_t|X_t) \tag{2.8}$$

which is similar to the result we got in (2.4), this result is an example of *conditional independence* [39], which is defined for three events a, b, and c as $P(a|b,c) = P(a|c)$ when $P(a|b,c)$ does not depend on b, here we say that a is conditionally independent of b given c.
Using (2.8) and *the law of total probability* [40], we obtain

$$\begin{aligned} P(X_t|z_{1:t}, U_{1:t}) &= \frac{P(z_t|X_t, z_t) \cdot P(X_t|z_{1:t-1}, U_{1:t})}{P(z_t|z_{1:t-1}, U_{1:t})} \\ &= \frac{P(z_t|X_t) \cdot \int_{-\infty}^{+\infty} P(X_t|X_{t-1}, z_{1:t-1}, U_{1:t}) P(X_{t-1}|z_{1:t-1}, U_{1:t}) dX_{t-1}}{P(z_t|z_{1:t-1}, U_{1:t})} \end{aligned}$$

We denote the predicted belief as $\overline{bel}(X_t)$ and we define it as follows,

$$\overline{bel}(X_t) = \int_{-\infty}^{+\infty} P(X_t|X_{t-1}, z_{1:t-1}, U_{1:t}) P(X_{t-1}|z_{1:t-1}, U_{1:t}) dX_{t-1} \tag{2.9}$$

Knowing that U_t adds no extra information about $P(X_{t-1}|z_{1:t-1}, U_{1:t})$, we conclude that $P(X_{t-1}|z_{1:t-1}, U_{1:t}) = P(X_{t-1}|z_{1:t-1}, U_{1:t-1})$, and using the Markov assumption of a complete state, we can bring the formula of the predicted Bayes filter down to

$$\overline{bel}(X_t) = \int_{-\infty}^{+\infty} P(X_t|X_{t-1}, U_t) bel(X_{t-1}) dX_{t-1} \tag{2.10}$$

Finally, we get the recursive formula of the Bayes filter

$$bel(X_t) = \frac{1}{P(z_t|z_{1:t-1}, U_{1:t})} P(z_t|X_t) \overline{bel}(X_t) \tag{2.11}$$

2.4.1.2 Kalman filter

Introduction

In this subsection, we will present a recursive state estimator called the *Kalman Filter*, which is discussed in [41]. This filter is a practical implementation of the *Bayes Filter* for linear systems. Its purpose is to estimate the belief of a current state given some noisy measurements, an input, and a belief representing the prior state. We encourage reading the *Kalman Filter* subsection in [41] for more details on how the algorithm is derived. We also provide some graphical demonstrations (Fig. 2.4) to better imagine how the estimator functions.

We will start first by defining the mathematical tools needed to understand the KF, then we will introduce the algorithm (1).

State estimation

Similar to any other Gaussian estimator, the KF uses the multivariate normal distribution over a state X, its mathematical formula is stated in (2.12), and its graphical representation is depicted in (Fig. 2.3)

$$P(X) = det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right) \tag{2.12}$$

where μ represents the mean with the same dimension as the state vector X, and represents the state vector with the highest probability. Meanwhile, the covariance matrix Σ represents the uncertainty of the state X and has a dimension equal to square the dimension of X, this matrix is symmetric positive semi-definite, where it controls the range around the peak at the mean. Throughout this subsection, we will refer to the set (μ, Σ) as a *moment*, while we will refer to (2.12) as the *moments' representation*.

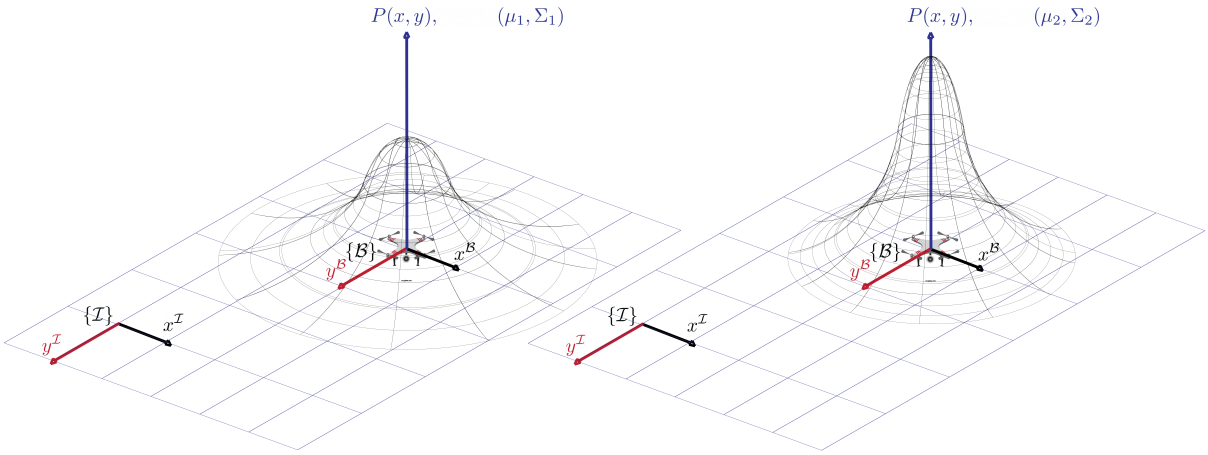


Figure 2.3: The influence of the covariance matrix on the normal distribution. This figure shows the normal probability distribution over two states $X_1 = (x_1, y_1) \in \mathbb{R}^2$ and $X_2 = (x_2, y_2) \in \mathbb{R}^2$ representing the x and y coordinates of the robot, one with a moment $(\mu_1, \Sigma_1) \in \mathbb{R}^2 \times \mathbb{R}^{2\times2}$ and the other with $(\mu_2, \Sigma_2) \in \mathbb{R}^2 \times \mathbb{R}^{2\times2}$. In this illustration, we assumed that the robot's orientation is completely noiseless, and the Gaussian noise affects only the x and y coordinates.

The linear mathematical model of any system is written as follows

$$X_t = A_t X_{t-1} + B_t U_t + \epsilon_t \tag{2.13}$$

where the state vector at instant t is denoted $X_t \in \mathbb{R}^n$, the input vector is denoted $U_t \in \mathbb{R}^m$, the state matrix is denoted $A_t \in \mathbb{R}^{n\times n}$, and the input matrix is denoted $B_t \in \mathbb{R}^{n\times m}$, while $\epsilon_t \sim \mathcal{N}(0_n, \Sigma_{\epsilon_t})$ represents a Gaussian additive noise with a mean 0_n and a covariance matrix $\Sigma_{\epsilon_t} \in \mathbb{R}^{n\times n}$. We note that the index t defines the dynamic nature of the indexed matrices and vectors.

Next, we will define the linear measurement model

$$z_t = C_t X_t + \xi_t \tag{2.14}$$

Where $z_t \in \mathbb{R}^p$ is the measurement of the state, $C_t \in \mathbb{R}^{p\times n}$ is the output matrix, $\xi_t \sim \mathcal{N}(0_p, \Sigma_{\xi_t})$ is a Gaussian additive noise with a mean 0_p and a covariance matrix $\Sigma_{\xi_t} \in \mathbb{R}^{p\times p}$.

The KF algorithm is built to estimate a corrected belief of a state X_t given a prior vector state X_{t-1}, an input U_t, and a measurement of the state z_t. However, for it to properly function, three assumptions are made. The first one is to have a normally distributed initial belief, while the second one is the linear nature of the motion model. The second constraint will allow the KF to maintain the Gaussian form of the belief over the X_t state vector. The last assumption is a linear measurement model, also to assure a normal distribution of the probability density function, which will help to maintain the Gaussian nature of the belief during the correction step.

The arguments of the KF algorithm are the moment $(\mu_{t-1}, \Sigma_{t-1})$, some input U_t, and some measurement z_t, while it returns the moment (μ_t, Σ_t) of the current state X_t. A graphical demonstration of the path estimation from the initial state X_0 to the state X_t is depicted in (Fig. 2.4), where the belief of a state X_i denoted $bel(X_i)$ is associated with a moment (μ_i, Σ_i)

Algorithm 1 Kalman_Filter$(\mu_{t-1}, \Sigma_{t-1}, U_t, z_t)$

Data: $\mu_{t-1}, \Sigma_{t-1}, U_t, z_t$

Result: μ_t, Σ_t

1 $\bar{\mu} \leftarrow A_t \mu_{t-1} + B_t U_t$

2 $\bar{\Sigma}_t \leftarrow A_t \Sigma_{t-1} A_t^T + \Sigma_{\epsilon_t}$

3 $K_t \leftarrow \bar{\Sigma}_t C_t^T \left(C_t \bar{\Sigma}_t C_t^T + \Sigma_{\xi_t} \right)^{-1}$

4 $\mu_t \leftarrow \bar{\mu}_t + K_t \left(z_t - C_t \bar{\mu}_t \right)$

5 $\Sigma_t \leftarrow (I - K_t C_t) \bar{\Sigma}_t$

where K_t is the Kalman gain and $I \in \mathbb{R}^{n \times n}$ is the identity matrix.
We describe the belief of a state X_t as

$$bel(X_t) = det(2\pi\Sigma_t)^{-\frac{1}{2}} \exp\left(-\frac{1}{2} (X_t - \mu_t)^T \Sigma_t^{-1} (X_t - \mu_t) \right) \tag{2.15}$$

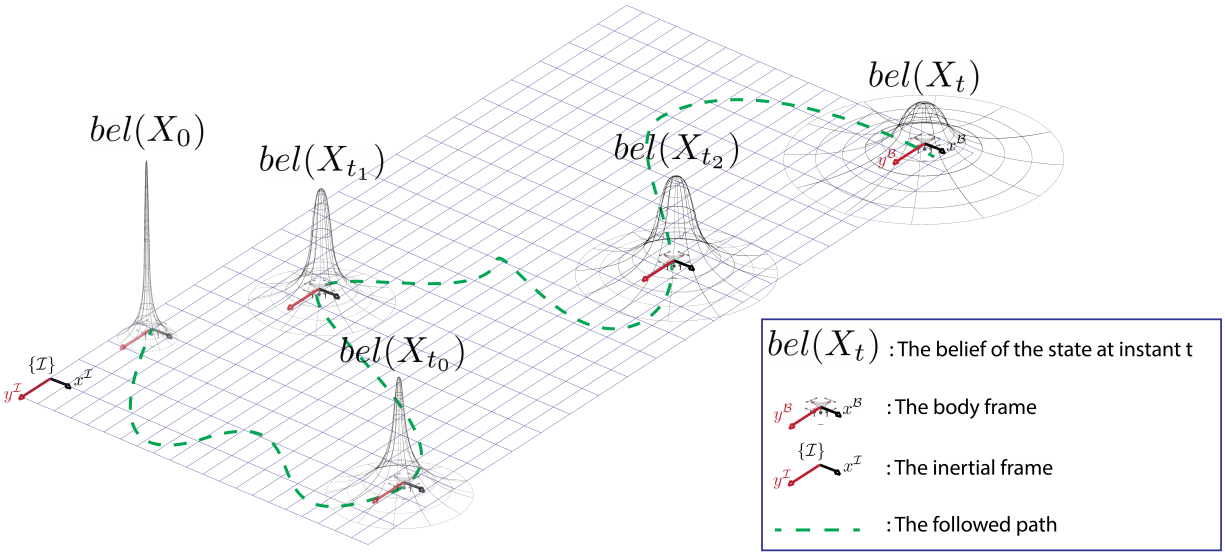


Figure 2.4: State estimation using KF

It is clear from (Fig. 2.4) that the estimation is getting less certain, that is why an accurate initial guess is preferred for the KF to perform well. We also refer to the fact that the KF helps reduce the uncertainty by combining measurements and predictions, where the kalman gain determines the dominance of both the prediction and the update steps. If measurement is very noisy, we find that $\|\Sigma_{\xi_t}\|_F$[3]$\rightarrow +\infty$ which implies that $\|K_t\|_F \rightarrow +\infty$, and then we get $\mu_t \leftarrow \bar{\mu}_t$, and that means that the measurement will be completely ignored. On the other hand, when the prediction is the noisiest, the measurement is the one that will be taken into account instead.

2.4.1.3 Extended Kalman Filter

Introduction

One of the limitations of the KF is the assumption that the mathematical model of the system is linear. However, it's impossible to find this property in reality, since every mathematical model is derived from laws of physics, which are most of the time non-linear, e.g. the drag fluid force used to draw the mathematical model of flying vehicles is relative to square the speed of the moving vehicle [42]. In order to overcome this limitation, the KF is extended to a more flexible algorithm called the *Extended Kalman Filter*, denoted EKF.

In this subsection, we will extend the KF and explain how the EKF estimates the belief of a state, by combining real-time measurements and predictions from a prior state. Also, we will present the mathematical tools needed for the reader to understand it. On the other hand, the mathematical development will not be discussed, though we direct to [41] for more details regarding the theory behind it.

State estimation

As it was mentioned in the introduction, the EKF deals with non-linear systems. It does so by calculating the linearized model around the state to be estimated at each iteration.
First, we will define the non-linear model of any system as :

$$X_t = f(X_{t-1}, U_t) + \epsilon_t \tag{2.16}$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$, while n, m, X_t, U_t, and ϵ_t are defined in the same way as for the KF.

We next will define the non-linear measurement model as :

$$z_t = h(X_t) + \xi_t \tag{2.17}$$

where $h : \mathbb{R}^n \rightarrow \mathbb{R}^p$, while p and ξ_t are defined in the same way as for the KF.
The linearized models of (2.17) and (2.16) will be calculated next to construct the EKF. Since we know that X_{t-1} is normally distributed around a mean state μ_{t-1}, we can cal-

[3]The frobenius norm $\|\cdot\|_F$ is defined for a matrix Σ as $\|\Sigma\|_F = tr(\Sigma^T\Sigma)$

culate the predicted state using first order Taylor series near μ_{t-1} as follows :

$$\begin{aligned} X_t &= f(X_{t-1}, U_t) + \epsilon_t \\ &\approx f(\mu_{t-1}, U_t) + \frac{\partial f}{\partial X_{t-1}}\Big|_{X_{t-1}=\mu_{t-1}} (X_{t-1} - \mu_{t-1}) + \epsilon_t \end{aligned}$$

denoting that :

$$F_t = \frac{\partial f}{\partial X_{t-1}}\Big|_{X_{t-1}=\mu_{t-1}} \tag{2.18}$$

we obtain :

$$X_t \approx f(\mu_{t-1}, U_t) + F_t(X_{t-1} - \mu_{t-1}) + \epsilon_t \tag{2.19}$$

now, we will linearize the observation model (2.17) around the predicted mean $\overline{\mu}$ to get:

$$z_t \approx h(\overline{\mu}_t) + H_t(X_t - \overline{\mu}_t) + \xi_t \tag{2.20}$$

where :

$$H_t = \frac{\partial h}{\partial X_t}\Big|_{X_t=\overline{\mu}_t} \tag{2.21}$$

Like the KF, the arguments of the EKF algorithm are the moment $(\mu_{t-1}, \Sigma_{t-1})$ of the prior state X_{t-1}, some input U_t, and some measurement z_t, while it returns the moment (μ_t, Σ_t) of the current state X_t.

Algorithm 2 Extended_Kalman_Filter$(\mu_{t-1}, \Sigma_{t-1}, U_t, z_t)$

Data: $\mu_{t-1}, \Sigma_{t-1}, U_t, z_t$
Result: μ_t, Σ_t

1 $\bar{\mu} \leftarrow g(\mu_{t-1}, U_t)$
2 Compute F_t and H_t using (2.18) and (2.21)
3 $\bar{\Sigma}_t \leftarrow F_t \Sigma_{t-1} F_t^T + \Sigma_{\epsilon_t}$
4 $K_t \leftarrow \bar{\Sigma}_t C_t^T \left(H_t \bar{\Sigma}_t H_t^T + \Sigma_{\xi_t}\right)^{-1}$
5 $\mu_t \leftarrow \bar{\mu}_t + K_t \left(z_t - h(\overline{\mu}_t)\right)$
6 $\Sigma_t \leftarrow (I - K_t H_t)\bar{\Sigma}_t$

2.4.1.4 2D velocity motion model

In this subsection, we will introduce the motion model of a robot moving with some angular and translational velocities.

Let R, p, $\varpi^{\mathcal{B}}$, $v^{\mathcal{B}}$, $\varpi^{\mathcal{I}}$, and $v^{\mathcal{I}}$ be the rotation of the body frame with respect to the inertial frame, the location of the body frame with respect to the inertial frame, the rotational velocity with respect to the body frame, the translational velocity with respect to the body frame, the rotational velocity with respect to the inertial frame, and the translational velocity with respect to the inertial frame, respectively.

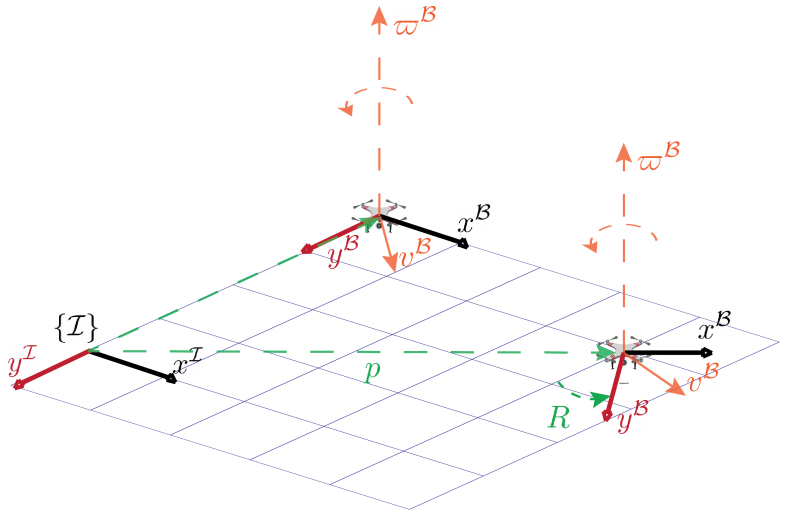


Figure 2.5: Graphical demonstration of the 2D motion model

The velocity motion model (2.22) which we will be using in the EKF SLAM, is mathematically proved in [41].

$$X_t = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} -\frac{\|v^{\mathcal{B}}\|}{\|\varpi^{\mathcal{B}}\|}\sin(\theta) + \frac{\|v^{\mathcal{B}}\|}{\|\varpi^{\mathcal{B}}\|}\sin(\theta + \|\varpi^{\mathcal{B}}\|dt) \\ \frac{\|v^{\mathcal{B}}\|}{\|\varpi^{\mathcal{B}}\|}\cos(\theta) - \frac{\|v^{\mathcal{B}}\|}{\|\varpi^{\mathcal{B}}\|}\cos(\theta + \|\varpi^{\mathcal{B}}\|dt) \\ \|\varpi^{\mathcal{B}}\|dt \end{bmatrix} + \epsilon_t \tag{2.22}$$

where ϵ_t is defined as in (2.16) with n = 3, and $X_t = \begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}^T$

2.4.1.5 2D observation model

In this subsection, we will introduce the observation model used in [41] to measure the pose of a landmark with respect to the initial pose of the body frame. We also provide a graphical demonstration in (Fig. 2.6) to better understand how noise affects the measurement process.

Let $\overline{\mu}_t$, $\overline{\mu}_{t,\theta}$, $\overline{\mu}_j$, and z_t^i be the predicted 2D position of the robot at time t with respect to the initial pose of the body frame, the predicted orientation of the robot at time t with respect to the initial pose of the body frame, the predicted location of the landmark with respect to the initial pose of the body frame, and the range-bearing relative measurement of the landmark j corresponding to the feature i at instant t. Where $z_t^i = (r_t^i, \phi_t^i)$ and r_t^i ,ϕ_t^i are defined to be the polar coordinates of the j^{th} landmark, respectively.

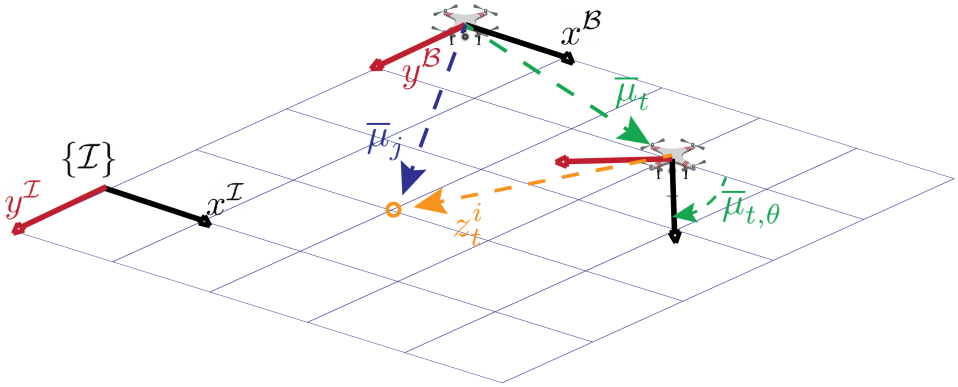


Figure 2.6: Graphical demonstration of landmarks' measurement

Thus, we define the observation model from range bearing measurements as :

$$\begin{bmatrix} \overline{\mu}_{j,x} \\ \overline{\mu}_{j,y} \end{bmatrix} = \begin{bmatrix} \overline{\mu}_{t,x} \\ \overline{\mu}_{t,y} \end{bmatrix} + \begin{bmatrix} r_t^i \cos(\phi_t^i + \overline{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \overline{\mu}_{t,\theta}) \end{bmatrix} \tag{2.23}$$

Using this observation model (2.23), we will introduce a predicted observation, which will be used next in the SLAM algorithm to correct the state of the prediction using the difference between what the robot really observed and what the robot should observe based on the mathematical model. The predicted observation is defined as follows :

$$\hat{z}_t^i = h(\overline{\mu}_t) = \begin{bmatrix} q \\ \text{atan2}(\delta_y, \delta_x) - \overline{\mu}_{t,\theta} \end{bmatrix} \tag{2.24}$$

where :

$$\delta = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \begin{bmatrix} \overline{\mu}_{j,x} - \overline{\mu}_{t,x} \\ \overline{\mu}_{j,y} - \overline{\mu}_{t,y} \end{bmatrix} \tag{2.25}$$

$$q = \sqrt{\delta^T \delta} \tag{2.26}$$

2.4.1.6 EKF SLAM presentation

In order to solve the slam problem using the EKF, we first will define the state vector $X_t \in \mathbb{R}^{2n+3}$ containing the pose of the vehicle, and the location of the landmarks, where :

$$X_t = \begin{bmatrix} x_t & y_t & \theta_t & m_{1,x} & m_{1,y} & \dots & m_{n,x} & m_{n,y} \end{bmatrix}^T \tag{2.27}$$

while we define the predicted state vector as :

$$\overline{\mu}_t = \begin{bmatrix} \overline{\mu}_{t,x} & \overline{\mu}_{t,y} & \overline{\mu}_{t,\theta} & \overline{\mu}_{1,x} & \overline{\mu}_{2,y} & \dots & \overline{\mu}_{n,x} & \overline{\mu}_{n,y} \end{bmatrix}^T \tag{2.28}$$

We, next, will calculate the Jacobian matrix of (2.24) using (2.21) :

$$H_t^i = \frac{1}{q^2} \begin{bmatrix} q\delta_x & -q\delta_y & 0 & -q\delta_x & q\delta_y \\ \delta_y & \delta_x & -1 & -\delta_y & -\delta_x \end{bmatrix} L_{x,j} \tag{2.29}$$

where :

$$L_{x,j} = \begin{bmatrix} I_3 & 0_{3\times(2j-2)} & 0_{3\times 2} & 0_{3\times(2n-2j)} \\ 0_{2\times 3} & 0_{2\times(2j-2)} & I_2 & 0_{2\times(2n-2j)} \end{bmatrix} \tag{2.30}$$

We will next calculate the Jacobian of the motion model (2.22) using (2.18) :

$$F_t = I_{2n+3} + L_x^T \begin{bmatrix} 0 & 0 & \frac{\|v^{\mathcal{B}}\|}{\|\varpi^{\mathcal{B}}\|}\cos(\theta) - \frac{\|v^{\mathcal{B}}\|}{\|\varpi^{\mathcal{B}}\|}\cos(\theta + \|\varpi^{\mathcal{B}}\|dt) \\ 0 & 0 & \frac{\|v^{\mathcal{B}}\|}{\|\varpi^{\mathcal{B}}\|}\sin(\theta) - \frac{\|v^{\mathcal{B}}\|}{\|\varpi^{\mathcal{B}}\|}\sin(\theta + \|\varpi^{\mathcal{B}}\|dt) \\ 0 & 0 & 0 \end{bmatrix} L_x \tag{2.31}$$

where L_x is defined as :

$$L_x = \begin{bmatrix} I_3 & 0_{3\times 2n} \end{bmatrix} \tag{2.32}$$

ϵ_t and ξ_t are defined in the same way as in (2.13) and (2.14), respectively.

Even though the mathematical development in this chapter is not rigorous enough, we redirect the reader to [41] for more details about the derivation of the EKF-SLAM presented below :

Algorithm 3 EKF_SLAM($\mu_{t-1}, \Sigma_{t-1}, U_t, z_t$)

Data: $\mu_{t-1}, \Sigma_{t-1}, U_t, z_t, n$
Result: μ_t, Σ_t

Define L_x as in (2.32)
$\overline{\mu}_t \leftarrow \mu_t + L_x^T \begin{bmatrix} -\frac{\|v^{\mathcal{B}}\|}{\|\varpi^{\mathcal{B}}\|}\sin(\theta) + \frac{\|v^{\mathcal{B}}\|}{\|\varpi^{\mathcal{B}}\|}\sin(\theta + \|\varpi^{\mathcal{B}}\|dt) \\ \frac{\|v^{\mathcal{B}}\|}{\|\varpi^{\mathcal{B}}\|}\cos(\theta) - \frac{\|v^{\mathcal{B}}\|}{\|\varpi^{\mathcal{B}}\|}\cos(\theta + \|\varpi^{\mathcal{B}}\|dt) \\ \|\varpi^{\mathcal{B}}\|dt \end{bmatrix}$
Define F_t as in (2.31)
Define Σ_{ϵ_t} and Σ_{ξ_t} as in (2.14)
$\overline{\Sigma}_t \leftarrow F_t \Sigma_{t-1} F_t^T + L_x^T \Sigma_{\epsilon_t} L_x$
while *i in observed features* **do**
  if *i is new* **then**
    $n \leftarrow n + 1$
    $\begin{bmatrix} \overline{\mu}_{n,x} \\ \overline{\mu}_{n,y} \end{bmatrix} = \begin{bmatrix} \overline{\mu}_{t,x} \\ \overline{\mu}_{t,y} \end{bmatrix} + \begin{bmatrix} r_t^i \cos(\phi_t^i + \overline{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \overline{\mu}_{t,\theta}) \end{bmatrix}$
  Compute δ and, q as in (2.25) and (2.26), respectively.
  Compute $\hat{z}_t^i$ as in (2.24)
  Compute $L_{x,i}$ as in (2.30)
  Compute H_t^i as in (2.29)
  $K_t^i \leftarrow \overline{\Sigma}_t {H_t^i}^T (H_t^i \overline{\Sigma}_t {H_t^i}^T + \Sigma_{\xi_t})^{-1}$
$\mu_t \leftarrow \overline{\mu}_t + \sum_i K_t^i (z_t^i - \hat{z}_t^i)$
$\Sigma_t = \left(I_{sizeof(observed\_features)} - \sum_i K_t^i H_t^i \right) \overline{\Sigma}_t$

We can see from the algorithm 3 that the pose of the robot is corrected using all the measurements of the landmarks in sight, which means that every time an error occurs during the prediction, the Kalman gain will help in reducing the effect using the observation model, while controlling the degree of how much each measurement is taken into consideration based on the noise affecting the measurement sensor.

2.4.2 Pose Graph Optimization based SLAM

[43] and [44] provide a complete, but computationally expensive smoothing-based solution to the SLAM problem. This method doesn't estimate the most recent pose of the robot only, but also helps improve the estimation of the past poses over time. In this approach, these poses are referred to as *nodes*, and are linked together with what are called *edges*, which represent constraints between poses that have to be respected to correctly improve the estimation of the motion path in the optimization phase. As a result, we find ourselves facing a problem where the state of the robot is a state vector with a large dimension and which contains some selected poses of the robot to approximate the real trajectory.

Throughout this subsection, we will explain the pose graph optimization SLAM. Also, why this method is computationally expensive. Again, we note that the mathematical derivation will not be fully explained in this work, and we direct to [43] for more details.

First, we will define some mathematical tools needed to understand the theory behind this approach. Then, we will explain how the pose graph optimization algorithm works. And finally, how and when should the optimization take place.

2.4.2.1 Mathematical definitions

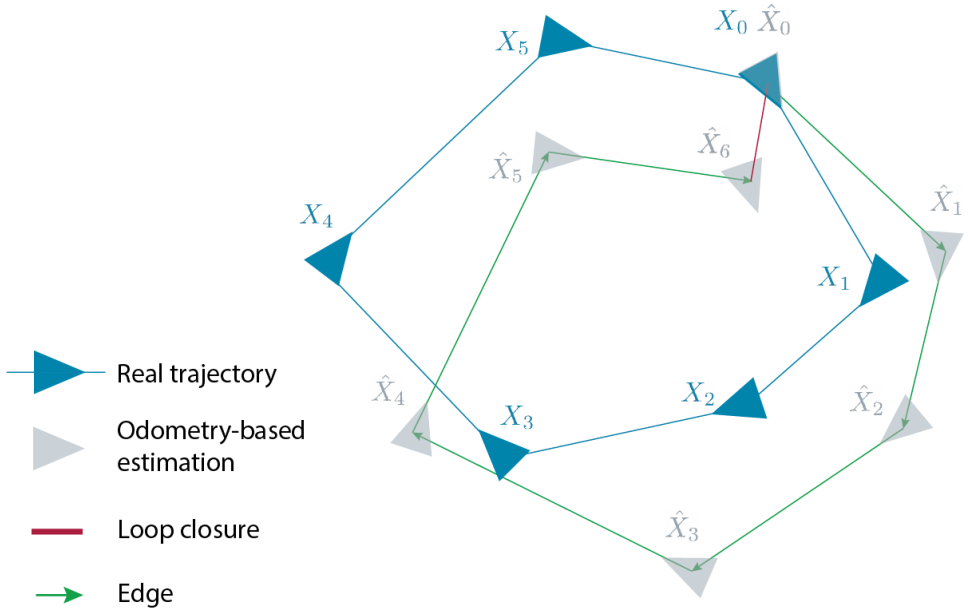


Figure 2.7: Odometry-based estimation

(Fig.2.7) shows the path of a robot generated over time in blue, and its estimation using the dead-reckoning method. Clearly, odometry alone is not sufficient to accurately estimate the robot's trajectory, as it usually drift over time. Thus, we rely on loop closure (illustrated in (Fig. 2.8)) to do a closed loop estimation. When loop closure takes place, the graph optimization algorithm runs to help improve the estimation of all poses in the state vector.

We define the state vector as :

$$X_{0:t} = \begin{bmatrix} X_0^T & X_1^T & \dots & X_t^T \end{bmatrix}^T \tag{2.33}$$

where $X_i \in \mathbb{R}^3$ is the i^{th} pose of the robot, which contains the x_i and y_i coordinates of the robot, as well as the orientation of the robot denoted θ_i

We call a pose of the robot a node, and the relation between two nodes an edge. (Fig. 2.7) shows a graph with six nodes, five edges, and one loop closure constraint. Also, we refer to the fact that a Loop closure constraint is introduced to the optimization problem when the exact same measurement is made twice, as explained in (Fig. 2.8) :

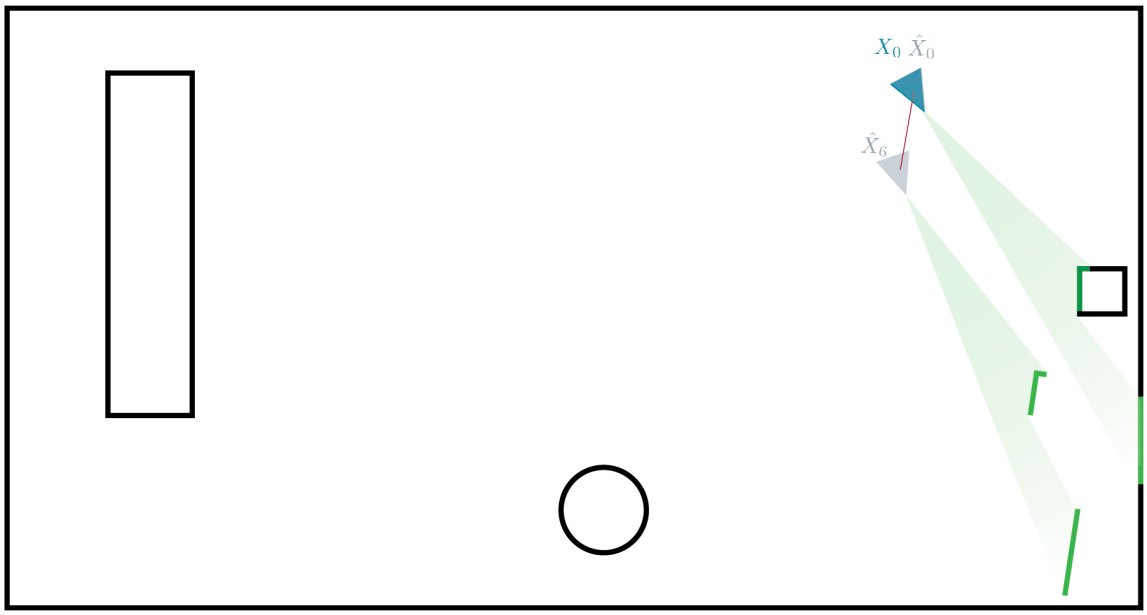


Figure 2.8: Loop closure detection

(Fig. 2.8) shows how the loop closure constraint in (Fig. 2.7) is formed using a measurement taken from X_0 and the most recent measurement taken from X_6. The front-end algorithm continues to compare all previous measurements to the most recent one, and the moment it finds two identical measurements, it concludes that the most recent pose and the pose with the similar measurement are the same. Thus, it includes a loop-closure constraint to the optimization problem between the two estimated poses.

In order to find the best estimate of the state vector $X_{0:n}$ containing n nodes, we will opt for maximizing the normal probability distribution of $X_{0:n}$ given the commands $U = \{U_{0:n}, U_{ij} | (i,j) \in \mathcal{LC}\}$, defined as in (2.12), since we assume that the noise affecting the motion model is Gaussian. And using the Markov assumption [45], we conclude that

$$P(X_{0:n}|U) = \prod_{i=1}^{n} P(X_i|X_{i-1}, U_i) \prod_{i,j \in \mathcal{LC}} P(X_j|X_i, U_{ij}) \tag{2.34}$$

where $\mathcal{LC}$ is a set of all loop closure pairs. In order to find the state with the highest probability, we have to compute the state that maximizes (2.35). We define the set $\mathcal{Q}$ of all constraints, including edges and loop closure constraints, to obtain.

$$P(X_{0:n}|U) = \prod_{i,j \in \mathcal{Q}} P(X_j|X_i, U_{ij}) \tag{2.35}$$

2.4.2.2 Objective function optimization

Next, we will prove that maximizing (2.35) can be brought down to a least-square optimization problem [44], where edges and loop closure constraints are defined as the error between states.

$$\begin{aligned} X^*_{0:n} &= \text{argmax } (P(X_{0:n}|U_{0:n})) \\ &= \text{argmin } (-\log(P(X_{0:n}|U_{0:n}))) \\ &= \text{argmin } \left(\sum_{i,j \in \mathcal{Q}} \|X_j - f(X_i, U_{i,j})\|^2_{\Sigma_{\epsilon_t}} \right) \end{aligned}$$

We will use the following objective function, which we denote $J(X_{0:n})$, to mathematically derive the PGO algorithm as in [43], and we put $J_{ij}(X_{0:n}) = \|X_j - f(X_i, U_{i,j})\|^2_{\Sigma_{\epsilon_t}}$

$$J(X_{0:n}) = \sum_{i,j \in \mathcal{Q}} J_{ij}(X_{0:n}) \tag{2.36}$$

Now we will use *Gauss-newton method* [46] to derive an iterative PGO algorithm to compute the local minimum of the objective function, starting from an initial guess $\hat{X}_{0:n}$. We put $e_{ij}(\hat{X}_{0:n}) = \hat{X}_j - f(\hat{X}_i, U_{i,j})$.

$$\begin{aligned} J_{ij}(\hat{X}_{0:n} + \Delta X_{0:n}) &= e_{ij}^T(\hat{X}_{0:n} + \Delta X_{0:n}) \Sigma_{\epsilon_t}^{-1} e_{ij}(\hat{X}_{0:n} + \Delta X_{0:n}) \\ &\approx \left(e_{ij}^T(\hat{X}_{0:n}) + \Delta X_{0:n}^T \frac{\partial e_{ij}}{\partial X_{0:n}}^T \right) \Sigma_{\epsilon_t}^{-1} \left(e_{ij}(\hat{X}_{0:n}) + \frac{\partial e_{ij}}{\partial X_{0:n}} \Delta X_{0:n} \right) \\ &= e_{ij}^T(\hat{X}_{0:n}) \Sigma_{\epsilon_t}^{-1} e_{ij}(\hat{X}_{0:n}) + 2\mathcal{B}_{ij} \Delta X_{0:n} + \Delta X_{0:n}^T \mathcal{H}_{ij} \Delta X_{0:n} \\ &= J_{ij}(\hat{X}_{0:n}) + 2\mathcal{B}_{ij} \Delta X_{0:n} + \Delta X_{0:n}^T \mathcal{H}_{ij} \Delta X_{0:n} \end{aligned}$$

where $\mathcal{H}_{ij} = \frac{\partial e_{ij}}{\partial X_{0:n}}^T \Sigma_{\epsilon_t}^{-1} \frac{\partial e_{ij}}{\partial X_{0:n}}$, and $\mathcal{B}_{ij} = e_{ij}^T(\hat{X}_{0:n}) \Sigma_{\epsilon_t}^{-1} \frac{\partial e_{ij}}{\partial X_{0:n}}$. From (2.36) we get

$$\begin{aligned} J(\hat{X}_{0:n} + \Delta X_{0:n}) &= \sum_{i,j \in \mathcal{Q}} \left(J_{ij}(\hat{X}_{0:n}) + 2\mathcal{B}_{ij} \Delta X_{0:n} + \Delta X_{0:n}^T \mathcal{H}_{ij} \Delta X_{0:n} \right) \\ &= \sum_{i,j \in \mathcal{Q}} J_{ij}(\hat{X}_{0:n}) + 2 \sum_{i,j \in \mathcal{Q}} \mathcal{B}_{ij} \Delta X_{0:n} + \Delta X_{0:n}^T \sum_{i,j \in \mathcal{Q}} \mathcal{H}_{ij} \Delta X_{0:n} \\ &= J(\hat{X}_{0:n}) + 2\mathcal{B} \Delta X_{0:n} + \Delta X_{0:n}^T \mathcal{H} \Delta X_{0:n} \end{aligned}$$

we set $\frac{\partial J(X_{0:n} + \Delta X_{0:n})}{\partial \Delta X_{0:n}} = 0$, and we get the linear system to be solved (2.37) at each iteration until (2.36) reaches its minimum at $X^*_{0:n}$:

$$\mathcal{H} \Delta X_{0:n} = -\mathcal{B} \tag{2.37}$$

2.4.2.3 PGO SLAM presentation

In this section, we present the PGO SLAM algorithm as a pseudocode :

Algorithm 4 PGO_SLAM($X^*_{0:n}$)

Data: $\Sigma_{\epsilon_t}, U_{ij}$
Result: $X^*_{0:n}$

Initialize $X^*_{0:n}$
while *!converged* **do**
  $\mathcal{H} \leftarrow 0$
  $\mathcal{B} \leftarrow 0$
  for $i, j \in \mathcal{Q}$ **do**
    $\mathcal{B} \leftarrow \mathcal{B} + e_{ij}^T(\hat{X}_{0:n})\Sigma_{\epsilon_t}^{-1}\dfrac{\partial e_{ij}}{\partial X_{0:n}}$
    $\mathcal{H} \leftarrow \mathcal{H} + \dfrac{\partial e_{ij}}{\partial X_{0:n}}^T \Sigma_{\epsilon_t}^{-1}\dfrac{\partial e_{ij}}{\partial X_{0:n}}$
  Compute $\Delta X_{0:n}$ as in (2.37);
  $X^*_{0:n} \leftarrow X^*_{0:n} + \Delta X_{0:n}$

2.5 Map representations

2.5.1 Sparse map representation (landmark-based)

This type of representation had been widely used in *Simultaneous Localization And Mapping* for mobile robotics, to localize a robot's pose, while estimating the location of 3-D landmarks to create the map. The majority of SLAM solutions use this type of map representation to represent the environment as a set of 3-D landmarks [9], which describe distinguishable features in the real world (e.g.trees and corners) ([1], [47]–[49]). In order to map the environment correctly, the measurement sensor (e.g. Camera) has to provide a *descriptor* to define what landmark corresponds to some made measurement. However, It is not possible to perfectly match measurements with their corresponding landmarks, to solve that problem, several outliers rejection methods had been proposed to make a more robust *feature matching* algorithm ([50]–[52]), to come up with a better landmark-based representation.

2.5.2 Dense map representation (dense cloud points)

Unlike sparse map representation, the dense representation provides a higher map resolution of the real world, so instead of representing only the location of a landmark, it represents all of its geometric properties (e.g. a tree is represented as a point cloud in the shape of the observed tree). This type of representation is widely used to solve the SLAM problem, when obstacle avoidance is more needed than navigation ([53]–[56]). In order to create a dense representation of the environment, sensors which can provide point cloud measurements of the environment like *RGB-D* cameras, stereo cameras, and LASER 3D scanners are mandatory. Due to the high resolution of the map, this type of representation requires a large size of memory.

2.5.3 Grid based map representation (occupancy grid)

Occupancy mapping refers to the family of algorithms that helps in representing the 2-D environment as a grid, where occupied spaces are represented with black squares, to indicate the existence of obstacles, while unoccupied spaces are represented in white. When the sensor returns noisy measurements, it represents the measurement in dark gray if the probability is high, or light gray if the probability is low. This type of representation is used to map the environment using wide range sonar in [57], and using a LIDAR in [58]. A novel approach proposed in [59] that uses reinforcement learning, to swap between two representations, grid occupancy and landmark-based, to increase the robustness of the map during autonomous navigation, and enhance the accuracy of pose estimation.

2.6 Conclusion

In this chapter, we first gave the reader a glimpse of the *history of SLAM*, where we focused on showing the gradual increasing need for *Simultaneous Localization And Mapping* over time, to replace incomplete techniques like *odometry*. In the section that follows, we explained how a SLAM method is made of two components, the *frontend* and the *backend* components, and presented the main role of each one. Subsequently, we tried to draw the reader's attention to how vast the SLAM problem can be, and we presented two possible classifications of SLAM (*LIDAR-SLAM* vs *V-SLAM* and *2D SLAM* vs *3D SLAM*), and insisted on the advantages and disadvantages of each category of SLAM. Then, we presented two approaches to solving the SLAM problem, a filtering approach (*EKF-SLAM*) and a smoothing approach (*PGO SLAM*). Finally, we presented three types of map representations, along with the strengths of each one. The overview of SLAM presented in this chapter was purposely light and shallow, as this is not the main focus of our work, and a more detailed overview of SLAM would've taken many more pages to present. However, we tried to provide some references in the different topics that we introduced in order to fill the gaps of the chapter.

Chapter 3

Lie groups for Robotics

3.1 The power of Lie theory

By the end of the 19^{th} century, the Norwegian mathematician *Sophus Lie* gave birth to a theory for continuous transformation groups, namely, **Lie groups**. For being highly abstract concepts, they have been reserved for the most seasoned mathematicians or served as a partially understood tool for chemists and physicists for a considerable amount of time [60]. It is only during the last couple of years that efforts have been made to break down the level of abstraction of Lie theory and bring it within the reach of researchers in various fields of study and even undergraduate students. To our knowledge, two of the most significant contributions towards this direction are the *"Naive Lie Theory"* book by John Stillwell [60], and the *"Very Basic Lie Theory"* article by Roger Howe [61]. The idea that the structure of a nonlinear space (the manifold of the Lie group) can be almost entirely recovered from a linear vector space (the Lie algebra) and a binary operation (the Lie bracket) has caught the interest of many fields, especially where state estimation is needed (e.g. in robotics). Therefore, several efforts have been made to bring Lie theory even closer to the robotics community and to make it easier to use in solving state estimation problems such as SLAM ([7], [62]–[64]). To do so, and because state estimation only requires a subset of Lie theory, these works intentionally omitted a large part of it in order to help roboticists focus only on the essential.

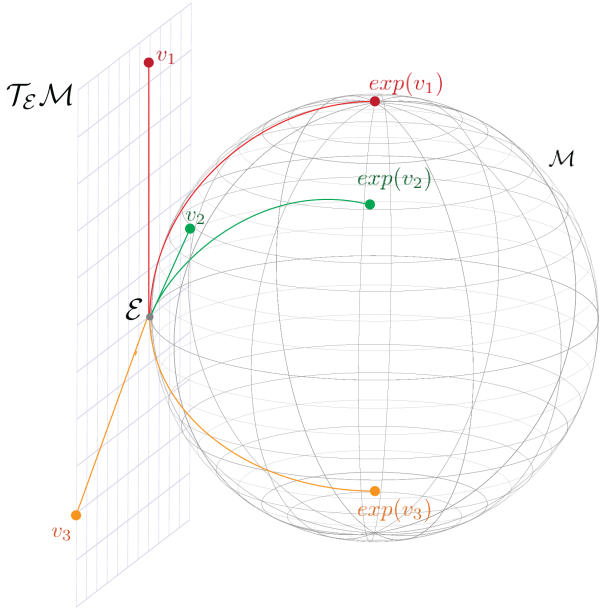


Figure 3.1: An intuitive representation of the relationship between the Manifold $\mathcal{M}$ and its Lie algebra $T_\mathcal{E}\mathcal{M}$

In this chapter, we will be even more selective than the above-mentioned works in choosing what to introduce from Lie theory, since we are more interested in reducing the learning curve of the reader rather than leveraging the full potential of this theory. We will try to tackle the subject from a modest roboticist point of view and introduce the must-know concepts of Lie theory in an intuitive way, and only focus on Matrix Lie groups that are used in robotics to represent orientations and poses of rigid bodies. The chapter itself should be very light in content, as we want it to be pleasant to read so that we can draw the reader's attention to what we believe to be the real power of Lie theory, which is

perfectly described by Roger Howe in [61] when he talked about the source of power of Lie theory : *"The essential phenomenon of Lie theory is that one may associate in a natural way to a Lie group $\mathcal{G}$ its Lie algebra $\mathfrak{g}$. The Lie algebra $\mathfrak{g}$ is first of all a vector space and secondly is endowed with a bilinear nonassociative product called the Lie bracket [...]. Amazingly, the group $\mathcal{G}$ is almost completely determined by $\mathfrak{g}$ and its Lie bracket. Thus, for many purposes, one can replace $\mathcal{G}$ with $\mathfrak{g}$. Since $\mathcal{G}$ is a complicated nonlinear object and $\mathfrak{g}$ is just a vector space, it is usually vastly simpler to work with $\mathfrak{g}$. [...] This is one source of the power of Lie theory"*. Meaning that we can leverage the exact relationship between the nonlinear Manifold defining the elements of the Lie group and its respective linear Lie algebras by moving back and forth between the two spaces using the exponential and logarithm maps (see Fig. 3.1) to do calculus and other mathematical operations on the Lie algebra, and express the final results on the nonlinear manifold. The argument just stated, together with the fact that Matrix Lie groups are very handy for modeling states and measurements in one bundle, illustrates how one can precisely and effectively model complex nonlinear dynamics and design nonlinear observers that imitate the true nature of the problem to be optimized (e.g. the SLAM problem) using Lie theory.

The material of this chapter is organized as follows : first, we introduce a Lie group as a regular group endowed with a smooth manifold, and we define the properties of both concepts. Afterwards, we define the must-know concepts of Lie theory and their properties, while giving practical examples of their application in robotics when appropriate. Finally, we will end by introducing two of the most popular Lie groups in robotics, namely, the special orthogonal group $SO(3)$[1], and the special euclidean group $SE(3)$[2] and we will define all the already-introduced concepts in these two groups.

3.2 What is a Lie group?

A very popular and yet very abstract definition of a Lie group states that a Lie group is a regular group which is also a smooth manifold. To understand this definition, we first need to know what is considered as a group? and what is a manifold defined by?
First, a group is a combination of a set $\mathcal{G}$ and a composition operation $\circ$, denoted $(\mathcal{G}, \circ)$. For elements of the group $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \in \mathcal{G}$, a valid group must satisfy the following axioms :

1. Closure under '$\circ$' : $\mathcal{X} \circ \mathcal{Y} \in \mathcal{G}$
2. Identity $\mathcal{E}$: $\mathcal{E} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{E} = \mathcal{X}$
3. Inverse $\mathcal{X}^{-1}$: $\mathcal{X}^{-1} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{X}^{-1} = \mathcal{E}$
4. Associativity : $(\mathcal{X} \circ \mathcal{Y}) \circ \mathcal{Z} = \mathcal{X} \circ (\mathcal{Y} \circ \mathcal{Z})$

meaning that (1) all the compositions of elements of the group remain in the group, (2) every element of the group has an inverse element that also belongs to the group, (3) one of the elements of the group is the identity element, and (4) is a basic associativity rule.

Second, a manifold is a topologically curved n-dimensional surface that is constructed by all the elements of the group. And it is said to be *smooth* or *differentiable* when it

[1]The special orthogonal group $SO(3)$ is the group of $3\times3 rotation matrices$.
[2]The special euclidean group $SE(3)$ is the group of rigid-bodies motions.

has a shape that has no spikes or edges. One can visualize a smooth manifold as a 3-dimensional sphere[3] contained in a higher dimension. One important implication of the smoothness of the manifold is the existence of a unique tangent space at each state, which is a vector space that allows calculus. As we will see later in this chapter, the manifold of the group of rotation matrices represents every possible orientation that a vehicle can have, and the tangent space at different states represents the possible angular velocities at that particular state.

Hence, Lie groups join forces of linear algebra and calculus in some way. On the one hand, they allow us to do calculus on tangent spaces, and on the other hand, the global properties of the group allow us to do composition of elements all across the manifold.

3.3 Group actions in robotics

What makes Lie groups so useful in robotics is their ability to act on elements of other sets and transform them. In robotics, for example, some Lie groups can rotate rigid bodies, translate them or even scale them. This ability to act on other sets is defined by the **group action**.
Let $\mathcal{G}$ a Lie group, $\mathcal{S}$ a set, and let $\mathcal{X}, \mathcal{Y} \in \mathcal{G}, x \in \mathcal{S}$. And let $\langle \cdot \rangle$ be the action of $\mathcal{G}$ on $\mathcal{S}$, such that $\mathcal{X} \cdot x \in \mathcal{S}$, then it must have the following properties to be considered a valid group action :

1. Identity : $\mathcal{E} \cdot x = x$
2. Compatibility : $(\mathcal{X} \circ \mathcal{Y}) \cdot x = \mathcal{X} \cdot (\mathcal{Y} \cdot x) \in \mathcal{S}$

In robotics, the group of rotations and the group of rigid-body motions act on vectors through matrix multiplication to create new vectors (we will see this more in depth when we will introduce these groups).

3.4 Tangent spaces and the Lie algebra

What Stillwell's [60] describes to be *"The miracle of Lie theory"* is the capacity of capturing a curved complex object, i.e. the Lie group, by a flat one, i.e. the tangent space at the identity. An intuitive way to visualize what a tangent space is, is to consider a point $\mathcal{X}(t)$ moving on the manifold of a Lie group $\mathcal{M}$, then its time derivative $\dot{\mathcal{X}}$ represents the velocity of that point, and it belongs to the tangent space [62] (see Fig. 3.2). The tangent space at a state $\mathcal{X}$ is denoted $T_{\mathcal{X}}\mathcal{M}$, and since the manifold of a Lie group is differentiable, there is only one distinct tangent space at each state, although the structure of the tangent space remains the same at every state described by the manifold. For multiplicative groups, the elements of the tangent space are defined as follows :

$$\dot{\mathcal{X}} = \mathcal{X}\varsigma^{\wedge} \ ; \ \mathcal{X} \in \mathcal{G}, \varsigma^{\wedge} \in \mathfrak{g} \tag{3.1}$$

Such result can be obtained by taking the time derivative of the constraint $\mathcal{X}\mathcal{X}^{-1} = \mathcal{E}$

[3]It is important to note that a 3-dimensional sphere is not a Lie group. However, a 4-dimensional hypersphere is. We represent a manifold as a 3D sphere only for the sake of visualization

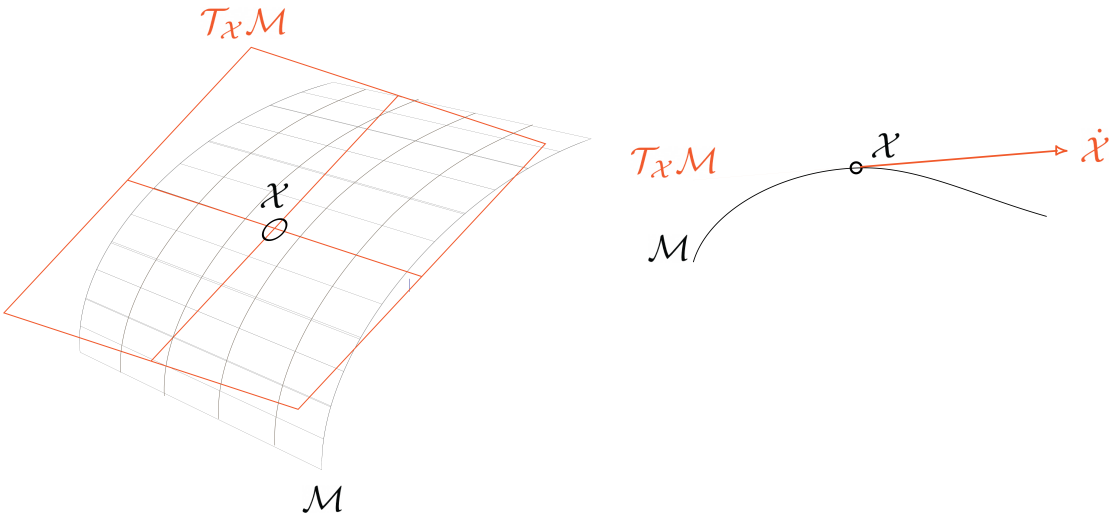


Figure 3.2: An illustration of the tangent space at a state $\mathcal{X}$

We consider the Lie algebra $\mathfrak{g}$ of a Lie group $\mathcal{G}$ to be the tangent space of its manifold $\mathcal{M}$ at the identity element, we also denote it $T_{\mathcal{E}}\mathcal{M}$, and it has the following properties :

1. The Lie algebra $\mathfrak{g}$ of a Lie group $\mathcal{G}$ is the vector space tangent to the manifold of the group at the identity.
2. The dimension of $\mathfrak{g}$, which we note m, is defined by the number of degrees of freedom of $\mathcal{M}$.
3. Since $\mathfrak{g}$ is an m-dimensional vector space, all its elements can be described by a vector in $\mathbb{R}^m$.
4. We can transform elements back and forth between $\mathfrak{g}$ and $\mathcal{G}$ through the *exponential* and the *logarithm* maps.
5. $\mathfrak{g}$ is also closed under a binary operation called the *Lie bracket*.
6. The structure of the Lie algebra $\mathfrak{g}$ can be defined by time differentiating the *inverse* constraint of $\mathcal{G}$: $\mathcal{X} \circ \mathcal{X}^{-1} = \mathcal{E}$
7. Elements of $\mathfrak{g}$ can be thought of as all possible values of $\dot{\mathcal{X}}$ when $\mathcal{X}$ is the identity of $\mathcal{G}$ [65].

3.4.1 The $\vee$ and $\wedge$ operators

Elements of the Lie algebra are sometimes given in complex and tedious-to-manipulate structures[4], but since the Lie algebra is a vector space, and its elements are described by m-vectors, it can be useful to use the *generators* [62] of the Lie algebra and those of the m-dimensional Cartesian space $\mathbb{R}^m$ to move back and forth from $\mathfrak{g}$ and $\mathbb{R}^m$ and vice versa.

[4]Elements of the Lie algebra of rotation matrices are skew symmetric matrices. And the Lie algebra of the homogenous transformation matrices group has an even more complex structure.

Such maps are called *hat* $(\cdot)^\wedge$ and *vee* $(\cdot)^\vee$. Consider $\varsigma \in \mathbb{R}^m$, then, the two maps or *isomorphisms* are given by:

$$Hat : \mathbb{R}^m \to \mathfrak{g}; \qquad \varsigma \mapsto \varsigma^\wedge = \sum_{i=1}^{m} \varsigma_i E_i \tag{3.2}$$

$$Vee : \ \mathfrak{g} \to \mathbb{R}^m; \quad \varsigma^\wedge \mapsto (\varsigma^\wedge)^\vee = \sum_{i=1}^{m} \varsigma_i e_i \tag{3.3}$$

Where E_i are the generators of the Lie algebra and e_i the base of the m-dimensional Cartesian space ($e_i^\wedge = E_i$). In other words, the $\wedge$ operator describes elements of the Lie algebra, and the $\vee$ operator cancels the $\wedge$. These transformations can be very useful since vectors are more convenient to use than the elements of the Lie algebra, whose structure can be complex. For example, we can easily calculate the Jacobian of a manipulator simply by stacking the screw vectors[5] of its joints as the columns of the Jacobian, however it is necessary to have their counterparts in the Lie algebra to calculate the forward kinematics of the manipulator.

3.5 The exponential and logarithm maps

We define the matrix exponential and the matrix logarithm by their respective infinite series for square matrices, with $\|A\| < 1$:

$$\exp(A) = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + ... \tag{3.4}$$

$$\log(I + A) = A - \frac{A^2}{2} + \frac{A^3}{3} - \frac{A^4}{4} + ... \tag{3.5}$$

These two matrix functions play a central role in Lie theory for matrix Lie groups, as they allow integrating elements of the Lie algebra to the manifold, and vice versa. The matrix exponential appears when integrating the ODE (3.1), which gives :

$$\mathcal{X}(t) = \mathcal{X}(0) \exp(\varsigma^\wedge t) \tag{3.6}$$

Since both $\mathcal{X}(t)$ and $\mathcal{X}(0)$ are in $\mathcal{G}$, then $\exp(\varsigma^\wedge t)$ must be in $\mathcal{G}$. And since $\varsigma^\wedge$ is in $\mathfrak{g}$, we say that the exponential maps elements from $\mathfrak{g}$ to elements from $\mathcal{G}$. Also, if $\varsigma^\wedge$ is close enough to zero, then :

$$\log(\exp(\varsigma^\wedge)) = \varsigma^\wedge \tag{3.7}$$

We can easily verify this by expanding the infinite series of both functions. From this, we infer the fact that the logarithm map is the inverse function of the exponential map. Hence, we can write :

$$\exp : \mathfrak{g} \to \mathcal{G}; \qquad \varsigma^\wedge \mapsto \mathcal{X} = \exp(\varsigma^\wedge) \tag{3.8}$$

$$\log : \mathcal{G} \to \mathfrak{g}; \qquad \mathcal{X} \mapsto \varsigma^\wedge = \log(\mathcal{X}) \tag{3.9}$$

[5]Screws are unitary twists, which are the 6-vectors that define the Lie algebra of the homogenous transformation group.

It is worth mention again that the exponential map is a general concept in Lie theory and that the matrix exponential is the same as the exponential map in our case because we are only interested in matrix Lie groups. Otherwise, the matrix exponential and the exponential map represent two different things [66].

Furthermore, consider $A, B \in \mathfrak{g}$ and $\mathcal{X} \in \mathcal{G}$, then the matrix exponential satisfies the following properties :

1. $\frac{d}{dt}\exp(At) = A\exp(At) = \exp(At)A$
2. $\exp(A)\exp(B) = \exp(A+B) \iff AB = BA$
3. $\exp(iA)\exp(jA) = \exp((i+j)A) \quad ; \quad i, j \in \mathbb{R}$
4. $\exp(A)^t = \exp(At)$
5. $\exp(A)^{-1} = \exp(-A)$
6. $\exp(\mathcal{X}A\mathcal{X}^{-1}) = \mathcal{X}\exp(A)\mathcal{X}^{-1}$

We also consider a vectorized version of the exponential and the logarithm matrices, which are used to map vectors from $\mathbb{R}^m$ directly to the group (see 3.3), and which we denote Exp and Log, respectively :

$$Exp : \mathbb{R}^m \to \mathcal{G}; \qquad \varsigma \mapsto \mathcal{X} = Exp(\varsigma) \tag{3.10}$$

$$Log : \mathcal{G} \to \mathbb{R}^m; \qquad \mathcal{X} \mapsto \varsigma = Log(\mathcal{X}) \tag{3.11}$$

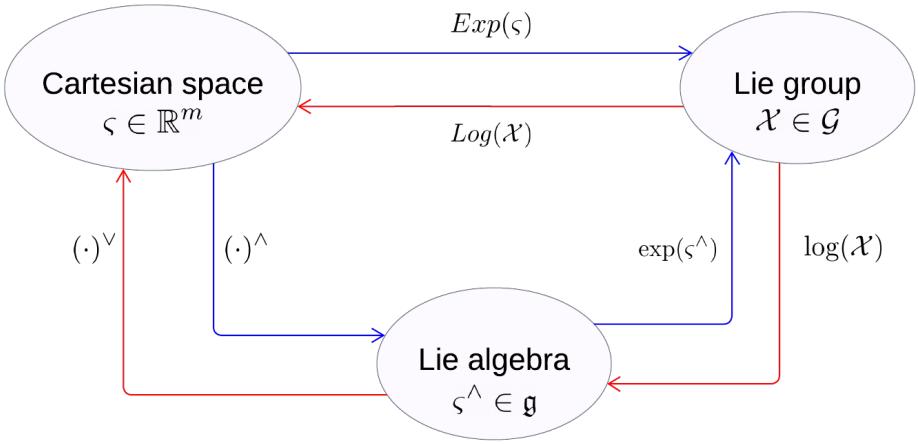


Figure 3.3: The mapping of the different spaces of a Lie group

3.6 The Lie bracket

As we mentioned earlier, the true power of Lie theory lies in the ability of capturing the complex structure of the manifold defining the Lie group $\mathcal{G}$, only by looking at

the properties of its Lie algebra $\mathfrak{g}$, and this is only possible because $\mathfrak{g}$ is closed under the *Lie bracket* operation. As we saw in earlier sections, considering $\mathbb{X}, \mathbb{Y} \in \mathfrak{g}$, then $\exp(\mathbb{X})\exp(\mathbb{Y}) = \exp(\mathbb{X}+\mathbb{Y})$ if and only if $\mathbb{X}\mathbb{Y} = \mathbb{Y}\mathbb{X}$. This means that the *sum* operation only captures the structure of groups endowed with commutative operations. However, as demonstrated in the Baker–Campbell–Hausdorff formula presented below, the behavior of non-commutative groups can only be inferred by the Lie bracket (which is always zero on the tangent space of a commutative group [60]).

The Lie bracket of two elements $\mathbb{X}, \mathbb{Y} \in \mathfrak{g}$ is denoted $[\mathbb{X}, \mathbb{Y}]$ and is given by :

$$[\mathbb{X}, \mathbb{Y}] = \mathbb{X}\mathbb{Y} - \mathbb{Y}\mathbb{X} \tag{3.12}$$

Considering $\mathbb{X}, \mathbb{Y}, \mathbb{Z} \in \mathfrak{g}$ and $a, b \in \mathbb{R}$, then the elements of the Lie algebra under its Lie bracket must satisfy the following identities [61], [63]:

- closure : $[\mathbb{X}, \mathbb{Y}] \in \mathfrak{g}$
- bilinearity : $[a\mathbb{X} + b\mathbb{Y}, Z] = a[\mathbb{X}, \mathbb{Z}] + b[\mathbb{Y}, \mathbb{Z}]; [\mathbb{Z}, a\mathbb{X} + b\mathbb{Y}] = a[\mathbb{Z}, \mathbb{X}] + b[\mathbb{Z}, \mathbb{Y}]$
- skew symmetry : $[\mathbb{X}, \mathbb{Y}] = -[\mathbb{Y}, \mathbb{X}]$
- alternating : $[\mathbb{X}, \mathbb{X}] = 0$
- Jacobi identity : $[\mathbb{X}, [\mathbb{Y}, \mathbb{Z}]] + [\mathbb{Y}, [\mathbb{Z}, \mathbb{X}]] + [\mathbb{Z}, [\mathbb{X}, \mathbb{Y}]] = 0$

3.6.1 The Baker–Campbell–Hausdorff Theorem

There is a very interesting relationship between the Lie bracket and the exponential map defined by the Baker–Campbell–Hausdorff formula that states that we can infer the product of two elements in $\mathcal{G}$ which are close enough to the identity, by the Lie bracket of two elements in $\mathfrak{g}$. In other words, we can determine the value of $\mathbb{Z} = \log(\exp(\mathbb{X})\exp(\mathbb{Y})) \in \mathfrak{g}$ such that $\mathbb{Z}$ is only expressed by $\mathbb{X}, \mathbb{Y}$ and their Lie bracket. We can prove this by expanding the logarithm and the two exponential terms into their Taylor series :

$$\exp(\mathbb{X}) = I + \mathbb{X} + \frac{\mathbb{X}^2}{2!} + \frac{\mathbb{X}^3}{3!} + ... \tag{3.13}$$

$$\exp(\mathbb{Y}) = I + \mathbb{Y} + \frac{\mathbb{Y}^2}{2!} + \frac{\mathbb{Y}^3}{3!} + ... \tag{3.14}$$

$$\log(I + \Psi) = \Psi - \frac{\Psi^2}{2} + \frac{\Psi^3}{3} - \frac{\Psi^4}{4} + ... \tag{3.15}$$

By only considering terms of second order and below :

$$\begin{aligned}
\mathbb{Z} &= log(\exp(\mathbb{X})\exp(\mathbb{Y})) \\
&= log((I + \mathbb{X} + \frac{\mathbb{X}^2}{2!} + ...)(I + \mathbb{Y} + \frac{\mathbb{Y}^2}{2!} + ...)) \\
&= log(I + \underbrace{\mathbb{X} + \mathbb{Y} + \mathbb{X}\mathbb{Y} + \frac{\mathbb{X}^2}{2!} + \frac{\mathbb{Y}^2}{2!} + ...}_{\Psi}) \\
&= (\mathbb{X} + \mathbb{Y} + \mathbb{X}\mathbb{Y} + \frac{\mathbb{X}^2}{2!} + \frac{\mathbb{Y}^2}{2!} + ...) - \frac{1}{2}(\mathbb{X} + \mathbb{Y} + \mathbb{X}\mathbb{Y} + \frac{\mathbb{X}^2}{2!} + \frac{\mathbb{Y}^2}{2!} + ...)^2 \\
&= (\mathbb{X} + \mathbb{Y} + \mathbb{X}\mathbb{Y} + \frac{\mathbb{X}^2}{2!} + \frac{\mathbb{Y}^2}{2!} + ...) - \frac{1}{2}(\mathbb{X}^2 + \mathbb{Y}^2 + \mathbb{X}\mathbb{Y} + \mathbb{Y}\mathbb{X} + ...) \\
&= \mathbb{X} + \mathbb{Y} + \frac{1}{2}(\underbrace{\mathbb{X}\mathbb{Y} - \mathbb{Y}\mathbb{X}}_{[\mathbb{X},\mathbb{Y}]} + ...)
\end{aligned}$$

Therefore :

$$\mathbb{Z} = \mathbb{X} + \mathbb{Y} + \frac{1}{2}[\mathbb{X}, \mathbb{Y}] + \xi \tag{3.16}$$

Which completes the proof up to the second order terms, with ξ containing higher order terms of the Taylor series.

In our demonstration, we just considered terms up to the second order, however it is important to know that the Lie bracket operation is enough to express all the higher order terms of $\mathbb{Z}$. We will not prove this point, but we redirect readers who are curious about this to Eichler's demonstration, which is way lighter to digest than the one proposed by Baker, Campbell and Hausdorff. Also, [66] expanded the Taylor series to the 4^{th} order, and [61] showed that the remaining term ξ was upper bounded by $65(\|\mathbb{X}\| + \|\mathbb{Y}\|)^3$.

A practical use of such a result is the ability to generate smooth trajectories (we will see this in more details when we introduce the special euclidean group $SE(3)$).

3.7 From mathematical abstraction to the real world

Until now, we introduced Lie theory as a set of pure mathematical concepts, even though we tried to break down the level of abstraction the best we could. However, the only way to effectively and correctly use the tools that Lie theory provides is by understanding their physical meaning. For example, it is because we specifically know how elements of $SE(3)$ describe orientations and positions and how elements of its Lie algebra represent linear and angular velocities that we can understand how to use them properly to solve robotics problems.

In this section, we will introduce two of the most popular matrix Lie groups in robotics, *the special orthogonal group* $SO(3)$, and *the special euclidean group* $SE(3)$. And we will try to describe all their properties from a geometric point of view (orientations and positions), as well as from a mechanical point of view (displacements and velocities), so that the reader can build a practical intuition about the usefulness of these two groups.

3.7.1 The special orthogonal group $SO(3)$

Suppose we want to describe the orientation of a vehicle with respect to an inertial frame, which we denote $\{\mathcal{I}\}$. One convenient way to do that in 3-dimensional space is by attaching a frame that describes the position and orientation of the vehicle at any time, which we call "body frame $\{\mathcal{B}\}$", and then express all its unitary vectors in $\{\mathcal{I}\}$. Say for example that our body frame is initially perfectly aligned with the inertial frame, then we perform a rotation around the $\hat{x}_\mathcal{B}$ axis by an angle of θ, followed by a rotation around $\hat{z}_\mathcal{B}$ by φ (see Fig. 3.4).

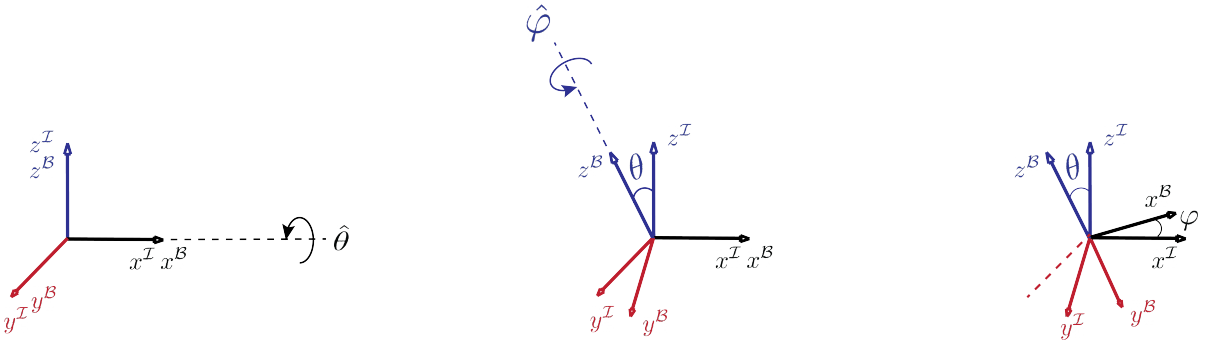


Figure 3.4: Rotation of a body frame with respect to an inertial frame.

Then we would express the vectors of $\{\mathcal{B}\}$ as follows :

$$\hat{x}_\mathcal{B} = \cos\varphi \;\; \hat{x}_\mathcal{I} + \sin\varphi\cos\theta \;\; \hat{y}_\mathcal{I} + \sin\varphi\sin\theta \;\; \hat{z}_\mathcal{I} \tag{3.17}$$

$$\hat{y}_\mathcal{B} = -\sin\varphi \;\; \hat{x}_\mathcal{I} + \cos\varphi\cos\theta \;\; \hat{y}_\mathcal{I} + \cos\varphi\sin\theta \;\; \hat{z}_\mathcal{I} \tag{3.18}$$

$$\hat{z}_\mathcal{B} = -\sin\theta \;\; \hat{y}_\mathcal{I} + \cos\theta \;\; \hat{z}_\mathcal{I} \tag{3.19}$$

If we now stack the 3 vectors into a 3×3 matrix R :

$$R = \begin{bmatrix} \hat{x}_\mathcal{B} & \hat{y}_\mathcal{B} & \hat{z}_\mathcal{B} \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \cos\theta\sin\varphi & \cos\theta\cos\varphi & -\sin\theta \\ \sin\theta\sin\varphi & \sin\theta\cos\varphi & \cos\theta \end{bmatrix} \tag{3.20}$$

The matrix R is an example of a 3×3 rotation matrix. We can express elements of the rotation matrix (3.20) r_{ij} as a dot product between two unitary vectors of the inertial and body frames :

$$R = \begin{bmatrix} \hat{x}_\mathcal{B} \cdot \hat{x}_\mathcal{I} & \hat{y}_\mathcal{B} \cdot \hat{x}_\mathcal{I} & \hat{z}_\mathcal{B} \cdot \hat{x}_\mathcal{I} \\ \hat{x}_\mathcal{B} \cdot \hat{y}_\mathcal{I} & \hat{y}_\mathcal{B} \cdot \hat{y}_\mathcal{I} & \hat{z}_\mathcal{B} \cdot \hat{y}_\mathcal{I} \\ \hat{x}_\mathcal{B} \cdot \hat{z}_\mathcal{I} & \hat{y}_\mathcal{B} \cdot \hat{z}_\mathcal{I} & \hat{z}_\mathcal{B} \cdot \hat{z}_\mathcal{I} \end{bmatrix} \tag{3.21}$$

Now, let's consider a general rotation matrix A :

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{3.22}$$

It is clear that a 3×3 rotation matrix has 9 entries. However, since a rotation matrix is composed of 3 unitary vectors, that are orthogonal two by two, the following constraints must be satisfied :

- The unit norm condition :

$$a_{11}^2 + a_{21}^2 + a_{31}^2 = 1 \tag{3.23}$$
$$a_{12}^2 + a_{22}^2 + a_{32}^2 = 1 \tag{3.24}$$
$$a_{13}^2 + a_{23}^2 + a_{33}^2 = 1 \tag{3.25}$$

- The orthogonality condition :

$$a_{11}a_{12} + a_{21}a_{22} + a_{31}a_{32} = 0 \tag{3.26}$$
$$a_{11}a_{13} + a_{21}a_{23} + a_{31}a_{33} = 0 \tag{3.27}$$
$$a_{12}a_{13} + a_{22}a_{23} + a_{32}a_{33} = 0 \tag{3.28}$$

These constraints can be expressed more compactly as follows :

$$A^T A = I \tag{3.29}$$

Additionally, from the last 6 constraints we conclude that $\det(A) = \pm 1$. To guarantee *proper rotations*[6] we choose $\det(A) = 1$.
With all this, we can finally derive a complete definition of the $SO(3)$ group.

3.7.1.1 Definition of $SO(3)$

The special orthogonal group is the group of rotations whose composition operation is the matrix multiplication, and it is defined as follows :

$$SO(3) = \{\mathcal{R} \in \mathbb{R}^{3\times 3} \mid \mathcal{R}^T\mathcal{R} = I, \det(\mathcal{R}) = 1\} \tag{3.30}$$

Consider $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3 \in SO(3)$, then all group axioms must be satisfied as proved below :

1. **Closure :** $\mathcal{R}_1\mathcal{R}_2 \in SO(3)$
   Because :

$$\begin{aligned}(\mathcal{R}_1\mathcal{R}_2)^T(\mathcal{R}_1\mathcal{R}_2) &= \mathcal{R}_2^T \underbrace{\mathcal{R}_1^T\mathcal{R}_1}_{I} \mathcal{R}_2 \\ &= \mathcal{R}_2^T\mathcal{R}_2 = I\end{aligned}$$

   and :

$$\det(\mathcal{R}_1\mathcal{R}_2) = \det(\mathcal{R}_1)\det(\mathcal{R}_2) = 1$$

2. **Associativity :** $\mathcal{R}_1(\mathcal{R}_2\mathcal{R}_3) = (\mathcal{R}_1\mathcal{R}_2)\mathcal{R}_3$
   This property follows from the associative nature of the matrix multiplication.

3. **Identity :** $\mathcal{R}_1 I = I\mathcal{R}_1 = \mathcal{R}_1$
   The identity element of the group is simply the 3-identity matrix.

[6] A proper rotation is a simple rotation around an axis, whereas an improper rotation is a rotation followed by a reflection in a mirror plane

4. **Inverse :** $\mathcal{R}_1\mathcal{R}_1^{-1} = \mathcal{R}_1^{-1}\mathcal{R}_1 = I$
   Simply because :

$$\begin{aligned} \mathcal{R}_1^T\mathcal{R}_1 = I &\implies \mathcal{R}^{-1} = \mathcal{R}^T \\ &\implies \det(\mathcal{R}_1^T) = \det(\mathcal{R}_1^{-1}) = 1 \end{aligned}$$

Additionally, it turns out that elements of this group form a smooth manifold. Therefore, we can safely say that the special orthogonal group is a Lie group.

3.7.1.2 Angular velocities and the Lie algebra of $SO(3)$

Suppose we are rotating the body frame $\{\mathcal{B}\}$, whose axes are $\{\hat{x}_\mathcal{B}, \hat{y}_\mathcal{B}, \hat{z}_\mathcal{B}\}$, around an arbitrary axis $\varpi = \hat{\varpi}\|\varpi\|$. Also consider $\mathcal{R}$ to be the rotation matrix describing the orientation of $\{\mathcal{B}\}$ with respect to an inertial frame $\{\mathcal{I}\}$, and $\dot{\mathcal{R}}$ its rate of change. By expressing ϖ in the inertial frame, it should be clear from Fig.3.5 that :

$$\dot{\hat{x}}_\mathcal{B} = \varpi_\mathcal{I} \times \hat{x}_\mathcal{B} \tag{3.31}$$

$$\dot{\hat{y}}_\mathcal{B} = \varpi_\mathcal{I} \times \hat{y}_\mathcal{B} \tag{3.32}$$

$$\dot{\hat{z}}_\mathcal{B} = \varpi_\mathcal{I} \times \hat{z}_\mathcal{B} \tag{3.33}$$

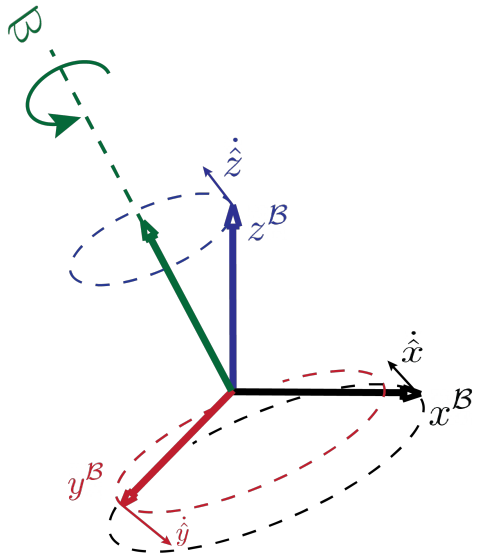


Figure 3.5: Angular rate of change of the body frame's axes

To get rid of the cross product, we introduce the 3×3 skew-symmetric matrix :

$$[\Psi] = \begin{bmatrix} 0 & -\Psi_3 & \Psi_2 \\ \Psi_3 & 0 & -\Psi_1 \\ -\Psi_2 & \Psi_1 & 0 \end{bmatrix} \; ; \; \Psi = \begin{bmatrix} \Psi_1 & \Psi_2 & \Psi_3 \end{bmatrix}^T \tag{3.34}$$

such that $\Psi \times \alpha = [\Psi]\alpha$. We can now right (3.32)-(3.33) in a more compact form :

$$\underbrace{\begin{bmatrix} \dot{\hat{x}}_\mathcal{B} & \dot{\hat{y}}_\mathcal{B} & \dot{\hat{z}}_\mathcal{B} \end{bmatrix}}_{\dot{\mathcal{R}}} = [\varpi_\mathcal{I}] \underbrace{\begin{bmatrix} \hat{x}_\mathcal{B} & \hat{y}_\mathcal{B} & \hat{z}_\mathcal{B} \end{bmatrix}}_{\mathcal{R}} \tag{3.35}$$

$$\dot{\mathcal{R}} = [\varpi_\mathcal{I}]\mathcal{R} \tag{3.36}$$

$$[\varpi_\mathcal{I}] = \dot{\mathcal{R}}\mathcal{R}^{-1} \tag{3.37}$$

Considering $\varpi_{\mathcal{B}} = \mathcal{R}^T \varpi_{\mathcal{I}}$, we can express $\dot{\mathcal{R}}$ in terms of $[\varpi_{\mathcal{B}}]$:

$$\begin{aligned}[\varpi_{\mathcal{B}}] &= [\mathcal{R}^T \varpi_{\mathcal{I}}] \\ &= \mathcal{R}^T [\varpi_{\mathcal{I}}] \mathcal{R} \\ &= \mathcal{R}^T (\dot{\mathcal{R}} \mathcal{R}^T) \mathcal{R} \\ &= \mathcal{R}^T \dot{\mathcal{R}} = \mathcal{R}^{-1} \dot{\mathcal{R}}\end{aligned}$$

Therefore, we write :

$$\dot{\mathcal{R}} = \mathcal{R}[\varpi_{\mathcal{B}}] \tag{3.38}$$

Following the definition of Lie algebra, we can see that if $\mathcal{R} = I$, then $\dot{\mathcal{R}} = [\varpi_{\mathcal{B}}]$ is the Lie algebra of $SO(3)$, which we denote $\mathfrak{so}(3)$. From the mechanical point of view, $[\varpi]$ can be seen as all the possible angular velocities when $\mathcal{R} = I$.

3.7.1.3 Definition of $\mathfrak{so}(3)$

We define the Lie algebra of $SO(3)$ as all the 3×3 skew-symmetric matrices :

$$\mathfrak{so}(3) = \{\varpi^\wedge = [\varpi] \in \mathbb{R}^{3\times 3} \mid \varpi \in \mathbb{R}^3\} \tag{3.39}$$

And it has the following properties :

1. $\mathfrak{so}(3)$ is a 3-dimensional vector space since the group $SO(3)$ has 3 degrees of freedom (9 entries - 6 constraints).
2. All elements of $\mathfrak{so}(3)$ are described by 3-vectors representing possible angular velocities.
3. We can infer the result of chaining multiple rotations only from the Lie algebra using the Lie bracket.

R.q. For the $SO(3)$ group, we use the $\wedge$ and the $\vee$ to transform a 3-vector into its corresponding skew-symmetric matrix and vice versa. Also, now that we know that $[\varpi] \in \mathfrak{so}(3)$, we will refer to it as $\varpi^\wedge$ for the sake of consistency.

3.7.1.4 The exponential and logarithm maps

By integrating (3.38), we obtain :

$$\mathcal{R}(t) = \mathcal{R} \exp(\varpi^\wedge t) \tag{3.40}$$

We will replace t by θ to express angles instead of times :

$$\mathcal{R}(\theta) = \mathcal{R} \exp(\varpi^\wedge \theta) \tag{3.41}$$

as we proved earlier, the exponentiation of an element of the Lie algebra belongs to the Lie group, so $\exp(\varpi^\wedge \theta) \in SO(3)$. Now, by expanding the exponential to its infinite series,

we observe that $(\varpi^\wedge)^3 = -\varpi^\wedge, (\varpi^\wedge)^5 = -(\varpi^\wedge)^3 = \varpi^\wedge ...$ and that $(\varpi^\wedge)^4 = -(\varpi^\wedge)^2 ...$, we can write:

$$\begin{aligned}\exp(\varpi^\wedge\theta) &= I + \varpi^\wedge\theta + \frac{(\varpi^\wedge)^2}{2!}\theta^2 + ... \\ &= I + (\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!}...)\varpi^\wedge + (\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + ...)(\varpi^\wedge)^2\end{aligned}$$

the use of the series expansion of $\cos\theta$ and $\sin\theta$ leads us to the final result :

$$\exp(\varpi^\wedge\theta) = I + \sin\theta\varpi^\wedge + (I - \cos\theta)\varpi^\wedge \in SO(3) \tag{3.42}$$

This formula is known as the *Rodrigues' formula* for rotations. Knowing the axis and the angle of rotation, one can recover the rotation matrix using this formula.

Now we will do the work the other way around. Starting from a rotation matrix, we define the axis and the angle of rotation that would lead to this rotation matrix. We will provide the final result of the logarithm map without proving it, but we redirect the reader to [65] for more details :

$$\begin{cases}\log(\mathcal{R}) = \frac{\mathcal{R}-\mathcal{R}^T}{2\sin\theta} \\ \theta = \cos^{-1}\frac{(tr(R)-1)}{2}\end{cases} \tag{3.43}$$

Note that if $\theta = 0$, the axis of rotation cannot be determined. Which makes sens because the frame did not move.

To sum up, given an axis and an angle of rotation, one can recover the rotation matrix through the exponential map. And given a rotation matrix, we can infer the axis and angle of rotation via the logarithm map.

3.7.2 The special euclidean group $SE(3)$

Until now, we have only been interested in expressing the orientation of our body frame $\{\mathcal{B}\}$. However, in order to describe complete poses of our vehicle, we need to find a way to express both orientations and positions in a handy way. For example, how can we describe both the orientation and position of the body frame with respect to $\{\mathcal{I}\}$ in Fig.3.6?

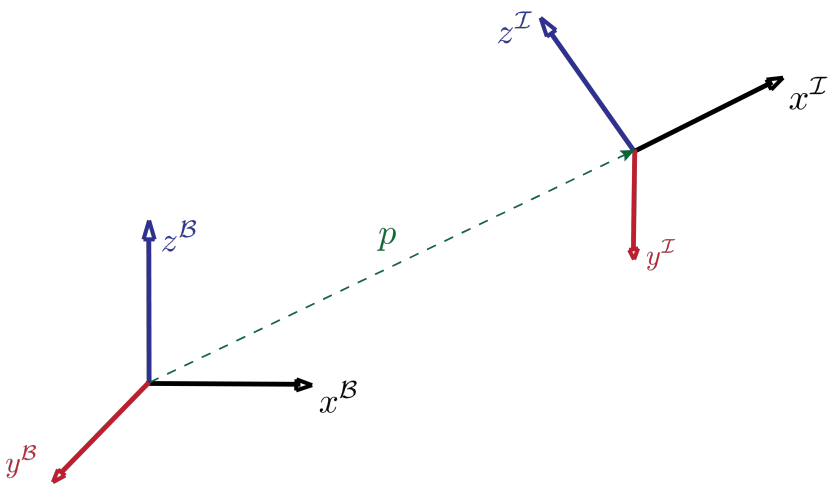


Figure 3.6: 3D pose representation

We can express such a representation by stacking a rotation matrix and a position vector into a single matrix :

$$T = \begin{bmatrix} \mathcal{R} & p \\ 0_3^T & 1 \end{bmatrix} \tag{3.44}$$

This is an example of a *homogeneous transformation matrix*. This type of matrices are the elements of the $SE(3)$ group.

3.7.2.1 Definition of $SE(3)$

The special euclidean group, whose composition operation is the matrix multiplication, is defined as follows :

$$SE(3) = \{T = \begin{bmatrix} \mathcal{R} & p \\ 0_3^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \mid \mathcal{R} \in SO(3), p \in \mathbb{R}^3, 0_3 \in \mathbb{R}^3\} \tag{3.45}$$

Again, we will prove that elements of this group satisfy all the group axioms. Consider $T_1, T_2, T_3 \in SE(3)$, then, the following must verify :

1. **Closure :** $T_1T_2 \in SE(3)$
   Because :
   $$T_1T_2 = \begin{bmatrix} \mathcal{R}_1 & p_1 \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} \mathcal{R}_2 & p_2 \\ 0_3^T & 1 \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1\mathcal{R}_2 & \mathcal{R}_1p_2 + p_1 \\ 0_3^T & 1 \end{bmatrix}$$
   since $\mathcal{R}_1\mathcal{R}_2 \in SO(3)$ and $\mathcal{R}_1p_2 + p_1 \in \mathbb{R}^3$, then $T_1T_2 \in SE(3)$.

2. **Associativity :** $T_1(T_2T_3) = (T_1T_2)T_3$
   Same as for SO(3), this property follows from the associative nature of the matrix multiplication.

3. **Identity :** $T_1I = IT_1 = T_1$
   The identity element of the group is simply the 4-identity matrix.

4. **Inverse :** $T_1T_1^{-1} = T_1^{-1}T_1 = I$
   If we calculate T_1^{-1}, we find that :
   $$T_1^{-1} = \begin{bmatrix} \mathcal{R}_1 & p_1 \\ 0_3^T & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathcal{R}_1^T & -\mathcal{R}_1^Tp_1 \\ 0_3^T & 1 \end{bmatrix} \tag{3.46}$$
   Again, since $\mathcal{R}^T \in SO(3)$ and $-\mathcal{R}^Tp_1 \in \mathbb{R}^3$, then $T_1^{-1} \in SE(3)$.

3.7.2.2 linear and angular velocities and the Lie algebra of $SE(3)$

We derived the structure of $\mathfrak{so}(3)$ using basic geometry and applied mechanics (angular velocities, axes of rotations, cross product...). This time, we will make use of some group properties for the sake of variety. By time-differentiating the inverse constraint of

multiplicative groups, we can find the structure of $\mathfrak{se}(3)$ which we denote $\Phi^\wedge$:

$$\Phi^\wedge = T^{-1}\dot{T} \tag{3.47}$$

$$= \begin{bmatrix} \mathcal{R}^T & -\mathcal{R}^T p \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} \dot{\mathcal{R}} & \dot{p} \\ 0_3^T & 0 \end{bmatrix} \tag{3.48}$$

$$= \begin{bmatrix} \underbrace{\mathcal{R}^T\dot{\mathcal{R}}}_{\varpi_\mathcal{B}^\wedge} & \underbrace{\mathcal{R}^T\dot{p}}_{v_\mathcal{B}} \\ 0_3^T & 0 \end{bmatrix} \tag{3.49}$$

$$= \begin{bmatrix} \varpi_\mathcal{B}^\wedge & v_\mathcal{B} \\ 0_3^T & 0 \end{bmatrix} \tag{3.50}$$

We have already seen that $\varpi_\mathcal{B}^\wedge$ is the matrix representation of angular velocity expressed in the $\{\mathcal{B}\}$ frame. Similarly, $v_\mathcal{B}$ is the linear velocity of the vehicle expressed in the body frame. A complete definition of $\mathfrak{se}(3)$ can be now derived.

3.7.2.3 definition of $\mathfrak{se}(3)$

The Lie algebra of the special euclidean group is defined as follows :

$$\mathfrak{se}(3) = \{\Phi^\wedge = \begin{bmatrix} \varpi^\wedge & v \\ 0_3^T & 0 \end{bmatrix} \in \mathbb{R}^{4\times4} \mid \varpi \in \mathfrak{so}(3), v \in \mathbb{R}^3\} \tag{3.51}$$

And it has the following properties :

1. $\mathfrak{se}(3)$ is a 6-dimensional vector space since the group $SE(3)$ has 6 degrees of freedom (12 entries - 6 constraints).
2. All elements of $\mathfrak{se}(3)$ are described by 6-vectors, which we call velocity twists $\mathcal{V}$.[7]
3. We can infer the result of chaining multiple transformations only from the Lie algebra using the Lie bracket. (see Fig.3.7).
4. Elements of $\mathfrak{se}(3)$ can be thought of as all possible linear and angular velocities when $T = I$.

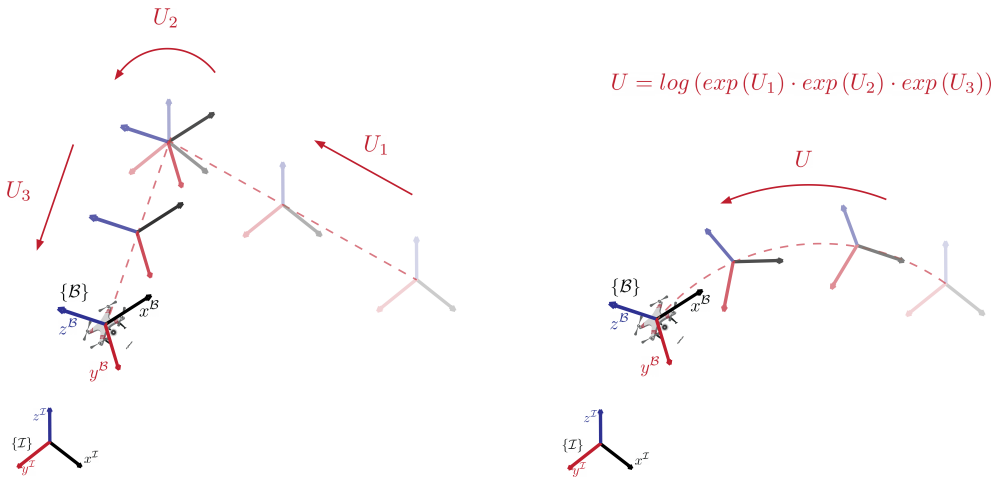


Figure 3.7: Smooth path generation using the Lie bracket

[7] Twists are 6-vectors describing angular and linear velocity in 3D space.

3.7.2.4 The exponential and logarithm maps

Just as we did a separation between the axis and angle of rotation when deriving the exponential map for rotations, we introduce the notion of screw vectors [65], which are unitary twists, and which allow us to do just the same.

Consider a twist :

$$\mathcal{V} = \begin{bmatrix} \varpi \\ v \end{bmatrix} = \mathcal{S}\theta \tag{3.52}$$

Such that $\mathcal{S}$ is a screw axis, and θ the angle of rotation around it. We define the screw axis as follows :

- If $\|\varpi\| \neq 0$:

$$\mathcal{S} = \begin{bmatrix} \frac{\varpi}{\|\varpi\|} \\ \frac{v}{\|\varpi\|} \end{bmatrix} \tag{3.53}$$

- If $\|\varpi\| = 0$ and $\|v\| \neq 0$:

$$\mathcal{S} = \begin{bmatrix} 0_3 \\ \frac{v}{\|v\|} \end{bmatrix} \tag{3.54}$$

Now we can write the infinite series of the exponential as follows :

$$\exp(\mathcal{S}^\wedge\theta) = I + \mathcal{S}^\wedge\theta + \frac{(\mathcal{S}^\wedge)^2}{2!}\theta^2 + ... \tag{3.55}$$

$$= \begin{bmatrix} \exp(\varpi\theta) & \rho(\theta)v \\ 0_3^T & 1 \end{bmatrix} \tag{3.56}$$

with :

$$\rho(\theta) = I\theta + (1 - \cos(\theta))\varpi^\wedge + (\theta - \sin\theta)(\varpi^\wedge)^2 \tag{3.57}$$

Therefore, given an arbitrary twist, we can recover its transformation matrix following these two cases :

- If $\|\varpi\| \neq 0$:

$$T = \begin{bmatrix} \exp(\varpi\theta) & (I\theta + (1 - \cos(\theta))\varpi^\wedge + (\theta - \sin\theta)\,(\varpi^\wedge)^2)v \\ 0_3^T & 1 \end{bmatrix} \tag{3.58}$$

- If $\|\varpi\| = 0$ and $\|v\| \neq 0$:

$$T = \begin{bmatrix} I & \theta v \\ 0_3^T & 1 \end{bmatrix} \tag{3.59}$$

Now again, the other way around, given a transformation matrix we derive the logarithm map to recover the screw axis and the angle of rotation as follows [65]:

- If $\mathcal{R} = I$ then $\|\varpi\| = 0$, $v = \frac{p}{\|p\|}$
- Otherwise, we use the logarithm map for $SO(3)$ to recover ϖ and θ, and we get v as follows :

$$v = \rho(\theta)^{-1} p \tag{3.60}$$

with :

$$\rho(\theta)^{-1} = \frac{1}{\theta} I + \frac{1}{2}\varpi^{\wedge} + \left(\frac{1}{\theta} - \frac{1}{2}\cot\frac{\theta}{2}\right)(\varpi^{\wedge})^2 \tag{3.61}$$

We now derived a way to move between $SE(3)$ and $\mathfrak{se}(3)$. Meaning that given an arbitrary twist, we can recover the resulting transformation matrix. And, inversely, given a transformation matrix, it is possible for us to extract the components of the screw vector, which are the angular and linear velocity, as well as the angle of rotation around the screw axis.

3.8 Conclusion

In this chapter, we tried to provide a light but complete introduction to Lie theory from a modest roboticist point of view. We were highly inspired by ([7], [62]–[64]) in selecting the material of the chapter, and by [65] in providing the physical interpretation of most of the abstract mathematical concepts of Lie theory. We tried to present Lie theory in a way that is beginner-friendly, by providing illustrations and giving robotics examples of some concepts when possible. We also tried to share some very interesting physical interpretations that we found in [65] and that allowed us to develop a certain way of thinking about Lie theory that made it more intuitive for us. We started by defining a Lie group as well as its properties, then we showed how they can act on other sets. We then introduced the notion of tangent spaces and Lie algebras. Subsequently, we defined the exponential and logarithm maps that allow the transition between the manifold of the Lie group and its Lie algebra, and we emphasized on how we can infer the behavior of elements in the manifold (which is nonlinear) only from elements of the Lie algebra (which is linear) and using the Lie bracket. Finally, and with the aim of transiting from the abstract mathematical world to a practical use of Lie theory, we introduced two of the most used Lie groups in robotics, namely, the *Special Orthogonal group* and the *Special Euclidean group*, and we made sure to emphasize on the geometrical and mechanical impact that these two groups have.

Chapter 4

Visual SLAM on Lie groups

Introduction

As we saw in the *literature review* chapter, several solutions were developed over the years to solve the SLAM problem, and the fact that they were made using a linearized motion model makes them subject to numerous problems like the high time complexity[1] and the lack of accuracy engendered by the linearization of the kinematics.

In this chapter, we will introduce a *Nonlinear Observer for SLAM on a Matrix Lie Group*, presented in [1]. Contrastingly to the old SLAM problem solutions, this approach, which makes a good use of the Lie theory, helps in preserving the non-linear properties of the motion model without any need for linearization, and reducing the time complexity through acting on a linear space (the Lie algebra) instead of directly acting on the non-linear space describing the dynamics of the system, and which are represented using a Lie group. First, we will start by introducing the mathematical tools needed for the reader to understand the theoretical proof, namely, the Lie group and its associated Lie algebra used throughout this chapter, as well as the 3D motion model. Next, we will mathematically prove how the observer is constructed and present the used theorems. Finally, we will write the algorithms of the two observers with and without velocity biases, (6) and (5) respectively, for the reader to have an idea on how to use it in practice.

4.1 Nonlinear Observer Design for SLAM on a Matrix Lie Group

In the work presented in [1], a non-linear observer is developed to estimate both the pose of a vehicle, along with the location of some surrounding landmarks. The mathematical development, which will be explained in the coming section, is performed on a matrix that belongs to the $SE_{1+n}(3)$ Lie group, which will later be defined, where it contains the pose of the vehicle and the location of the landmarks at every instant t, relative to the inertial frame. In order to observe the state, we will use the input matrix $U \in \mathfrak{se}_{1+n}(3)$, the location of the landmarks with respect to the body frame, which we assume available for measurement using a camera, and a correcting term, which will be designed later on.

4.1.1 Mathematical background

We define the sets of real and natural numbers as $\mathbb{R}$ and $\mathbb{N}$, respectively. The frobenius norm is defined for a real valued $n \times m$ matrix G, as $\|G\|_F = \sqrt{\langle\langle G, G\rangle\rangle}$, where $n, m \in \mathbb{N}$, while the operation $\langle\langle , \rangle\rangle$ is called inner product and defined for two given matrices $A, B \in \mathbb{R}^{n\times m}$ as $\langle\langle A, B\rangle\rangle = tr(A^T B)$. The $n \times n$ identity matrix is denoted I_n, and the i^{th} column of I_n is denoted e_i, for $i \in 0, 1...n$. The n dimensional column zero vector is denoted 0_n, while $0_{n\times m}$ describes the $n \times m$ zero matrix.

We denote $\mathcal{T}(R, p, \mathrm{L}) \in SE_{1+n}(3)$ as the transformation matrix on Lie group $SE_{1+n}(3)$, where $R \in SO(3)$ is a 3×3 rotation matrix which represents the rotation of the body

[1]Based on the study made in [67], the worst time complexity for the EKF SLAM is $O(n^3)$ for n landmarks, where it can be reduced in some cases to $O(n^2)$.

frame (which also corresponds to the rotation of the measurement tool, in our case the camera), p is a 3×1 vector which represents the location of the body frame, and L is a $3 \times n$ matrix which contains the locations of the n landmarks in the form $\mathrm{L} = (l_1, l_2, ..., l_n)$. The matrix $\mathcal{T}$ is introduced as:

$$\mathcal{T}(R, \mathrm{L}, p) = \left[\begin{array}{cc|c} R & p & \mathrm{L} \\ 0_{1\times 3} & 1 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & I_n \end{array}\right] \tag{4.1}$$

The Lie algebra associated with $\mathcal{T}$ is denoted $U \in \mathfrak{se}_{1+n}(3)$, it represents a compact form containing the angular $\Omega \in \mathfrak{so}(3)$ and translational $v \in \mathbb{R}^3$ velocities of the body frame with respect to the body frame, and also the velocities of the landmarks. We denote the velocity of the i^{th} landmark with respect to the body frame $v_i \in \mathbb{R}^3$, we will next use the $3 \times n$ matrix $\mathrm{v} = \begin{bmatrix} v_1 & v_2 & ... & v_n \end{bmatrix}$ to represent all the landmarks' velocities in a more compact form.

$$U = \left[\begin{array}{cc|c} \Omega & v & \mathrm{v} \\ \hline 0_{(n+1)\times 3} & 0_{n+1} & 0_{(n+1)\times n} \end{array}\right] \tag{4.2}$$

We define $X, X_1, X_2 \in SE_{1+n}(3)$, where we can easily verify that $SE_{1+n}(3)$ respects the properties of a Lie group: $X^{-1} = \mathcal{T}(R^T, -R^T p, -R^T \mathrm{L}) \in SE_{1+n}(3)$, $X_1 X_2 \in SE_{1+n}(3)$ and $X_1 X_1^{-1} = I_{n+4}$.
For a $(n+4)\times(n+4)$ real valued matrix $M = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}$, where $A_1 \in \mathbb{R}^{3\times 3}$, $A_2 \in \mathbb{R}^{3\times(n+1)}$, $A_3 \in \mathbb{R}^{(n+1)\times 3}$, and $A_4 \in \mathbb{R}^{(n+1)\times(n+1)}$, we define the function $\mathbb{P} : \mathbb{R}^{(n+4)\times(n+4)} \to \mathfrak{se}_{1+n}(3)$

$$\mathbb{P}(M) = \begin{bmatrix} \mathbb{P}_a(A_1) & A_2 \\ 0_{(n+1)\times 3} & 0_{(n+1)\times(n+1)} \end{bmatrix} \tag{4.3}$$

where : $\mathbb{P}_a : \mathbb{R}^{3\times 3} \to \mathfrak{so}(3)$, and defined as $\mathbb{P}_a(A_1) = (A_1 - A_1^T)/2$

We will next define the adjoint function $ad_X : SE_{1+n}(3) \times \mathfrak{se}_{1+n}(3) \to \mathfrak{se}_{1+n}(3)$

$$ad_X(U) = XUX^{-1} \tag{4.4}$$

The tangent space at an element $X \in SE_{1+n}(3)$ is denoted

$$T_X SE_{1+n}(3) := \{XU \in SE_{1+n}(3) | X \in SE_{1+n}(3), U \in \mathfrak{se}_{1+n}(3)\}$$

we define the Riemannian metric $\langle , \rangle_X : T_X SE_{1+n}(3) \times T_X SE_{1+n}(3) \longrightarrow \mathbb{R}$ as:

$$\langle XU_1, XU_2 \rangle_X = \langle\langle U_1, U_2 \rangle\rangle \tag{4.5}$$

For a smooth differentiable function $h : SE_{1+n}(3) \longrightarrow \mathbb{R}$, its gradient denoted $\nabla_X h \in T_X SE_{1+n}(3)$ relative to the Riemannian metric $\langle , \rangle_X$ is defined as :

$$\dot{h} = \langle \nabla_X h, \dot{X} \rangle_X = \langle\langle X^{-1} \nabla_X h, X^{-1} \dot{X} \rangle\rangle \tag{4.6}$$

Let $\{\mathcal{B}\}$ be the body frame attached to the vehicle, and $\{\mathcal{I}\}$ be the inertial frame as demonstrated in (Fig. 4.1). v_i represents the translational velocity of the i^{th} landmark with respect to the body frame $\{\mathcal{B}\}$, v the translational velocity of the vehicle with respect to $\{\mathcal{B}\}$, $\Omega = \omega^\wedge$ where $\Omega \in \mathfrak{so}(3)$ is the skew-symmetric representation of $\omega \in \mathbb{R}^3$ which represents the angular velocity of the body frame with respect to $\{\mathcal{B}\}$, $R \in SO(3)$ is a 3×3 rotation matrix with respect to $\{\mathcal{I}\}$, $p \in \mathbb{R}^3$ is the location of the body frame with respect to $\{\mathcal{I}\}$, and l_i defines the location vector of the i^{th} landmark with respect to the inertial frame $\{\mathcal{I}\}$.

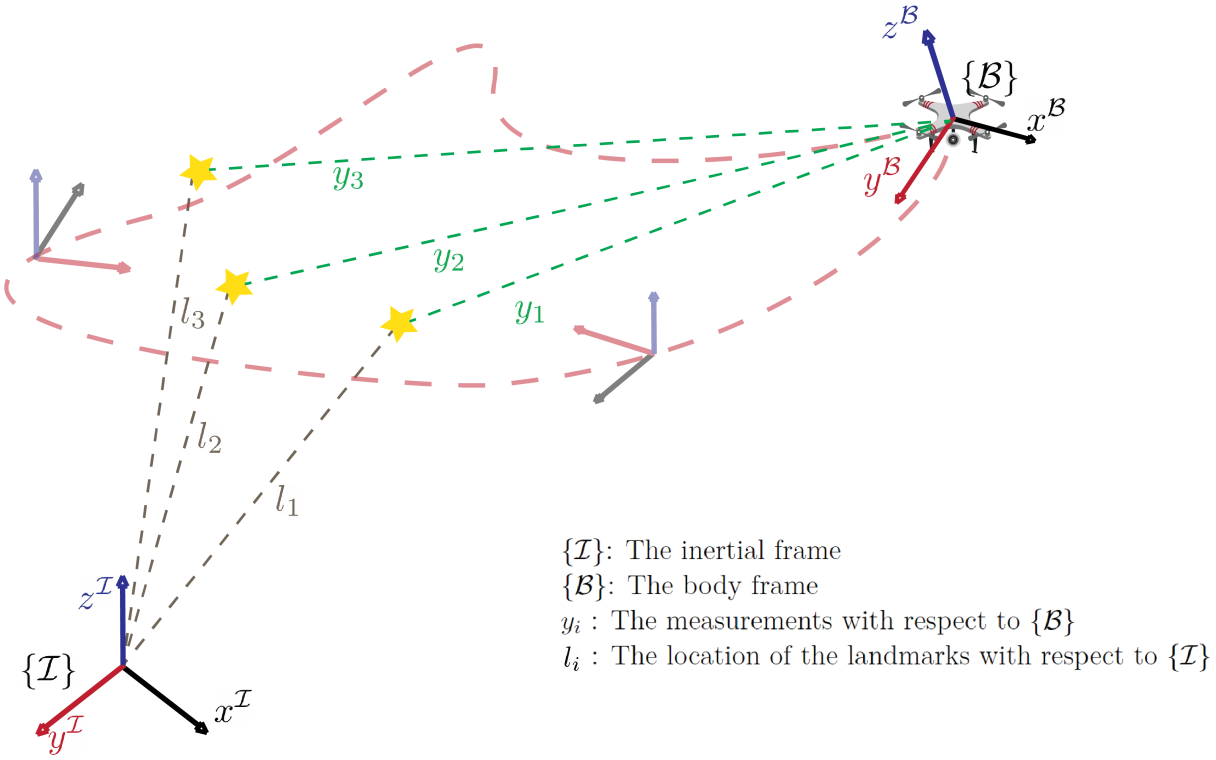


Figure 4.1: Graphical demonstration of the V-SLAM problem

For some given velocities Ω, v, and v_i, the 3D non-linear velocity motion model is defined as follows:

$$\dot{R} = R\Omega \tag{4.7}$$
$$\dot{p} = Rv \tag{4.8}$$
$$\dot{l}_i = Rv_i \qquad \forall i \in (1, 2, ..., n) \tag{4.9}$$

An equivalent compact form to write (4.7), (4.8), and (4.9) using $X = \mathcal{T}(R, p, \mathrm{L})$ and U defined in (4.1) and (4.2) respectively, is as follows:

$$\dot{X} = \frac{d}{dt}\left[\begin{array}{cc|c} R & p & \mathrm{L} \\ 0_{1\times 3} & 1 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & I_n \end{array}\right]$$

$$
\begin{aligned}
&= \left[\begin{array}{cc|c} \dot{R} & \dot{p} & \dot{\mathrm{L}} \\ \hline 0_{(n+1)\times 3} & 0_{n+1} & 0_{(n+1)\times n} \end{array}\right] \\
&= \left[\begin{array}{cc|c} R\Omega & Rv & R\mathrm{v} \\ \hline 0_{(n+1)\times 3} & 0_{n+1} & 0_{(n+1)\times n} \end{array}\right] \\
&= \left[\begin{array}{cc|c} R & p & \mathrm{L} \\ 0_{1\times 3} & 1 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & I_n \end{array}\right] \left[\begin{array}{cc|c} \Omega & v & \mathrm{v} \\ \hline 0_{(n+1)\times 3} & 0_{n+1} & 0_{(n+1)\times n} \end{array}\right] \\
&= XU
\end{aligned}
$$

$$\dot{X} = XU \tag{4.10}$$

where U is assumed to be bounded and available for measurement.

We define the measurement model (4.11) as the measurement of the i^{th} landmark with respect to the body frame $\{\mathcal{B}\}$:

$$y_i = R^T(l_i - p) \tag{4.11}$$

In order to solve the SLAM problem, we have to construct another augmented $(n+4)$ dimensional measurement vector b_i that respects the measurement model : (4.11).

$$
\begin{aligned}
y_i &= R^T(l_i - p) \\
&= \begin{bmatrix} R^T & -R^T p \end{bmatrix} \begin{bmatrix} l_i \\ 1 \end{bmatrix}
\end{aligned} \tag{4.12}
$$

Combining all the landmarks' measurements in a single $3 \times n$ matrix, gives

$$
\begin{aligned}
\begin{bmatrix} y_1 & y_2 & ... & y_n \end{bmatrix} &= \begin{bmatrix} R^T & -R^T p \end{bmatrix} \begin{bmatrix} l_1 & l_2 & ... & l_3 \\ 1 & 1 & ... & 1 \end{bmatrix} \\
&= \left[\begin{array}{cc|c} R^T & -R^T p & -R^T \mathrm{L} \end{array}\right] \begin{bmatrix} 0_{3\times n} \\ 1_{1\times n} \\ -I_n \end{bmatrix}
\end{aligned} \tag{4.13}
$$

Now to make the $(3 \times (n+4))$ matrix $\left[\begin{array}{cc|c} R^T & -R^T p & -R^T \mathrm{L} \end{array}\right]$ square and invertible we will define $b_i = \begin{bmatrix} y_i^T & 1 & -e_i^T \end{bmatrix}^T$, to get

$$
\begin{aligned}
\begin{bmatrix} b_1 & b_2 & ... & b_n \end{bmatrix} &= \left[\begin{array}{cc|c} R^T & -R^T p & -R^T \mathrm{L} \\ 0_3^T & 1 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & I_n \end{array}\right] \begin{bmatrix} 0_{3\times n} \\ 1_{1\times n} \\ -I_n \end{bmatrix} \\
&= X^{-1} \begin{bmatrix} r_1 & r_2 & ... & r_n \end{bmatrix}
\end{aligned} \tag{4.14}
$$

we finally find an equivalent measurement model (4.15) which will prove to be useful to construct the observer :

$$b_i = X^{-1} r_i \qquad \forall i \in (0, ..., n) \tag{4.15}$$

where :

$$r_i = \begin{bmatrix} 0_3^T & 1 & -e_i^T \end{bmatrix}^T$$

4.1.2 Gradient observer without velocity biases

In this section, we will construct an observer using (4.10) to estimate the state of the robot by measuring the surrounding landmarks with respect to the body frame.

We denote $\hat{R}(t) \in SO(3), \hat{p}(t) \in \mathbb{R}^3$, and $\hat{l}_i(t) \in \mathbb{R}^3$, $i = 1, 2..., n$, the estimates of $R(t) \in SO(3), p(t) \in \mathbb{R}^3$, and $l_i(t) \in \mathbb{R}^3$ at instant t, respectively. The estimate of the state $X(t) \in SE_{n+1}(3)$ is denoted $\hat{X}(t) \in SE_{n+1}(3)$, where $\hat{X} = \mathcal{T}(\hat{R}(t), \hat{p}(t), \hat{l}_i(t))$ and $X = \mathcal{T}(R(t), p(t), l_i(t))$.
Next, we will define the error $\tilde{X}(t) = \mathcal{T}(\tilde{R}(t), \tilde{p}(t), \tilde{l}_i(t))$ between the estimate $\hat{X}(t)$ and the real state $X(t)$ in a way that the error $\tilde{X}(t)$ remains in the Lie group $SE_{n+1}(3)$.

$$\tilde{X}(t) = X(t)\hat{X}(t)^{-1} \tag{4.16}$$

We have $\tilde{R}(t) = R(t)R(t)^T$, $\tilde{p}(t) = p(t) - \tilde{R}(t)\hat{p}(t)$

We can see that: $\hat{X}(t) \longrightarrow X(t) \Rightarrow \tilde{X}(t) \longrightarrow I_{n+4} \Rightarrow \tilde{X}(t) - I_{n+4} \longrightarrow 0_{(n+4)\times(n+4)}$

Now, we consider the smooth differentiable function $\mathcal{U} : SE_{n+1}(3) \longrightarrow \mathbb{R}$ that we will be using as a lyapunov function candidate

$$\mathcal{U}(X) = \frac{1}{2} tr\left((I_{n+4} - X)\, A\, (I_{n+4} - X)^T \right) \tag{4.17}$$

where A is a gain matrix defined as $A = \sum_{i=1}^{n} k_i r_i r_i^T$, implying that $A^T = A$

We replace h with $\mathcal{U}$ in (4.6) and we get :

$$\dot{\mathcal{U}}(X) = \langle \nabla_X \mathcal{U}, \dot{X} \rangle_X = \langle\langle X^{-1} \nabla_X \mathcal{U}, X^{-1}\dot{X} \rangle\rangle \tag{4.18}$$

The gradient $\nabla_X \mathcal{U}$ is defined as :

$$\nabla_X \mathcal{U} = X\mathbb{P}((I_{n+4} - X^{-1})A) \tag{4.19}$$

Notice that $\mathbb{P}((I_{n+4} - X^{-1})A) \in \mathfrak{se}_{n+1}(3)$ and therefore $\nabla_X \mathcal{U} \in T_X SE_{n+1}(3)$.

The observer proposed in [1] is the following,

$$\dot{\hat{X}} = \hat{X}(U - \Delta) \tag{4.20}$$

where $\Delta \in \mathfrak{se}_{n+1}(3)$ is a correcting term that we will use to stabilize the error dynamics.

We differentiate the equation (4.16) with respect to time to get :

$$\begin{aligned} \dot{\tilde{X}} &= \frac{d}{dt}\left(X\hat{X}^{-1} \right) \\ &= \dot{X}\hat{X}^{-1} + X\dot{\hat{X}}^{-1} \end{aligned} \tag{4.21}$$

knowing that $\hat{X}^{-1}\hat{X} = I_{n+4}$, we get $\dot{\hat{X}}^{-1} = -\hat{X}^{-1}\dot{\hat{X}}\hat{X}^{-1}$, we substitute it in (4.21) to obtain

$$\begin{aligned} \dot{\tilde{X}} &= \dot{X}\hat{X}^{-1} - X\hat{X}^{-1}\dot{\hat{X}}\hat{X}^{-1} \\ &= XU\hat{X}^{-1} - X\hat{X}^{-1}\hat{X}(U - \Delta)\hat{X}^{-1} \\ &= XU\hat{X}^{-1} - XU\hat{X}^{-1} + \underbrace{X\hat{X}^{-1}}_{\tilde{X}}\underbrace{\hat{X}\Delta\hat{X}^{-1}}_{ad_{\hat{X}}\Delta} \end{aligned} \tag{4.22}$$

we finally get

$$\dot{\tilde{X}} = \tilde{X} ad_{\hat{X}} \Delta \tag{4.23}$$

We now consider the following lyapunov candidate function

$$\mathcal{U}(\tilde{X}) = \frac{1}{2} tr\left(\left(I_{n+4} - \tilde{X}\right) A \left(I_{n+4} - \tilde{X}\right)^T\right) \tag{4.24}$$

knowing that $A = \sum_{i=1}^{n} k_i r_i r_i^T$, we can prove that A has the following form:

$$A = \left[\begin{array}{cc|c} 0_{3\times3} & 0_3 & 0_{3\times n} \\ 0_3^T & \sum_{i=1}^{n} k_i & -K^T \\ \hline 0_{n\times3} & -K & K_{diag} \end{array}\right] \tag{4.25}$$

Where $K = \begin{bmatrix} k_1 & k_2 & ... & k_n \end{bmatrix}^T$, and $K_{diag} = diag(K^T)$.
We also know that:

$$I_{n+4} - \tilde{X} = \left[\begin{array}{cc|c} I_3 - \tilde{R} & -\tilde{p} & -\tilde{\mathrm{L}} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times3} & 0_n & 0_{n\times n} \end{array}\right] \tag{4.26}$$

We substitute (4.25) and (4.26) in (4.24) :

$$\begin{aligned}
\mathcal{U}(\tilde{X}) &= \frac{1}{2} tr\left(\left[\begin{array}{cc|c} I_3 - \tilde{R} & -\tilde{p} & -\tilde{\mathrm{L}} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times3} & 0_n & 0_{n\times n} \end{array}\right] A \left[\begin{array}{cc|c} I_3 - \tilde{R}^T & 0_3 & 0_{3\times n} \\ -\tilde{p}^T & 0 & 0_{1\times n} \\ \hline -\tilde{\mathrm{L}}^T & 0_n & 0_{n\times n} \end{array}\right]\right) \\
&= \frac{1}{2} tr\left(\left[\begin{array}{cc|c} I_3 - \tilde{R} & -\tilde{p} & -\tilde{\mathrm{L}} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times3} & 0_n & 0_{n\times n} \end{array}\right] \left[\begin{array}{cc|c} 0_{3\times3} & 0_3 & 0_{3\times n} \\ \sum_{i=1}^{n} k_i \tilde{l}_i^T - \sum_{i=1}^{n} k_i \tilde{p}^T & 0 & 0_{1\times n} \\ \hline K\tilde{p}^T - K_{diag}\tilde{\mathrm{L}}^T & 0_n & 0_{n\times n} \end{array}\right]\right) \\
&= \frac{1}{2} tr\left(\left[\begin{array}{cc|c} -\sum_{i=1}^{n} k_i \tilde{p}\tilde{l}_i^T + \sum_{i=1}^{n} k_i \tilde{p}\tilde{p}^T - \tilde{\mathrm{L}}K\tilde{p}^T + \tilde{\mathrm{L}}K_{diag}\tilde{p}^T & 0_3 & 0_{3\times n} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times n} & 0_n & 0_{n\times n} \end{array}\right]\right) \\
&= \frac{1}{2} tr\left(\left[\begin{array}{cc|c} -\sum_{i=1}^{n} k_i \tilde{p}\tilde{l}_i^T + \sum_{i=1}^{n} k_i \tilde{p}\tilde{p}^T - \sum_{i=1}^{n} k_i \tilde{l}_i\tilde{p}^T + \sum_{i=1}^{n} k_i \tilde{l}_i\tilde{l}_i^T & 0_3 & 0_{3\times n} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times n} & 0_n & 0_{n\times n} \end{array}\right]\right) \\
&= \frac{1}{2} tr\left(\left[\begin{array}{cc|c} \sum_{i=1}^{n} k_i \tilde{p}\tilde{p}^T - \sum_{i=1}^{n} k_i \tilde{p}\tilde{l}_i^T - \sum_{i=1}^{n} k_i \tilde{l}_i\tilde{p}^T + \sum_{i=1}^{n} k_i \tilde{l}_i\tilde{l}_i^T & 0_3 & 0_{3\times n} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times n} & 0_n & 0_{n\times n} \end{array}\right]\right) \\
&= \frac{1}{2} \sum_{i=1}^{n} k_i \|\tilde{p} - \tilde{l}_i\|^2
\end{aligned}$$

and since the vector norm is preserved when applying rotations, we can write :

$$\mathcal{U}(\tilde{X}) = \frac{1}{2} \sum_{i=1}^{n} k_i \|\tilde{R}^T \left(\tilde{p} - \tilde{l}_i\right)\|^2 \tag{4.27}$$

We define the estimation error of the i^{th} landmark with respect to the inertial frame as :

$$\epsilon_i = \hat{l}_i - \hat{p} - \hat{R} y_i \tag{4.28}$$

we substitute (4.12) to get :

$$\begin{aligned}\epsilon_i &= \hat{l}_i - \hat{p} - \hat{R}R^T(l_i - p) \\ &= \hat{l}_i - \hat{p} - \tilde{R}^T(l_i - p) \\ &= \tilde{R}^T\left(\underbrace{p - \tilde{R}\hat{p}}_{\tilde{p}} - \left(\underbrace{l_i - \tilde{R}\hat{l}_i}_{\tilde{l}_i}\right)\right)\end{aligned}$$

$$\epsilon_i = \tilde{R}^T(\tilde{p} - \tilde{l}_i) \tag{4.29}$$

we substitute (4.29) in (4.27) to obtain :

$$\mathcal{U}(\tilde{X}) = \frac{1}{2}\sum_{i=1}^{n} k_i \|\epsilon_i\|^2 \tag{4.30}$$

We calculate the derivative of $\mathcal{U}(\tilde{X})$ with respect to time using the definition (4.18), and by substituting (4.19) and (4.23) we get :

$$\begin{aligned}\dot{\mathcal{U}}(\tilde{X}) &= \langle\langle \tilde{X}^{-1}\tilde{X}\mathbb{P}((I_{n+4} - \tilde{X}^{-1})A), \tilde{X}^{-1}\tilde{X}ad_{\hat{X}}\Delta\rangle\rangle \\ &= \langle\langle \mathbb{P}((I_{n+4} - \tilde{X}^{-1})A), ad_{\hat{X}}\Delta\rangle\rangle\end{aligned}$$

We choose $ad_{\hat{X}}\Delta$ to assure that $\dot{\mathcal{U}}(\tilde{X})$ is negative definite, by putting :

$$ad_{\hat{X}}\Delta = -\mathbb{P}((I_{n+4} - \tilde{X}^{-1})A)$$

we get :

$$\Delta = -ad_{\hat{X}^{-1}}\left(\mathbb{P}((I_{n+4} - \tilde{X}^{-1})A)\right) \tag{4.31}$$

we substitute (4.31) to obtain :

$$\begin{aligned}\dot{\mathcal{U}}(\tilde{X}) &= \langle\langle \mathbb{P}((I_{n+4} - \tilde{X}^{-1})A), -\mathbb{P}((I_{n+4} - \tilde{X}^{-1})A)\rangle\rangle \\ &= -\|\mathbb{P}((I_{n+4} - \tilde{X}^{-1})A)\|_F^2\end{aligned}$$

$$\dot{\mathcal{U}}(\tilde{X}) = -\|\mathbb{P}((I_{n+4} - \tilde{X}^{-1})A)\|_F^2 \tag{4.32}$$

By applying the second stability theorem of lyapunov explained in [68], we deduce that the error dynamics become Globally uniformly asymptotically stable.

We will next prove the global exponential stability :

$$
\begin{aligned}
\mathbb{P}((I_{n+4}-\tilde{X}^{-1})A) &= \mathbb{P}\left(\left[\begin{array}{cc|c} I_3-\tilde{R}^T & \tilde{R}^T\tilde{p} & \tilde{R}^T\tilde{\mathrm{L}} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n}\end{array}\right]\left[\begin{array}{cc|c} 0_{3\times 3} & 0_3 & 0_{3\times n} \\ 0_3^T & \sum_{i=1}^n k_i & -K^T \\ \hline 0_{n\times 3} & -K & K_{diag}\end{array}\right]\right) \\
&= \mathbb{P}\left(\left[\begin{array}{cc|c} 0_{3\times 3} & \sum_{i=1}^n k_i\tilde{R}^T\tilde{p}-\tilde{R}^T\tilde{\mathrm{L}}K & \mathcal{P} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n}\end{array}\right]\right) \\
&= \mathbb{P}\left(\left[\begin{array}{cc|c} 0_{3\times 3} & \sum_{i=1}^n k_i\tilde{R}^T\tilde{p}-\sum_{i=1}^n k_i\tilde{R}^T\tilde{l}_i & \mathcal{P} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n}\end{array}\right]\right) \\
&= \mathbb{P}\left(\left[\begin{array}{cc|c} 0_{3\times 3} & \sum_{i=1}^n k_i\tilde{R}^T\left(\tilde{p}-\tilde{l}_i\right) & \mathcal{P} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n}\end{array}\right]\right) \\
&= \mathbb{P}\left(\left[\begin{array}{cc|c} 0_{3\times 3} & \sum_{i=1}^n \epsilon_i & \epsilon \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n}\end{array}\right]\right) \\
&= \left[\begin{array}{cc|c} 0_{3\times 3} & \sum_{i=1}^n \epsilon_i & \epsilon \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n}\end{array}\right]
\end{aligned}
$$

Where $\mathcal{P} = -\tilde{R}^T\tilde{p}K^T + \tilde{R}^T\tilde{\mathrm{L}}K_{diag}$, and $\epsilon = \begin{bmatrix}-\epsilon_1 & -\epsilon_2 & \dots & -\epsilon_n\end{bmatrix}$.

$$
\mathbb{P}((I_{n+4}-\tilde{X}^{-1})A) = \left[\begin{array}{cc|c} 0_{3\times 3} & \sum_{i=1}^n k_i\epsilon_i & \epsilon K_{diag} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n}\end{array}\right] \tag{4.33}
$$

by substituting (4.33) in (4.32) we get :

$$
\begin{aligned}
\dot{\mathcal{U}}(\tilde{X}) &= -tr\left(\left[\begin{array}{cc|c} 0_{3\times 3} & \sum_{i=1}^n k_i\epsilon_i & \epsilon K_{diag} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n}\end{array}\right]\left[\begin{array}{cc|c} 0_{3\times 3} & \sum_{i=1}^n k_i\epsilon_i & \epsilon K_{diag} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n}\end{array}\right]^T\right) \\
&= -\|\sum_{i=1}^n k_i\epsilon_i\|^2 - \sum_{i=1}^n k_i^2\|\epsilon_i\|^2 \\
&\leq -\lambda\mathcal{U}
\end{aligned}
$$

Where λ is a real constant, by integrating both terms we will get :

$$
\mathcal{U}(t) \leq -e^{-\lambda t}\mathcal{U}(0) \tag{4.34}
$$

thus the landmark estimation errors converge exponentially to 0_3.
Next we will study the convergence of $\tilde{R}$ and $\tilde{p}$. From (4.31), (4.33), and (4.23) we get

$$\dot{\tilde{X}} = \left[\begin{array}{cc|c} \tilde{R} & \tilde{p} & \tilde{\mathrm{L}} \\ 0_3^T & 1 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & I_n \end{array}\right] \left[\begin{array}{cc|c} 0_{3\times 3} & -\sum_{i=1}^n k_i\epsilon_i & -\epsilon K_{diag} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n} \end{array}\right]$$
$$= \left[\begin{array}{cc|c} 0_{3\times 3} & -\tilde{R}\sum_{i=1}^n k_i\epsilon_i & -\tilde{R}\epsilon K_{diag} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n} \end{array}\right]$$

we obtain the following dynamics :

$$\dot{\tilde{R}} = 0_{3\times 3} \tag{4.35}$$
$$\dot{\tilde{p}} = -\tilde{R}\sum_{i=1}^n k_i\epsilon_i \tag{4.36}$$
$$\dot{\tilde{l}}_i = k_i\tilde{R}\epsilon_i \tag{4.37}$$

From these dynamics we conclude that $\tilde{R}$ and $\tilde{p}$ converge to some arbitrary constants $\tilde{R}^*$ and $\tilde{p}^*$ respectively. This result is due to the non-observability of the SLAM problem, where the absolute pose cannot be recovered, unless a full knowledge of the inertial frame's location and orientation is available. Which is natural, since SLAM itself finds applications in areas where we don't have access to such measurements. In other words, if we can measure the absolute position and orientation of our vehicle, then **SLAM wouldn't even be necessary**.

Now we will write the correcting term Δ in a way that all its parameters are known, using (4.33) and (4.28) we get :

$$(I_{n+4} - \tilde{X}^{-1})A = \left[\begin{array}{cc|c} 0_{3\times 3} & \sum_{i=1}^n k_i\epsilon_i & \epsilon K_{diag} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n} \end{array}\right]$$
$$= \sum_{i=1}^n \left[\begin{array}{cc|c} 0_{3\times 3} & k_i\epsilon_i & -k_i\epsilon_i e_i^T \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n} \end{array}\right]$$
$$= \sum_{i=1}^n k_i \left[\begin{array}{cc|c} 0_{3\times 3} & \epsilon_i & -\epsilon_i e_i^T \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n} \end{array}\right]$$
$$= \sum_{i=1}^n k_i \left(\left[\begin{array}{cc|c} 0_{3\times 3} & 0_3 & 0_3 \\ 0_3^T & 1 & -e_i^T \\ \hline 0_{n\times 3} & -e_i & e_i e_i^T \end{array}\right] - \left[\begin{array}{cc|c} 0_{3\times 3} & -\epsilon_i & \epsilon_i e_i^T \\ 0_3^T & 1 & -e_i^T \\ \hline 0_{n\times 3} & -e_i & e_i e_i^T \end{array}\right] \right)$$
$$= \sum_{i=1}^n k_i \left(r_i r_i^T - \left[\begin{array}{c} -\epsilon_i \\ \hline 1 \\ \hline -e_i \end{array}\right] \left[\begin{array}{ccc} 0_3^T & 1 & -e_i^T \end{array}\right] \right)$$
$$= \sum_{i=1}^n k_i \left(r_i r_i^T - \left[\begin{array}{c} -\hat{l}_i + \hat{p} + \hat{R}y_i \\ \hline 1 \\ \hline -e_i \end{array}\right] r_i^T \right)$$

$$
\begin{aligned}
&= \sum_{i=1}^{n} k_i \left(r_i r_i^T - \left[\begin{array}{cc|c} \hat{R} & \hat{p} & \hat{\mathrm{L}} \\ 0_{1\times 3} & 1 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & I_n \end{array}\right] \begin{bmatrix} y_i \\ 1 \\ -e_i \end{bmatrix} r_i^T \right) \\
&= \sum_{i=1}^{n} k_i \left(r_i r_i^T - \hat{X} b_i r_i^T \right) \\
&= \sum_{i=1}^{n} k_i \left(\left(r_i - \hat{X} b_i \right) \right) r_i^T
\end{aligned}
$$

thus we can compute Δ using :

$$
\Delta = -ad_{\hat{X}^{-1}} \left(\mathbb{P} \left(\sum_{i=1}^{n} k_i \left(\left(r_i - \hat{X} b_i \right) \right) r_i^T \right) \right) \tag{4.38}
$$

Finally the proposed observer is written as follows

$$
\begin{cases} \dot{\hat{X}}(t) = \hat{X}(t)(U - \Delta) \\ \Delta = -ad_{\hat{X}^{-1}} \sum_{i=1}^{n} k_i \left(\mathbb{P} \left(r_i - \hat{X}(t) b_i \right) r_i^T \right) \end{cases} \tag{4.39}
$$

Algorithm 5 Gradient observer without velocity biases

Data: $k_i > 0 \quad \forall i \in (1, 2, ..., n)$, $\hat{X}_0 \in SE_{1+n}(3)$, $U \in \mathfrak{se}_{1+n}(3)$, $r_i \in \mathbb{R}^{n+4} \quad \forall i \in (1, 2, ..., n), dt$

Result: $\hat{X}_k$

```
k ← 0
while 1 do
    Get measurments y_i(k)    ∀i ∈ (1, 2, ..., n)
    Construct b_i(k) from y_i(k) as in (4.15)
    Δ ← −ad_{X̂_k^{-1}} Σ_{i=1}^{n} k_i ( P ( r_i − X̂_k b_i ) r_i^T )
    X̂_k ← X̂_k exp ((U − Δ) dt)
    k ← k + 1
```

4.1.3 Observer design with velocity biases compensation

In this section, we will construct an observer of the state $X(t)$ while taking into account some biases on the velocity matrix U. We define the biased rotational and translational velocities as $\Omega_y = \Omega + b_\Omega \in \mathfrak{so}(3)$ and $v_y = v + b_v \in \mathbb{R}^3$ respectively, where $b_\Omega \in \mathfrak{so}(3)$ and $b_v \in \mathbb{R}^3$ are the rotational and translational velocity biases respectively.

The velocity biases matrix is denoted $b_U \in \mathfrak{se}_{n+1}(3)$, where $b_U = \mathcal{P}_b(b_\Omega, b_v)$, and $\mathcal{P}_b : \mathfrak{so}(3) \times \mathbb{R}^3 \longrightarrow \mathfrak{se}_{n+1}(3)$ defined as :

$$
\mathcal{P}_b(b_\Omega, b_v) = \left[\begin{array}{cc|c} b_\Omega & b_v & 0_{3\times n} \\ \hline 0_{(n+1)\times 3} & 0_{n+1} & 0_{(n+1)\times n} \end{array}\right] \tag{4.40}
$$

Using the kinematics model in (4.10), we define the biased velocity matrix $U_y = U + b_U \in \mathfrak{se}_{n+1}(3)$, let's now find the estimation error for the following observer :

$$\dot{\hat{X}} = \hat{X}\left(Uy - \hat{b}_U - \Delta\right) \tag{4.41}$$

From (4.16), and by following the same approach as in (4.22), we get :

$$\begin{aligned}
\dot{\tilde{X}} &= \dot{X}\hat{X}^{-1} + X\dot{\hat{X}}^{-1} \\
&= X(Uy - b_U)\hat{X}^{-1} - X\hat{X}^{-1}\hat{X}(U_y - \hat{b}_U - \Delta)\hat{X}^{-1} \\
&= X(\hat{b}_U - b_U)\hat{X}^{-1} + X\Delta\hat{X}^{-1} \\
&= X\hat{X}^{-1}\left(\hat{X}(\hat{b}_U - b_U)\hat{X}^{-1} + \hat{X}\Delta\hat{X}^{-1}\right) \\
&= \tilde{X}\left(-ad_{\hat{X}}\tilde{b}_U + ad_{\hat{X}}\Delta\right)
\end{aligned}$$

$$\dot{\tilde{X}} = \tilde{X}\left(-ad_{\hat{X}}\tilde{b}_U + ad_{\hat{X}}\Delta\right) \tag{4.42}$$

We consider the lyapunov function $\mathcal{V}(\tilde{X})$ as :

$$\mathcal{V}(\tilde{X}) = \mathcal{U}(\tilde{X}) + \frac{1}{2}tr(\tilde{b}_U K_b \tilde{b}_U^T) \tag{4.43}$$

where $K_b = diag(1/k_w, 1/k_w, 1/k_w, 1/k_v, 0_n^T)$

Next we will calculate $\dfrac{\partial}{\partial t}\left(\dfrac{1}{2}tr\left(\tilde{b}_U K_b \tilde{b}_U^T\right)\right)$, In order to mathematically derive $\dot{\mathcal{V}}$:

$$\frac{\partial}{\partial t}\left(\frac{1}{2}tr\left(\tilde{b}_U K_b \tilde{b}_U^T\right)\right) = \frac{\partial}{\partial t}\left(\frac{1}{2k_w}tr\left(\tilde{b}_\Omega^T\tilde{b}_\Omega\right) + \frac{1}{2k_w}\tilde{b}_v^T\tilde{b}_v\right) \tag{4.44}$$

since $\tilde{b}_\Omega = \tilde{b}_\omega^\wedge$, where $\tilde{b}_\omega = \begin{bmatrix}\tilde{b}_{\omega,x} & \tilde{b}_{\omega,y} & \tilde{b}_{\omega,z}\end{bmatrix}^T$, we can prove that :

$$\begin{aligned}
tr\left(\tilde{b}_\Omega^T\tilde{b}_\Omega\right) &= tr\left(\begin{bmatrix} 0 & \tilde{b}_{\omega,z} & -\tilde{b}_{\omega,y} \\ -\tilde{b}_{\omega,z} & 0 & \tilde{b}_{\omega,x} \\ \tilde{b}_{\omega,y} & -\tilde{b}_{\omega,x} & 0 \end{bmatrix}\begin{bmatrix} 0 & -\tilde{b}_{\omega,z} & \tilde{b}_{\omega,y} \\ \tilde{b}_{\omega,z} & 0 & -\tilde{b}_{\omega,x} \\ -\tilde{b}_{\omega,y} & \tilde{b}_{\omega,x} & 0 \end{bmatrix}\right) \\
&= 2\left(\tilde{b}_{\omega,x}^2 + \tilde{b}_{\omega,y}^2 + \tilde{b}_{\omega,z}^2\right) \\
&= 2\|\tilde{b}_\omega\|^2
\end{aligned}$$

$$tr\left(\tilde{b}_\Omega^T\tilde{b}_\Omega\right) = 2\tilde{b}_\omega^T\tilde{b}_\omega \tag{4.45}$$

we substitute (4.45) in (4.44), and we obtain :

$$\begin{aligned}
\frac{\partial}{\partial t}\left(\frac{1}{2}tr\left(\tilde{b}_U K_b \tilde{b}_U^T\right)\right) &= \frac{\partial}{\partial t}\left(\frac{1}{k_w}\tilde{b}_\omega^T\tilde{b}_\omega + \frac{1}{2k_w}\tilde{b}_v^T\tilde{b}_v\right) \\
&= \frac{2}{k_w}b_\omega^T\dot{\tilde{b}}_\omega + \frac{1}{k_v}\tilde{b}_v^T\dot{\tilde{b}}_v \\
&= \frac{1}{k_w}tr\left(\mathcal{P}_b^T(\tilde{b}_\Omega, 0_3)\mathcal{P}_b(\dot{\tilde{b}}_\Omega, 0_3)\right) + \frac{1}{k_v}tr\left(\mathcal{P}_b^T(0_{3\times3}, \tilde{b}_v)\mathcal{P}_b(0_{3\times3}, \dot{\tilde{b}}_v)\right)
\end{aligned}$$

where $\mathcal{P}_b$ is defined in (4.40). Using the properties of the trace $tr(A+B) = tr(A)+tr(B)$ and $tr(A^T) = tr(A)$ for $A \in \mathbb{R}^{(n+4)\times(n+4)}, B \in \mathbb{R}^{(n+4)\times(n+4)}$, we get :

$$\begin{aligned}
\frac{\partial}{\partial t}\left(\frac{1}{2}tr\left(\tilde{b}_U K_b \tilde{b}_U^T\right)\right) &= tr\left(\frac{1}{k_w}\mathcal{P}_b^T(\tilde{b}_\Omega, 0_3)\mathcal{P}_b(\dot{\tilde{b}}_\Omega, 0_3) + \frac{1}{k_v}\mathcal{P}_b^T(0_{3\times3}, \tilde{b}_v)\mathcal{P}_b(0_{3\times3}, \dot{\tilde{b}}_v)\right) \\
&= tr\left(\left[\begin{array}{cc|c} \frac{1}{k_\omega}\tilde{b}_\Omega^T\dot{\tilde{b}}_\Omega & 0_3 & 0_{3\times n} \\ \hline 0_3^T & \frac{1}{k_v}\tilde{b}_v^T\dot{\tilde{b}}_v & 0_n^T \\ 0_{(n)\times3} & 0_n & 0_{(n)\times n}\end{array}\right]\right) \\
&= tr\left(\left[\begin{array}{cc|c} \frac{1}{k_\omega}\tilde{b}_\Omega^T\dot{\tilde{b}}_\Omega & \frac{1}{k_\omega}\tilde{b}_\Omega^T\dot{\tilde{b}}_v & 0_{3\times n} \\ \hline \frac{1}{k_v}\tilde{b}_v^T\dot{\tilde{b}}_\Omega & \frac{1}{k_v}\tilde{b}_v^T\dot{\tilde{b}}_v & 0_n^T \\ 0_{(n)\times3} & 0_n & 0_{(n)\times n}\end{array}\right]\right) \\
&= tr\left(\left[\begin{array}{cc|c} \frac{1}{k_\omega}\tilde{b}_\Omega & \frac{1}{k_v}\tilde{b}_v & 0_{3\times n} \\ \hline 0_{(n+1)\times3} & 0_{n+1} & 0_{(n+1)\times n}\end{array}\right]^T\left[\begin{array}{cc|c} \dot{\tilde{b}}_\Omega & \dot{\tilde{b}}_v & 0_{3\times n} \\ \hline 0_{(n+1)\times3} & 0_{n+1} & 0_{(n+1)\times n}\end{array}\right]\right) \\
&= tr\left(\left[\begin{array}{cc|c} \dot{\tilde{b}}_\Omega & \dot{\tilde{b}}_v & 0_{3\times n} \\ \hline 0_{(n+1)\times3} & 0_{n+1} & 0_{(n+1)\times n}\end{array}\right]^T\left[\begin{array}{cc|c} \frac{1}{k_\omega}\tilde{b}_\Omega & \frac{1}{k_v}\tilde{b}_v & 0_{3\times n} \\ \hline 0_{(n+1)\times3} & 0_{n+1} & 0_{(n+1)\times n}\end{array}\right]\right) \\
&= tr\left(\dot{\tilde{b}}_U^T\tilde{b}_U K_b\right) \\
&= \left\langle\left\langle \dot{\tilde{b}}_U, \tilde{b}_U K_b\right\rangle\right\rangle
\end{aligned}$$

$$\frac{\partial}{\partial t}\left(\frac{1}{2}tr\left(\tilde{b}_U K_b \tilde{b}_U^T\right)\right) = \left\langle\left\langle \dot{\tilde{b}}_U, \tilde{b}_U K_b\right\rangle\right\rangle \tag{4.46}$$

Knowing that $K_b = K_b^T$, we will next calculate $\nabla_{\tilde{b}_U}\mathcal{V}^2$:

$$\begin{aligned}
\nabla_{\tilde{b}_U}\mathcal{V} &= \frac{\partial \mathcal{V}}{\partial \tilde{b}_U} \\
&= \frac{1}{2}\tilde{b}_U(K_b + K_b^T) \\
&= \tilde{b}_U K_b \\
&= K_b \tilde{b}_U
\end{aligned}$$

$$\nabla_{\tilde{b}_U}\mathcal{V} = \tilde{b}_U K_b \tag{4.47}$$

Using (4.6),(4.19), (4.42), (4.47), and the fact that $\nabla_{\tilde{X}}\mathcal{V} = \nabla_{\tilde{X}}\mathcal{U}$ we find :

$$\begin{aligned}
\dot{\mathcal{V}}(\tilde{X}) &= \langle \nabla_{\tilde{X}}\mathcal{V}, \dot{\tilde{X}}\rangle_{\tilde{X}} + \langle\langle \dot{\tilde{b}}_U, \nabla_{\tilde{b}_U}\mathcal{V}\rangle\rangle \\
&= \langle\langle \tilde{X}^{-1}\nabla_{\tilde{X}}\mathcal{V}, \tilde{X}^{-1}\dot{\tilde{X}}\rangle\rangle + \langle\langle \dot{\tilde{b}}_U, \nabla_{\tilde{b}_U}\mathcal{V}\rangle\rangle \\
&= \langle\langle \tilde{X}^{-1}\nabla_{\tilde{X}}\mathcal{U}, \tilde{X}^{-1}\dot{\tilde{X}}\rangle\rangle + \langle\langle \dot{\tilde{b}}_U, \tilde{b}_U K_b\rangle\rangle \\
&= \left\langle\left\langle \mathbb{P}\left(\left(I_{n+4} - \tilde{}^{-1}\right)A\right), -ad_{\hat{X}}\tilde{b}_U + ad_{\hat{X}}\Delta\right\rangle\right\rangle + \langle\langle \dot{\tilde{b}}_U, \tilde{b}_U K_b\rangle\rangle \\
&= \left\langle\left\langle \mathbb{P}\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right), ad_{\hat{X}}\Delta\right\rangle\right\rangle + \left\langle\left\langle \mathbb{P}\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right), -ad_{\hat{X}}\tilde{b}_U\right\rangle\right\rangle + \langle\langle \dot{\tilde{b}}_U, \tilde{b}_U K_b\rangle\rangle
\end{aligned}$$

[2]We calculate the gradient using $\frac{\partial}{\partial X}tr(XGX^T) = X(G+G^T)$, proved in [69]

by choosing :

$$\Delta = -ad_{\hat{X}^{-1}}\left(\mathbb{P}\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right)\right) \tag{4.48}$$

$$\dot{\tilde{b}}_U = \mathbb{P}\left(\hat{X}^T\left(I_{n+4} - \tilde{X}^{-1}\right)A\hat{X}^{-T}\right)K_b' \tag{4.49}$$

where $K_b' = diag(k_w, k_w, k_w, k_v, 0_n^T)$, we also define $\mathbb{I} = diag(1,1,1,1,0_n^T)$
and using the following lemmas :
for $M \in \mathfrak{se}_{n+1}(3)$ and $D, G \in \mathbb{R}^{(n+4)\times(n+4)}$:

$$\langle\langle M, b_U\rangle\rangle = \langle\langle \mathbb{P}(M), b_U\rangle\rangle = \langle\langle \mathbb{P}(M)\mathbb{I}, b_U\rangle\rangle \tag{4.50}$$

$$tr(GD) = tr(DG) \tag{4.51}$$

we get :

$$\begin{aligned}
\langle\langle \dot{\tilde{b}}_U, \tilde{b}_U K_b\rangle\rangle &= \langle\langle \mathbb{P}\left(\hat{X}^T\left(I_{n+4} - \tilde{X}^{-1}\right)A\hat{X}^{-T}\right)K_b', \tilde{b}_U K_b\rangle\rangle \\
&= \langle\langle \mathbb{P}\left(\hat{X}^T\left(I_{n+4} - \tilde{X}^{-1}\right)A\hat{X}^{-T}\right)\mathbb{I}, \tilde{b}_U\rangle\rangle \\
&= \langle\langle \mathbb{P}\left(\hat{X}^T\left(I_{n+4} - \tilde{X}^{-1}\right)A\hat{X}^{-T}\right), \tilde{b}_U\rangle\rangle \\
&= \langle\langle \hat{X}^T\left(I_{n+4} - \tilde{X}^{-1}\right)A\hat{X}^{-T}, \tilde{b}_U\rangle\rangle \\
&= tr\left(\left(\hat{X}^T\left(I_{n+4} - \tilde{X}^{-1}\right)A\hat{X}^{-T}\right)^T\tilde{b}_U\right) \\
&= tr\left(\hat{X}^{-1}\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right)^T\hat{X}\tilde{b}_U\right) \\
&= tr\left(\hat{X}^{-1}\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right)^T\left(\hat{X}\tilde{b}_U\right)\right) \\
&= tr\left(\left(\hat{X}\tilde{b}_U\right)\hat{X}^{-1}\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right)^T\right) \\
&= tr\left(\left(\hat{X}\tilde{b}_U\hat{X}^{-1}\right)\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right)^T\right) \\
&= tr\left(\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right)^T\left(\hat{X}\tilde{b}_U\hat{X}^{-1}\right)\right) \\
&= \left\langle\left\langle\left(I_{n+4} - \tilde{X}^{-1}\right)A, ad_{\hat{X}}\tilde{b}_U\right\rangle\right\rangle \\
&= \left\langle\left\langle\mathbb{P}\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right), ad_{\hat{X}}\tilde{b}_U\right\rangle\right\rangle
\end{aligned}$$

$$\langle\langle \dot{\tilde{b}}_U, \tilde{b}_U K_b\rangle\rangle = \left\langle\left\langle\mathbb{P}\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right), ad_{\hat{X}}\tilde{b}_U\right\rangle\right\rangle \tag{4.52}$$

thus :

$$\begin{aligned}
\dot{\mathcal{V}}(\tilde{X}) &= -\left\|\mathbb{P}\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right)\right\|_F^2 + \underbrace{\left\langle\left\langle\mathbb{P}\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right), -ad_{\hat{X}}\tilde{b}_U\right\rangle\right\rangle + \langle\langle \dot{\tilde{b}}_U, K_b\tilde{b}_U\rangle\rangle}_{=0} \\
&= -\left\|\mathbb{P}\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right)\right\|_F^2
\end{aligned}$$

Using (4.33), we obtain the semi-negative definite function $\dot{\mathcal{V}}(\tilde{X})$:

$$\dot{\mathcal{V}}(\tilde{X}) = -\|\sum_{i=1}^{n} k_i\epsilon_i\|^2 - \sum_{i=1}^{n} k_i^2\|\epsilon_i\|^2 \tag{4.53}$$

thus by applying the second stability theorem of lyapunov, we proved the uniform stability of the proposed observer for all $\tilde{b}_U$ and ϵ_i where $i \in (1, 2, ..., n)$, which implies that $\mathcal{V}$ is non-increasing. As a result, $\|\epsilon_i\|$ and $\|\tilde{b}_U\|$ are also non-increasing and remain always bounded, for all $\forall i \in (1, 2, ..., n)$.

Next we will prove the asymptotic stability of the observer, to demonstrate that $\dot{\mathcal{V}}$ decreases to zero.

First we will calculate $ad_{\hat{X}}\tilde{b}_U$ by applying (4.4) :

$$\begin{aligned}
ad_{\hat{X}}\tilde{b}_U &= \hat{X}\tilde{b}_U\hat{X}^{-1} \\
&= \left[\begin{array}{ccc|c} \hat{R} & \hat{p} & \hat{\mathrm{L}} & \\ 0_{1\times3} & 1 & 0_{1\times n} & \\ \hline 0_{n\times3} & 0_n & I_n & \end{array}\right] \left[\begin{array}{ccc|c} \tilde{b}_\Omega & \tilde{b}_v & 0_{3\times n} & \\ 0_{1\times3} & 0 & 0_{1\times n} & \\ \hline 0_{n\times3} & 0_n & 0_{n\times n} & \end{array}\right] \left[\begin{array}{ccc|c} \hat{R}^T & -\hat{R}^T\hat{p} & -\hat{R}^T\hat{\mathrm{L}} & \\ 0_{1\times3} & 1 & 0_{1\times n} & \\ \hline 0_{n\times3} & 0_n & I_n & \end{array}\right] \\
&= \left[\begin{array}{ccc|c} \hat{R} & \hat{p} & \hat{\mathrm{L}} & \\ 0_{1\times3} & 1 & 0_{1\times n} & \\ \hline 0_{n\times3} & 0_n & I_n & \end{array}\right] \left[\begin{array}{cc|c} \tilde{b}_\Omega R^T & -\tilde{b}_\Omega\hat{R}^T\hat{p} + \tilde{b}_v & -\tilde{b}_\Omega\hat{R}^T\hat{\mathrm{L}} \\ 0_{1\times3} & 0 & 0_{1\times n} \\ \hline 0_{n\times3} & 0_n & 0_{n\times n} \end{array}\right] \\
&= \left[\begin{array}{cc|c} \hat{R}\tilde{b}_\Omega\hat{R}^T & -\hat{R}\tilde{b}_\Omega\hat{R}^T\hat{p} + \hat{R}\tilde{b}_v & -\hat{R}\tilde{b}_\Omega\hat{R}^T\hat{\mathrm{L}} \\ 0_{1\times3} & 0 & 0_{1\times n} \\ \hline 0_{n\times3} & 0_n & 0_{n\times n} \end{array}\right]
\end{aligned}$$

$$ad_{\hat{X}}\tilde{b}_U = \left[\begin{array}{cc|c} \hat{R}\tilde{b}_\Omega\hat{R}^T & -\hat{R}\tilde{b}_\Omega\hat{R}^T\hat{p} + \hat{R}\tilde{b}_v & -\hat{R}\tilde{b}_\Omega\hat{R}^T\hat{\mathrm{L}} \\ 0_{1\times3} & 0 & 0_{1\times n} \\ \hline 0_{n\times3} & 0_n & 0_{n\times n} \end{array}\right] \tag{4.54}$$

From (4.42),(4.48),(4.49),(4.33), and (4.54) we obtain :

$$\begin{aligned}
\dot{\tilde{X}} &= \tilde{X}\left(-ad_{\hat{X}}\tilde{b}_U + ad_{\hat{X}}\Delta\right) \\
&= \left[\begin{array}{ccc|c} \tilde{R} & \tilde{p} & \tilde{\mathrm{L}} & \\ 0_{1\times3} & 1 & 0_{1\times n} & \\ \hline 0_{n\times3} & 0_n & I_n & \end{array}\right] \left[\begin{array}{cc|c} -\hat{R}\tilde{b}_\Omega\hat{R}^T & \mathcal{A} & \mathcal{B} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times3} & 0_n & 0_{n\times n} \end{array}\right] \\
&= \left[\begin{array}{cc|c} -\tilde{R}\hat{R}\tilde{b}_\Omega\hat{R}^T & \tilde{R}\left(\hat{R}\tilde{b}_\Omega\hat{R}^T\hat{p} - \hat{R}\tilde{b}_v - \sum_{i=1}^{n} k_i\epsilon_i\right) & \tilde{R}\left(\hat{R}\tilde{b}_\Omega\hat{R}^T\hat{l}_i + k_i\epsilon_i\right) \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times3} & 0_n & 0_{n\times n} \end{array}\right]
\end{aligned}$$

where $\mathcal{A} = \hat{R}\tilde{b}_\Omega\hat{R}^T\hat{p} - \hat{R}\tilde{b}_v - \sum_{i=1}^{n} k_i\epsilon_i$, and $\mathcal{B} = \hat{R}\tilde{b}_\Omega\hat{R}^T\hat{\mathrm{L}} - \epsilon K_{diag}$
after identification, we get :

$$\dot{\tilde{R}} = -\tilde{R}\hat{R}\tilde{b}_\Omega\hat{R}^T \tag{4.55}$$

$$\dot{\tilde{p}} = \tilde{R}\left(\hat{R}\tilde{b}_\Omega\hat{R}^T\hat{p} - \hat{R}\tilde{b}_v - \sum_{i=1}^{n} k_i\epsilon_i\right) \tag{4.56}$$

$$\dot{\tilde{l}}_i = \tilde{R}\left(\hat{R}\tilde{b}_\Omega\hat{R}^T\hat{l}_i + k_i\epsilon_i\right), \quad \forall i \in (1, 2, ..., n) \tag{4.57}$$

using (4.49) and (4.33) we obtain :

$$
\begin{aligned}
\dot{\tilde{b}}_U &= \mathbb{P}\left(\left[\begin{array}{cc|c} \hat{R} & 0_3^T & 0_{3\times n} \\ \hat{p}^T & 1 & 0_{1\times n} \\ \hline \hat{\mathrm{L}}^T & 0_n & I_n \end{array}\right]\left[\begin{array}{cc|c} 0_{3\times 3} & \sum_{i=1}^n k_i\epsilon_i & \epsilon K_{diag} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n} \end{array}\right]\hat{X}^{-T}\right)K_b' \\
&= \mathbb{P}\left(\left[\begin{array}{cc|c} 0_{3\times 3} & \hat{R}^T\sum_{i=1}^n k_i\epsilon_i & \hat{R}^T\epsilon K_{diag} \\ 0_3^T & \hat{p}^T\sum_{i=1}^n k_i\epsilon_i & \hat{p}^T\epsilon K_{diag} \\ \hline 0_{n\times 3} & \hat{\mathrm{L}}^T\sum_{i=1}^n k_i\epsilon_i & \hat{\mathrm{L}}^T\epsilon K_{diag} \end{array}\right]\hat{X}^{-T}\right)K_b' \\
&= \mathbb{P}\left(\left[\begin{array}{cc|c} 0_{3\times 3} & \hat{R}^T\sum_{i=1}^n k_i\epsilon_i & \hat{R}^T\epsilon K_{diag} \\ 0_3^T & \hat{p}^T\sum_{i=1}^n k_i\epsilon_i & \hat{p}^T\epsilon K_{diag} \\ \hline 0_{n\times 3} & \hat{\mathrm{L}}^T\sum_{i=1}^n k_i\epsilon_i & \hat{\mathrm{L}}^T\epsilon K_{diag} \end{array}\right]\left[\begin{array}{cc|c} \hat{R} & 0_3^T & 0_{3\times n} \\ -\hat{p}^T\hat{R} & 1 & 0_{1\times n} \\ \hline -\hat{\mathrm{L}}^T\hat{R} & 0_n & I_n \end{array}\right]\right)K_b' \\
&= \mathbb{P}\left(\left[\begin{array}{cc|c} \hat{R}^T\sum_{i=1}^n k_i\epsilon_i\left(\hat{l}_i^T-\hat{p}^T\right)\hat{R} & \hat{R}^T\sum_{i=1}^n k_i\epsilon_i & \hat{R}^T\epsilon K_{diag} \\ \hat{p}^T\sum_{i=1}^n k_i\epsilon_i\left(\hat{l}_i^T-\hat{p}^T\right)\hat{p} & \hat{p}^T\sum_{i=1}^n k_i\epsilon_i & \hat{p}^T\epsilon K_{diag} \\ \hline \hat{\mathrm{L}}^T\sum_{i=1}^n k_i\epsilon_i\left(\hat{l}_i^T-\hat{p}^T\right)\hat{R} & \hat{\mathrm{L}}^T\sum_{i=1}^n k_i\epsilon_i & \hat{\mathrm{L}}^T\epsilon K_{diag} \end{array}\right]\right)K_b' \\
&= \left[\begin{array}{cc|c} P_a(\hat{R}^T\sum_{i=1}^n k_i\epsilon_i\left(\hat{l}_i^T-\hat{p}^T\right)\hat{R}) & \hat{R}^T\sum_{i=1}^n k_i\epsilon_i & \hat{R}^T\epsilon K_{diag} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n} \end{array}\right]K_b' \\
&= \left[\begin{array}{cc|c} k_w P_a(\hat{R}^T\sum_{i=1}^n k_i\epsilon_i\left(\hat{l}_i^T-\hat{p}^T\right)\hat{R}) & k_v\hat{R}^T\sum_{i=1}^n k_i\epsilon_i & 0_{3\times n} \\ 0_3^T & 0 & 0_{1\times n} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times n} \end{array}\right]
\end{aligned}
$$

after identification we obtain :

$$\dot{\tilde{b}}_\Omega = k_w P_a(\hat{R}^T\sum_{i=1}^n k_i\epsilon_i\left(\hat{l}_i^T-\hat{p}^T\right)\hat{R}) \tag{4.58}$$

$$\dot{\tilde{b}}_v = k_v\hat{R}^T\sum_{i=1}^n k_i\epsilon_i \tag{4.59}$$

We will next prove the convergence of $\dot{\mathcal{V}}$ to 0, which is equivalent to the convergences of $\epsilon_i, \forall i \in (1,2,...,n)$ to 0_3. Based on Barbalat theorem which states that $\lim_{t\to\infty}\dot{\mathcal{V}}\to 0$, if $\mathcal{V}$ has a finite limit and $\ddot{\mathcal{V}}$ is bounded.
We previously proved that $\mathcal{V}$ has a finite limit, now it remains to prove that $\ddot{\mathcal{V}}$ is bounded, which can be achieved by proving the boundedness of $\dot{\epsilon}_i$.
Using (4.29) and the fact that $\hat{R}^T(\hat{p}-\hat{l}_i) = -\hat{R}^T\epsilon_i + R^T(p-l_i)$ we get :

$$
\begin{aligned}
\dot{\epsilon}_i &= \tilde{R}^T\left(\dot{\tilde{p}}-\dot{\tilde{l}}_i\right)+\dot{\tilde{R}}^T(\tilde{p}-\tilde{l}_i) \\
&= \tilde{R}^T\left(\tilde{R}\left(\hat{R}\tilde{b}_\Omega\hat{R}^T\hat{p}-\hat{R}\tilde{b}_v-\sum_{i=1}^n k_i\epsilon_i\right)-\tilde{R}\left(\hat{R}\tilde{b}_\Omega\hat{R}^T\hat{l}_i+k_i\epsilon_i\right)\right)+\left(-\tilde{R}\hat{R}\tilde{b}_\Omega\hat{R}^T\right)^T\left(\tilde{p}-\tilde{l}_i\right) \\
&= \hat{R}\tilde{b}_\Omega\hat{R}^T\hat{p}-\hat{R}\tilde{b}_v-\sum_{i=1}^n k_i\epsilon_i-\hat{R}\tilde{b}_\Omega\hat{R}^T\hat{l}_i-k_i\epsilon_i-\left(\tilde{R}\hat{R}\tilde{b}_\Omega\hat{R}^T\right)^T\tilde{R}\epsilon_i
\end{aligned}
$$

$$= -\hat{R}\tilde{b}_v - \sum_{i=1}^{n} k_i\epsilon_i - k_i\epsilon_i + \hat{R}\tilde{b}_\Omega\hat{R}^T\left(\hat{p} - \hat{l}_i\right) + \tilde{R}\hat{R}\tilde{b}_\Omega\hat{R}^T\epsilon_i$$

$$= \hat{R}\left(\tilde{b}_\Omega\hat{R}^T\left(\hat{p} - \hat{l}_i\right) - \tilde{b}_v + \tilde{b}_\Omega\hat{R}^T\epsilon_i\right) - \sum_{i=1}^{n} k_i\epsilon_i - k_i\epsilon_i$$

$$= \hat{R}\left(\tilde{b}_\Omega\hat{R}^T\left(p - l_i\right) - \tilde{b}_v\right) - \sum_{i=1}^{n} k_i\epsilon_i - k_i\epsilon_i$$

$$\dot{\epsilon}_i = \hat{R}\left(\tilde{b}_\Omega R^T\left(p - l_i\right) - \tilde{b}_v\right) - \sum_{i=1}^{n} k_i\epsilon_i - k_i\epsilon_i \tag{4.60}$$

Knowing from (4.53), that $\tilde{b}_U$,$\hat{R}$ and ϵ_i are bounded, and that p and l_i are bounded by assumption, we conclude that $\dot{\epsilon}_i$ s also bounded, thus $\ddot{\mathcal{V}}$ is also bounded, and we conclude that $\dot{\mathcal{V}}$ converges to 0 which also implies that $\epsilon_i, \forall i \in (1, 2, ..., n)$ converge to 0_3. Since $\lim_{t\to\infty}\epsilon_i \to 0_3, \forall i \in (1, 2, ..., n)$ and $\dot{\epsilon}_i$ is bounded, we conclude that $\lim_{t\to\infty}\dot{\epsilon}_i \to 0_3, \forall i \in (1, 2, ..., n)$. As a result, we conclude that when convergence is achieved (4.60) becomes :

$$\lim_{t\to\infty}\hat{R}\left(\tilde{b}_\Omega R^T\left(p - l_i\right) - \tilde{b}_v\right) = 0_3$$

and since $\lim_{t\to\infty}\hat{R} \neq 0_{3\times3}$ we find :

$$\lim_{t\to\infty}\left(\tilde{b}_\Omega R^T\left(p - l_i\right) - \tilde{b}_v\right) = 0_3 \tag{4.61}$$

Next we assume that at least three landmarks, which are also assumed to be available for measurement, form a plane. By substituting their respective position with respect to the inertial frame $\{\mathcal{I}\}$ in (4.61) we obtain :

$$\lim_{t\to\infty}\left(\tilde{b}_\Omega R^T\left(p - l_1\right) - \tilde{b}_v\right) = 0_3 \tag{4.62}$$

$$\lim_{t\to\infty}\left(\tilde{b}_\Omega R^T\left(p - l_2\right) - \tilde{b}_v\right) = 0_3 \tag{4.63}$$

$$\lim_{t\to\infty}\left(\tilde{b}_\Omega R^T\left(p - l_3\right) - \tilde{b}_v\right) = 0_3 \tag{4.64}$$

we next will subtract (4.64) from (4.62) and (4.63), we find :

$$\lim_{t\to\infty}\tilde{b}_\Omega\left(l_3 - l_1\right) = 0_3 \tag{4.65}$$

$$\lim_{t\to\infty}\tilde{b}_\Omega\left(l_3 - l_2\right) = 0_3 \tag{4.66}$$

since $l_3 - l_1$ and $l_3 - l_2$ are noncollinear we conclude that :

$$\lim_{t\to\infty}\tilde{b}_\Omega = 0_{3\times3} \tag{4.67}$$

we replace (4.67) in (4.61), and we get :

$$\lim_{t\to\infty}\tilde{b}_v = 0_3 \tag{4.68}$$

When convergence is attained, (4.55) and (4.56) become :

$$\dot{\tilde{R}} = 0_{3\times3}$$

$$\dot{\tilde{p}} = 0_3$$

which implies that $\tilde{R}$ and $\tilde{p}$ converge to some arbitrary constants $\tilde{R}^*$ and $\tilde{p}^*$, respectively.

The proposed observer is finalized from (4.41), (4.48), and (4.49), knowing that $\dot{\tilde{b}}_U = \dot{b}_U - \dot{\hat{b}}_U \approx -\dot{\hat{b}}_U$ since the variation of b_U is considered far lower compared to the variation of $\hat{b}_U$ with respect to time t :

$$\dot{\hat{X}} = \hat{X}\left(Uy - \hat{b}_U - \Delta\right) \tag{4.69}$$

$$\Delta = -ad_{\hat{X}^{-1}}\left(\mathbb{P}\left(\left(I_{n+4} - \tilde{X}^{-1}\right)A\right)\right) \tag{4.70}$$

$$\dot{\hat{b}}_U = -\mathbb{P}\left(\hat{X}^T\left(I_{n+4} - \tilde{X}^{-1}\right)A\hat{X}^{-T}\right)K'_b \tag{4.71}$$

Algorithm 6 Gradient observer with velocity biases

Data: $k_i > 0 \quad \forall i \in (1,2,...,n)$, $\hat{X}_0 \in SE_{1+n}(3)$, $U_y \in \mathfrak{se}_{1+n}(3)$, $r_i \in \mathbb{R}^{n+4} \quad \forall i \in (1,2,...,n), \hat{b}_U(0) \in \mathfrak{se}_{n+1}(3), k_w > 0, k_v > 0, dt$

Result: $\hat{X}_k$

$k \leftarrow 0$

$K'_b \leftarrow diag(k_w, k_w, k_w, k_v, 0_n^T)$

while 1 **do**

  Get measurements $y_i(k) \quad \forall i \in (1,2,...,n)$ and Uy

  Construct $b_i(k)$ from $y_i(k)$ as in (4.15)

  $\Delta \leftarrow -ad_{\hat{X}_k^{-1}} \sum_{i=1}^{n} k_i \left(\mathbb{P}\left(r_i - \hat{X}_k b_i\right) r_i^T\right)$

  $\hat{X}_{k+1} \leftarrow \hat{X}_k \exp\left(\left(Uy - \hat{b}_U(k) - \Delta\right) dt\right)$

  $\hat{b}_U(k+1) \leftarrow \hat{b}_U(k) - dt\mathbb{P}\left(\hat{X}_k^T \sum_{i=1}^{n} k_i \left(\mathbb{P}\left(r_i - \hat{X}_k b_i\right) r_i^T\right) \hat{X}_k^{-T}\right) K'_b$

  $k \leftarrow k+1$

4.2 Simulation

In this section, we simulate the observer (with biases) in order to test its performance. We consider a vehicle moving at a 10-meter height, with an angular velocity $\omega = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T rad/s$, and a translational velocity $v = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T m/s$. Also, the rotation matrix was initialized as $R(0) = I_3$, and a set of 16 landmarks were randomly chosen following a uniform distribution in a range of $\{-10, 10\}$ meters. Additionally, we consider the biases of the angular and translational velocity measurements as $b_\omega = \begin{bmatrix}-0.02 & 0.02 & 0.01\end{bmatrix}^T$ and $b_v = \begin{bmatrix}0.2 & -0.1 & 0.1\end{bmatrix}^T$, respectively. Moreover, the estimates are initialized as $\hat{R}(0) = I_3$, $\hat{p}(0) = 0_3$, $\hat{b}_\omega = 0_3$, $\hat{b}_v = 0_3$ and $\hat{l}_i = 0_3$ for $i = 1, ..., 16$. Finally, we chose the gains involved in the algorithm as follows : $k_\omega = 0.02$, $k_v = 1$ and $k_i = \frac{5}{22}$ for $i = 1, ..., 16$.

4.2.1 Simulation results

Given the above-mentioned conditions, we get the following results after running the simulation on python :

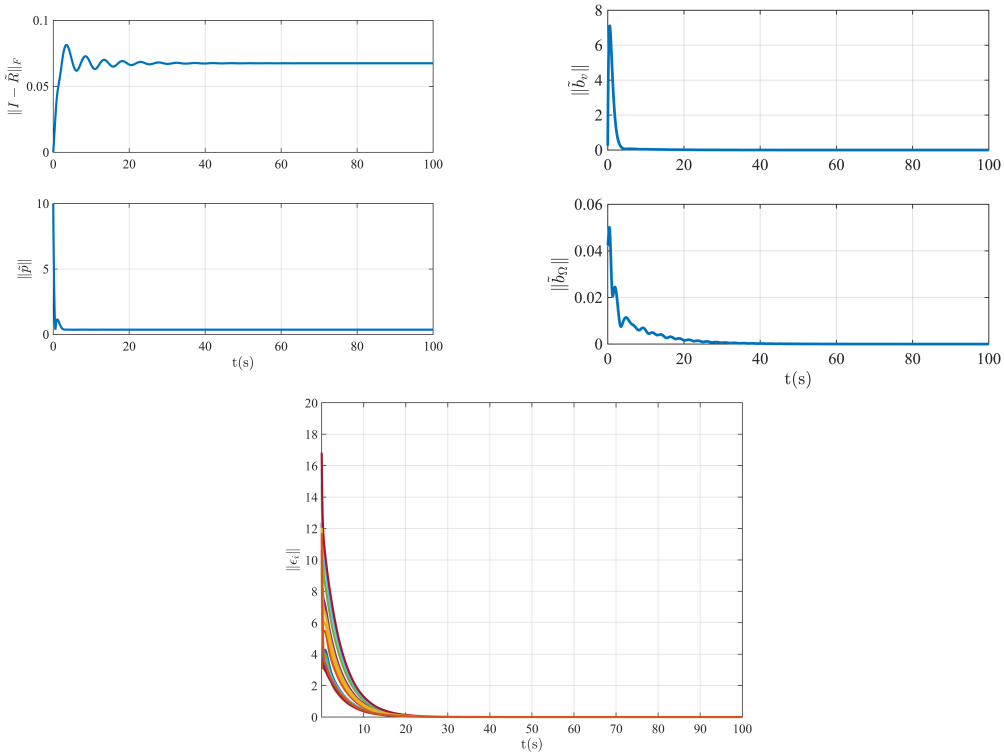


Figure 4.2: Estimation errors of rotation, position, velocity biases, and landmarks with respect to time.

4.2.2 Results discussion

As expected from the previous mathematical analysis, the landmark estimation errors along with the angular and translational velocity biases converge to zero, and the pose estimation error $(\tilde{R}, \tilde{p})$ converges to some constant (R^*, p^*), since the SLAM problem is not observable. We also notice that the position estimation error and the translational velocity bias converge faster to their final values compared to the rotation estimation error, the angular velocity bias and the landmark estimation errors.

By both considering the graphs above and the derivative equations (4.55), (4.56), (4.57), (4.58) and (4.59), we give the following remarks :

- $\|\tilde{R}\|$ converges relatively slower (after more than 20s) because $\dot{\tilde{R}}$ depends on $\tilde{b}_\Omega$ which in turn converges to 0 after more than 20s.

- $\|\tilde{p}\|$ converges almost instantly (after barely 3s), even though its derivative depends on $\tilde{b}_\Omega$ and $\sum k_i \epsilon_i$. We can explain this on the one hand, because $\tilde{b}_\Omega \leq 0.05$, hence its influence on $\dot{\tilde{p}}$ can be considered as negligible, and on the other hand, as we can see in (Fig 4.3), it turns out that $\sum k_i \epsilon_i$ tends to zero before the landmark estimation errors converge to zero (i.e. the landmark estimation errors start cancelling each other before ϵ_i tend to zero). Therefore, $\tilde{p}$ converges once $\sum k_i \epsilon_i$ goes to zero.

- Same as for $\dot{\tilde{p}}$, $\dot{\tilde{b}}_v$ depends on $\sum k_i \epsilon_i$. Therefore, $\tilde{b}_v$ stabilizes once $\sum k_i \epsilon_i$ converges to 0.

- $\tilde{b}_\Omega$ Converges to zeros after more than 20s because it depends on landmark estimation errors, which in turn converge after 20s.
- Since $\dot{\epsilon}_i$ depend on ϵ_i, the landmark estimation errors stabilize once $\dot{\epsilon}_i = \epsilon_i = 0$, which is also illustrated in the landmark estimation errors graph.

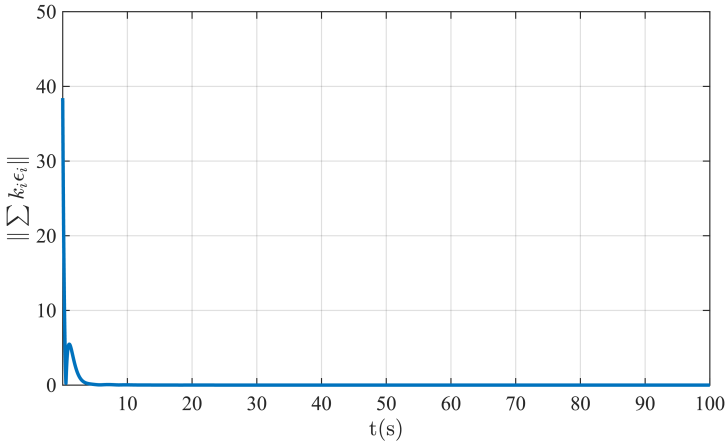


Figure 4.3: Sum of weighted landmarks estimation errors over time

4.3 Conclusion

In this chapter, we discussed the work of Miaomiao Wang and Abdelhamid Tayebi [1] which uses a Lie groups based geometric approach to design a nonlinear observer that solves the SLAM problem, and which has proven to have many advantages over the classical approaches. First, as we saw in the second chapter, EKF-SLAM and PGO-SLAM are based on linear approximations that do not fit the nonlinear nature of the SLAM estimation problem. However, a Lie groups based nonlinear approach can mimic almost perfectly the structure of the SLAM problem. Also, we saw how the position and orientation of the vehicle moving in 3D space was best modeled using the special euclidean group $SE(3)$ and the special orthogonal group $SO(3)$. Additionally, a Lie groups based approach leads to a nonlinear observer that converges globally without depending on the quality of the initial guess, in contrast to approaches that are based on linear approximations. Moreover, the fact that the approach presented in this chapter is deterministic compared to the solutions presented in chapter 2 which are probabilistic makes the Lie groups based SLAM much less computationally expensive. Finally, this work leverages the full potential of Lie theory (at least from a roboticist point of view) as it designs the observer parameter on the Lie algebra (linear space), then it expresses the result on the Lie group (curved space) through the exponential map.

Chapter 5

Robust visual SLAM in presence of landmarks uncertainties

5.1 Introduction

Several problems appear when trying to implement a SLAM method in practice, among these problems we find the varying number of landmarks used for estimation; in this case, new features can enter the state estimation process, as others can be dropped. Another practical problem are faulty measurements, where a sensor can be damaged or can't function properly due to harsh environmental conditions.

In this chapter, we will deal with the above-mentioned two problems that can affect the non-linear observer (with biases) (6) proposed in chapter 4. In the first section, we will give a demonstration to the problem appearing when changing the dimension of the state, and then present some simulation results to better see it's behavior. In the section that follows, we will propose a measurement detector and isolator, to help diagnose measurement tools (Camera and IMU) used to provide feedback for estimation, and then show the simulation results to see the performance of the suggested corrector.

5.2 State re-dimensioning for the SLAM problem

In practice, SLAM methods don't rely on a fixed number of landmarks to do the estimation. Sometimes, the state estimation algorithm drops a landmark during the estimation process, because the sensor doesn't recognize a memorized feature in the scene. While some other times, it adds landmarks to the state to increase the accuracy of the map.

In this section, we will test the performance of the non-linear observer (with biases) (6) presented in chapter 4, when adding or dropping landmarks.

5.2.1 Missing landmark recognition

We assign to each landmark l_i an index i to indicate which feature it corresponds to. We define the following measurements matrix

$$\overline{y}_{1:n} = \begin{bmatrix} y_1 & y_2 & ... & y_n \\ 1 & 2 & ... & n \end{bmatrix}$$

When a landmark i disappears from the scene, the camera returns the following measurements

$$\overline{y}_{1:i-1,i+1,n} = \begin{bmatrix} y_1 & ... & y_{i-1} & y_{i+1} & ... & y_n \\ 1 & ... & i-1 & i+1 & ... & n \end{bmatrix}$$

We can see that indices will allow us to detect the missing landmarks, which will allow us to change the dimension of the estimated state.

Then we denote the new state $\hat{X}^-$, and we define it as

$$\hat{X}^- = \left[\begin{array}{cc|c} \hat{R} & \hat{p} & \hat{L}^- \\ 0_3^T & 1 & 0_{n-1}^T \\ \hline 0_{(n-1)\times n} & 0_{n-1} & I_{n-1} \end{array}\right]$$

where $\hat{L}^- = \begin{bmatrix} \hat{l}_1 & ... & \hat{l}_{i-1} & \hat{l}_{i+1} & ... & \hat{l}_n \end{bmatrix}$

The new input matrix U_y^- is defined as

$$U_y^- = \left[\begin{array}{cc|c} \Omega_y & v_y & \mathrm{v}^- \\ \hline 0_{n\times 3} & 0_n & 0_{n\times(n-1)} \end{array}\right]$$

where $\mathrm{v}^- = \begin{bmatrix} v_1 & ... & v_{i-1} & v_{i+1} & ... & v_n \end{bmatrix}$

And the new velocity biases matrix b_U^- is defined as

$$b_U^- = \left[\begin{array}{cc|c} b_\Omega & b_v & 0_{3\times(n-1)} \\ \hline 0_{n\times 3} & 0_n & 0_{n\times(n-1)} \end{array}\right]$$

5.2.2 State augmentation

The SLAM algorithm starts with few landmarks, and continues to add others along its motion. Let's say that the estimated state $\hat{X}$ at instant t has a dimension of $(n+4)\times(n+4)$, where the upper left 4×4 matrix defines the pose of the robot, while the upper right $3\times n$ elements represent the estimated locations of n distinguishable landmarks. Let's also say that at instant t+dt, the camera returned an additional measurement of a new feature, and the frontend algorithm assigned the index $n+1$ to it. In this case, the measurement matrix would be defined as

$$\overline{y}_{1:n+1} = \begin{bmatrix} y_1 & y_2 & ... & y_n & y_{n+1} \\ 1 & 2 & ... & n & n+1 \end{bmatrix}$$

In this case we have to change the dimension of the estimated state matrix, where we initialize the estimated pose of the new landmark for the observer to estimate its true location using measurements.

Then we denote the new state $\hat{X}^+$, and we define it as

$$\hat{X}^+ = \left[\begin{array}{cc|c} \hat{R} & \hat{p} & \hat{L}^+ \\ 0_3^T & 1 & 0_{n+1}^T \\ \hline 0_{(n+1)\times n} & 0_{n+1} & I_{n+1} \end{array}\right]$$

where $\hat{L}^+ = \begin{bmatrix} \hat{l}_1 & ... & \hat{l}_n & \hat{l}_{n+1} \end{bmatrix}$

The new input matrix U_y^+ is defined as

$$U_y^+ = \left[\begin{array}{cc|c} \Omega_y & v_y & v^+ \\ \hline 0_{(n+2)\times 3} & 0_{n+2} & 0_{(n+2)\times(n+1)} \end{array}\right]$$

where $\mathrm{v}^+ = \begin{bmatrix} v_1 & ... & v_n & v_{n+1} \end{bmatrix}$

And the new velocity biases matrix b_U^+ is defined as

$$b_U^+ = \left[\begin{array}{cc|c} b_\Omega & b_v & 0_{3\times(n+2)} \\ \hline 0_{(n+2)\times 3} & 0_{n+2} & 0_{(n+2)\times(n+1)} \end{array}\right]$$

5.2.3 Simulation

We consider a vehicle moving under the same conditions as the simulation section of chapter 4. We initialize the estimated velocity biases, landmarks, and the vehicle's pose in the same way as in chapter 4, while $\hat{R}(0) = \exp(0.2\pi u^\wedge)$ where $u = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. In order to visualize the behavior of the observer, we drop a landmark at $t = 60s$.

5.2.3.1 Simulation results

Given the above conditions, we obtain the following results after running the simulation on python :

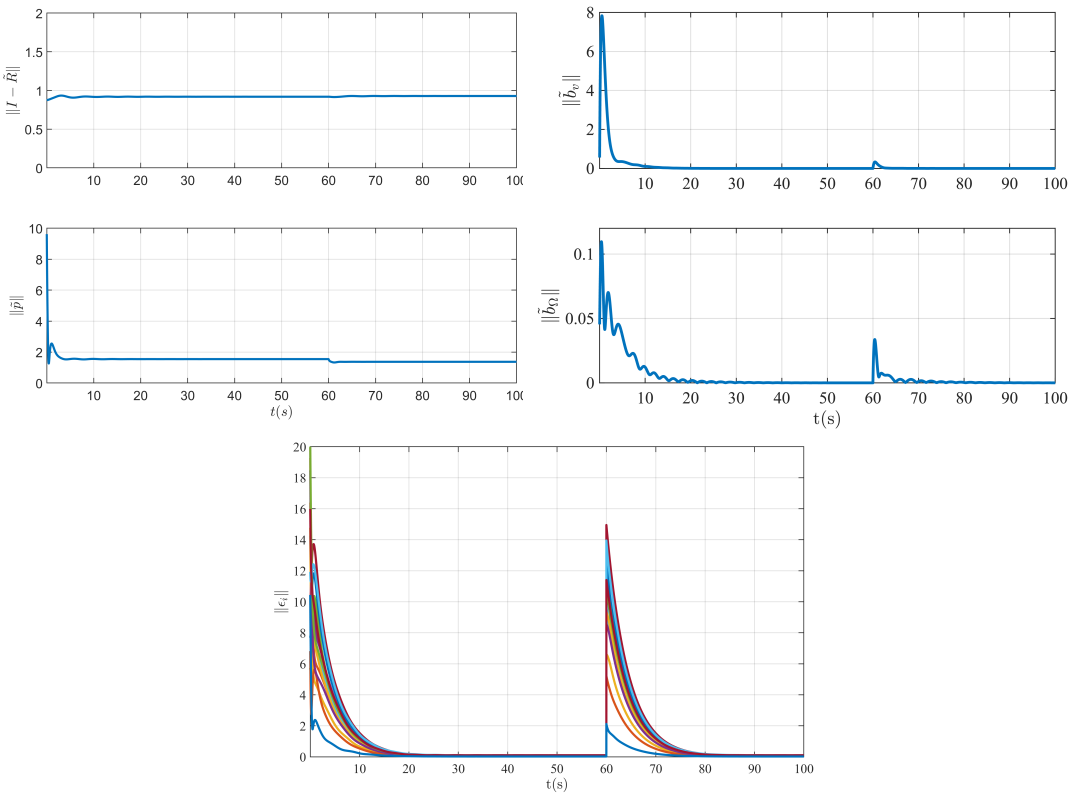


Figure 5.1: Estimation errors of rotation, position, velocity biases, and landmarks when dropping a landmark.

5.2.3.2 Results discussion

As it is anticipated, changing the dimension of the state matrix $\hat{X}_k$ will not affect the stability of the system, as we can see from (Fig. 5.1) all estimation errors return to the stability point after a certain time duration. Also, we can see that the convergence time of landmark's estimation error after changing the state matrix, which is equal to $20 seconds$, is similar to the convergence time when starting the estimation process. As a result, we conclude that successive changes in state must be done after a time duration equal to the convergence time, to obtain a correct sparse mapping of the environment.

5.3 Fault detection and isolation block for the SLAM problem

Some of the major problems that can affect the results of any kind of state observer are faulty sensor data, where these incorrect measurements can be the result of environmental conditions (e.g. the moist accumulated on the lens of a camera can change its intrinsic

parameters, resulting in false measurements), or sensor-related problems (e.g. a damaged IMU). Therefore, several researches had been made to improve information diagnosis, among them, there is a solution based on sensor redundancy suggested in [70]. This method, which is explained in detail in [71], detects wrong data based on what the majority of sensors return. Although, using multiple sensors to diagnose measurements can be accurate, it can also be financially very expensive, and can only be implemented in critical applications where fault is highly intolerable and budget is less important than quality, like military applications. Clearly, making information diagnosis cheaper, will help to increase the robustness of SLAM systems dedicated for low budget applications.

In our work, we will present a solution to detect false measurements for V-SLAM applications, by combining measurements from an IMU and a camera. This solution can help increase the efficiency of the non-linear observer presented in (6), when used in practice, and especially when multiple sensors of the same type cannot be used to reduce the budget spent on hardware.

In order to visualize the efficiency of the proposed FDI, we present the results of a simulation at the end of the chapter.

5.3.1 Mathematical definitions

For two n-dimensional vectors $x = \begin{bmatrix} x_1 & x_2 & ... & x_n \end{bmatrix}^T$ and, $\delta = \begin{bmatrix} \delta_1 & \delta_2 & ... & \delta_n \end{bmatrix}^T$ we define the function $M : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n \times n}$, as

$$M(x, \delta) = diag\begin{pmatrix} H(x_1, \delta_1) & H(x_2, \delta_2) & ... & H(x_n, \delta_n) \end{pmatrix} \tag{5.1}$$

with $H : \mathbb{R} \times \mathbb{R} \to \{0, 1\}$ is defined as

$$H(x_i, \delta_i) = \begin{cases} 1, & x_i > \delta_i \\ 0, & x_i \leq \delta_i \end{cases} \tag{5.2}$$

5.3.2 FDI presentation

We introduce the selection matrix $\Gamma = diag\begin{pmatrix} \alpha_1 & \alpha_2 & ... & \alpha_n \end{pmatrix} \in \mathbb{R}^{n \times n}$, where n is the number of landmarks available for measurement, as:

$$\Gamma = H\left(\frac{n - n_F}{n}, \delta_{IMU}\right) M\left(d, \delta_{camera}\right) \tag{5.3}$$

where $\delta_{IMU} \in [0, 1]$ depends on the quality of the camera and the IMU, and defined as the percentage of the accurately measured landmarks among the total number of landmarks. This threshold allows us to determine whether the IMU is giving false measurements or not; when the number of false measurements $n_F \in \mathbb{N}$ is close enough to the total number of landmarks n, we can say that the IMU is giving false data, as it is very unlikely that a camera makes a large percentage of wrong measurements. The vector $d \in \mathbb{R}^n$ is defined as a vector containing all the euclidean norms between the predicted measurements denoted $y_p = \begin{bmatrix} y_{p,1} & y_{p,2} & ... & y_{p,n} \end{bmatrix} \in \mathbb{R}^{3 \times n}$, and the real measurements

denoted $y = \begin{bmatrix} y_1 & y_2 & ... & y_n \end{bmatrix} \in \mathbb{R}^{3\times n1}$

$$d = \begin{bmatrix} \|y_{p,1} - y_1\| & \|y_{p,2} - y_2\| & ... & \|y_{p,n} - y_n\| \end{bmatrix}^T \tag{5.4}$$

$\delta_{camera} = \begin{bmatrix} \delta_{camera,1} & \delta_{camera,2} & ... & \delta_{camera,n} \end{bmatrix}^T \in \mathbb{R}^n$ is the threshold that decides whether the measurement is wrong or not; if the euclidean norm between the i^{th} predicted measurement and the i^{th} real measurement, denoted $d_i = \|y_{p,i} - y_i\|$, is above a threshold $\delta_{camera,i}$, which depends on how much we tolerate the error, it means that the measurement is false.

The Heaviside function will assure an alternation between the predicted measurement and the real measurement when a measurement problem occurs. The first function $H\left(\frac{n - n_F}{n}, \delta_{IMU}\right)$ allows the state observer to ignore temporarily the IMU while it is considered to be wrong, and that is determined by the number of false measurements. Meanwhile, the function $M\left(d, \delta_{camera}\right)$ returns a matrix, that helps to select the correct real measurements and replace the wrong ones with the predicted measurements, all that while assuming that the predicted measurement is within tolerable error.

Finally, we propose the corrected measurement $y_c = \begin{bmatrix} y_{c,1} & y_{c,2} & ... & y_{c,n} \end{bmatrix} \in \mathbb{R}^{3\times n}$ as

$$y_c = y_p\Gamma + y(I_n - \Gamma) \tag{5.5}$$

knowing from (5.5) that:

$$y_{c,i} = \alpha_i y_{p,i} + (1 - \alpha_i) y_i \tag{5.6}$$

We can detect which measurement is wrong by observing the state of α_i; when $\alpha_i = 1$ we deduce that the measurement of the i^{th} landmark is incorrect.

Next we will present the correction algorithm :

Algorithm 7 FDI($y_p, y, \delta_{IMU}, \delta_{camera}$)

Data: $n \in \mathbb{N}$
Result: y_c

```
Compute d as in (5.4)
n_F ← 0
for i ∈ range(n) do
    if ‖y_{p,i} − y_i‖ > δ_{camera,i} then
        n_F ← n_F + 1
Compute Γ as in (5.3)
y_c ← y_pΓ + y(I_n − Γ)
```

5.3.3 Fault-tolerant Nonlinear observer presentation

In this section, we will present the algorithm of the non-linear observer (6) when using the FDI, along with a block diagram that illustrates the flow of data between the observer and the FDI :

[1]Note that the predicted measurement is estimated using odometry (through the IMU) and the real measurements are made from the camera.

Algorithm 8 Fault-tolerant non-linear observer

Data: $k_i > 0 \quad \forall i \in (1,2,...,n)$, $\hat{X}_0 \in SE_{1+n}(3)$, $U \in \mathfrak{se}_{1+n}(3)$, $r_i \in \mathbb{R}^{n+4} \quad \forall i \in (1,2,...,n), \hat{b}_U(0) \in \mathfrak{se}_{n+1}(3), k_w > 0, k_v > 0, \delta_{IMU} \in \mathbb{R}, \delta_{camera} \in \mathbb{R}^n, dt$

Result: $\hat{X}_k$

$k \leftarrow 0$

$K'_b \leftarrow diag(k_w, k_w, k_w, k_v, 0_n^T)$

while 1 **do**

  Get measurements $y_i(k) \quad \forall i \in (1,2,...,n)$ and Uy

  $b_{p,i}(k) \leftarrow \hat{X}_k^{-1} r_i, \quad \forall i \in (1,2,...,n)$

  Extract $y_{p,i}$ from $b_{p,i}$

  $y(k) \leftarrow \begin{bmatrix} y_1(k) & y_2(k) & ... & y_n(k) \end{bmatrix}$

  $y_p(k) \leftarrow \begin{bmatrix} y_{p,1}(k) & y_{p,2}(k) & ... & y_{p,n}(k) \end{bmatrix}$

  $y_c(k) \leftarrow FDI\left(y_p(k), y(k), \delta_{IMU}, \delta_{camera}\right)$

  $b_i(k) \leftarrow \begin{bmatrix} y_{c,i}^T(k) & 1 & -e_i^T \end{bmatrix}^T, \forall i \in (1,2,...,n)$

  $\Delta \leftarrow -ad_{\hat{X}_k^{-1}} \sum_{i=1}^n k_i \left(\mathbb{P}\left(r_i - \hat{X}_k b_i\right) r_i^T\right)$

  $\hat{X}_{k+1} \leftarrow \hat{X}_k \exp\left(\left(Uy - \hat{b}_U(k) - \Delta\right) dt\right)$

  $\hat{b}_U(k+1) \leftarrow \hat{b}_U(k) - dt\mathbb{P}\left(\hat{X}_k^T \sum_{i=1}^n k_i \left(\mathbb{P}\left(r_i - \hat{X}_k b_i\right) r_i^T\right) \hat{X}_k^{-T}\right) K'_b$

  $k \leftarrow k+1$

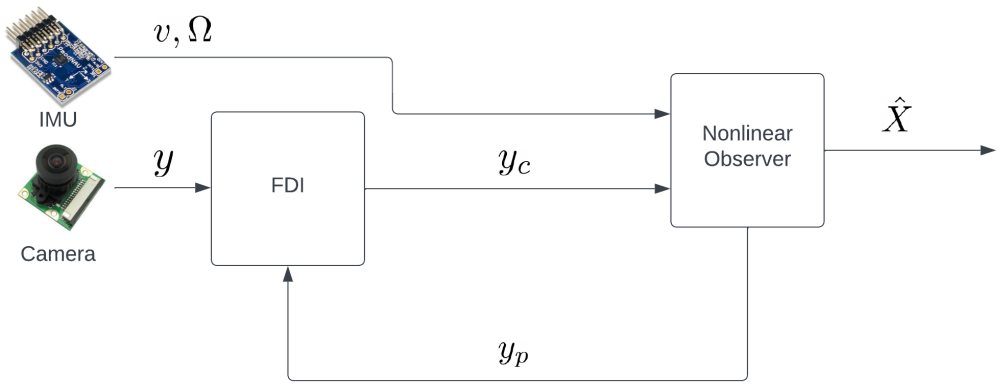


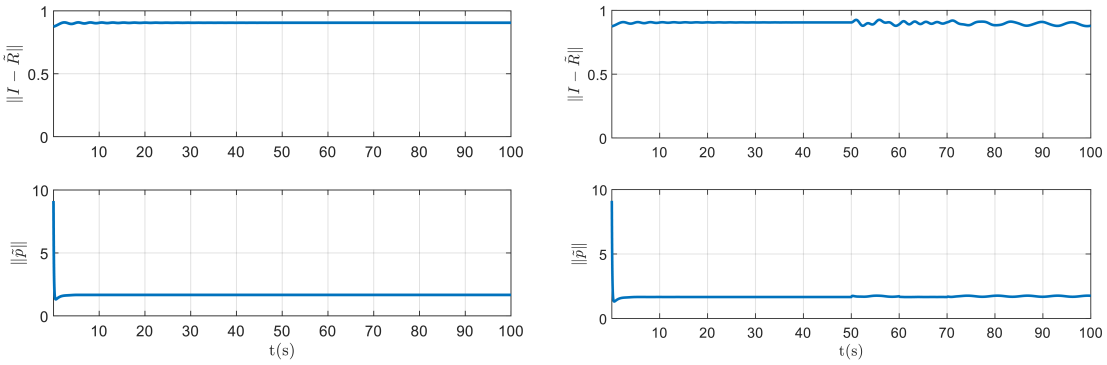
Figure 5.2: Data flow between the FDI and the observer

5.3.4 Simulation

In this section, we simulated the above-mentioned observers (8) and (6), while introducing measurement errors of 3 meters in the 5^{th} landmark for 10 *seconds* starting from $t = 50s$, and the 10^{th} for 30 *seconds* starting from $t = 70s$. We consider a vehicle moving under the same conditions as in the simulation section in chapter 4. We initialize the estimated velocity biases and the vehicles pose in the same way as in chapter 4, while $\hat{R}(0) = \exp(0.2\pi u^\wedge)$ where $u = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. In order to effectively diagnose the IMU, we increase the number of landmarks to 45, which are also chosen randomly using a uniform distribution between -10 and 10 *meters*, and initialized in the same way as in chapter 4. We choose, $\delta_{IMU} = 0.1$, and $\delta_{camera,i} = 0.1$ *meters* $, \forall i \in (1,2,...,45)$.

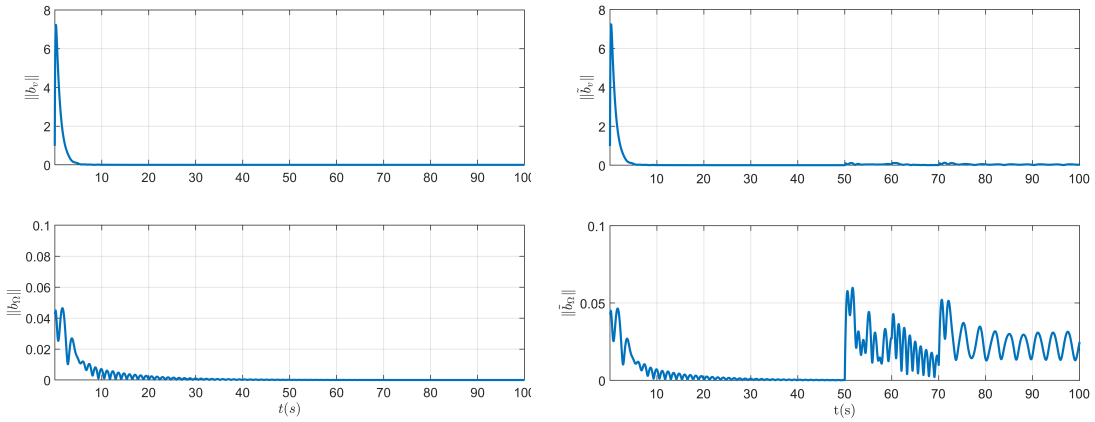
5.3.4.1 Simulation results

Given the above conditions, we obtain the following results after running the simulation on python :



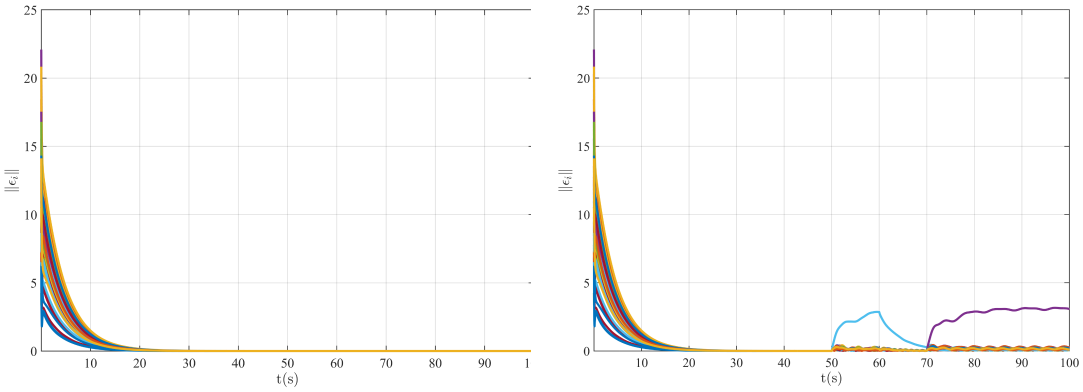
(a) With FDI (b) Without FDI

Figure 5.3: Estimation errors of rotation and position.



(a) With FDI (b) Without FDI

Figure 5.4: Estimation errors of velocity biases.



(a) With FDI (b) Without FDI

Figure 5.5: Estimation errors of landmarks.

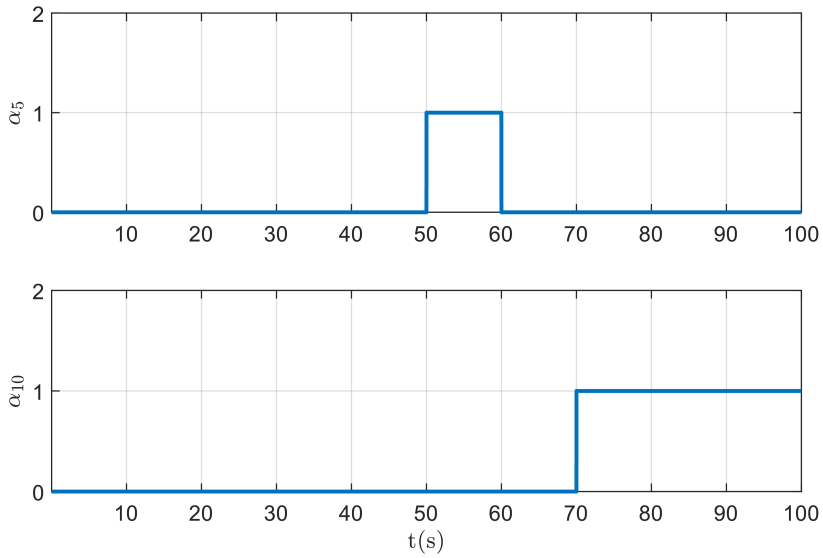


Figure 5.6: The correction gains α_5 and α_{10} of the measurements of the 5^{th} and 10^{th} landmarks, respectively.

5.3.4.2 Results discussion

Before using the FDI, we can clearly see by looking at (Fig. 5.3b), (Fig. 5.4b), and (Fig. 5.5b) that the estimation errors destabilize, because of the faulty measurements caused by the camera. On the other hand, using an FDI can help to temporarily switch to odometry predictions while the measurements are considered to be wrong.(Fig. 5.6) depicts that the coefficient α_5 goes from 0 to 1 at $t = 50s$ which indicates that it took the prediction, and ignored the false measurement. After 10 *seconds*, α_5 goes back to 0 where the measurement of the 5^{th} landmark is taken again into consideration. At $t = 70s$, the error introduced in the 10^{th} measurement is detected using α_{10}, where it also helps to ignore the false measurement of the 10^{th} landmark while it is wrong.

5.4 Conclusion

Throughout this chapter, we tried to provide some solutions to two major problems that a SLAM method can face when implemented in practice. The first one is when a change in the state's dimension happens, either from a measurement of a completely new feature in the scene viewed by the camera, or because of a missing measurement, because a known feature escapes the field of view of the camera. The second solution deals with the problem of faulty measurements, where we proposed a corrector that takes the predicted measurement temporarily into account while the measurement is considered to be wrong. The proposed solutions proved to be useful in practice, as the first one allows the algorithm to introduce new feature or drop old ones. Meanwhile, the FDI helps to detect false measurements and to keep the estimated state stable.

Chapter 6

General Conclusion

In this project, we attempted to understand a Matrix Lie groups based nonlinear observer for solving the SLAM problem proposed by [1]. After a general introduction, we provided a brief overview of SLAM in chapter 2, in order to make the reader appreciate the kind of problem that SLAM solves. Then, in chapter 3, we tried to provide the mathematical background needed to tackle the work of [1], by introducing the reader to the main concepts of Lie theory that are used in state estimation, and especially in robotics. Subsequently, in chapter 4, we presented the work of [1], we provided a more thorough mathematical derivation of the observer as we demonstrated all the assumptions and mathematical tools needed to design it. Afterwards, we implemented the observer and simulated its behavior in presence of velocity biases, and we also tried to provide a mathematical interpretation of the obtained results.

Our practical contribution was presented in chapter 5, where we endowed the observer with two practical features that we believe to be useful in real world applications. First, we added the ability of dynamically changing the dimension of the system, as it is common in practice to add new landmarks to the estimation process as the vehicle explores the environment on the one hand, and to remove landmarks from the state matrix in case of absence of measurements of a certain landmark because of harsh environmental conditions on the other hand. Second, we implemented a fault detection and isolation block. As the name suggests, this block detects faulty measurements, isolates them and corrects them. It consists of two switches, the first one determines whether it is the IMU or the camera that is making faulty measurements, by considering the percentage of correctly measured landmarks among the total number of landmarks, and comparing it to a certain threshold that depends on the quality of the available sensors. The second switch determines which landmarks are wrongly measured, in order to correct their measurements, by considering the euclidean distance between the measured positions of the landmarks (through camera) and the predicted positions of the landmarks (through IMU), and comparing it to a certain threshold that also depends on the application.

Bibliography

[1] M. Wang and A. Tayebi, "Geometric nonlinear observer design for slam on a matrix lie group," in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 1488–1493.

[2] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: A survey," *Artificial intelligence review*, vol. 43, no. 1, pp. 55–81, 2015.

[3] B. Alsadik and S. Karam, "The simultaneous localization and mapping (slam)-an overview," *Surv. Geospat. Eng. J*, vol. 2, pp. 34–45, 2021.

[4] P. van Goor, R. Mahony, T. Hamel, and J. Trumpf, "A geometric observer design for visual localisation and mapping," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, 2019, pp. 2543–2549.

[5] S. Labsir, G. Pages, and D. Vivet, "Lie group modelling for an ekf-based monocular slam algorithm," *Remote Sensing*, vol. 14, no. 3, p. 571, 2022.

[6] A. Barrau and S. Bonnabel, "An ekf-slam algorithm with consistency properties," *arXiv preprint arXiv:1510.06263*, 2015.

[7] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.

[8] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[9] C. Cadena, L. Carlone, H. Carrillo, *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[10] N. Ayache and O. D. Faugeras, "Building, registrating, and fusing noisy visual maps," *The International Journal of Robotics Research*, vol. 7, no. 6, pp. 45–65, 1988.

[11] R. Chatila and J.-P. Laumond, "Position referencing and consistent world modeling for mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, IEEE, vol. 2, 1985, pp. 138–145.

[12] J. L. Crowley, "World modeling and position estimation for a mobile robot using ultrasonic ranging.," in *ICRA*, vol. 89, 1989, pp. 674–680.

[13] H. F. Durrant-Whyte, "Uncertain geometry in robotics," *IEEE Journal on Robotics and Automation*, vol. 4, no. 1, pp. 23–31, 1988.

[14] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot.," in *IROS*, vol. 3, 1991, pp. 1442–1447.

[15] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The international journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[16] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous robot vehicles*, Springer, 1990, pp. 167–193.

[17] H. Durrant-Whyte, D. Rye, and E. Nebot, "Localization of autonomous guided vehicles," *Robotics Research*, pp. 613–625, 1996.

[18] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017.

[19] G. Reitmayr, T. Langlotz, D. Wagner, *et al.*, "Simultaneous localization and mapping for augmented reality," in *2010 International Symposium on Ubiquitous Virtual Reality*, 2010, pp. 5–8. DOI: 10.1109/ISUVR.2010.12.

[20] A. Yao, "Teaching robots presence: What you need to know about slam," *Comet Labs Research Team. https://blog. cometlabs. io/teaching-robotspresence-whatyou-need-to-know-about-slam-9bf0ca037553 (accessed 30 August 2020)*, 2017.

[21] Y. Nava Chocron, *Visual-lidar slam with loop closure*, 2019.

[22] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii," *IEEE robotics & automation magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[23] R. Szeliski, "Structure from motion," in *Computer Vision*, Springer, 2011, pp. 303–334.

[24] J. Procházková and D. Martišek, "Notes on iterative closest point algorithm," in *Proc. in 17th Conference on Applied Mathematics*, 2018, pp. 876–884.

[25] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, IEEE International Conference on*, IEEE Computer Society, vol. 3, 2003, pp. 1403–1403.

[26] 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.

[27] T. Whelan, M. Kaess, J. J. Leonard, and J. McDonald, "Deformation-based loop closure for large scale dense rgb-d slam," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 548–555.

[28] E. H. C. Harik, A. Korsaeth, *et al.*, "Combining hector slam and artificial potential field for autonomous navigation inside a greenhouse," *Robotics*, vol. 7, no. 2, p. 22, 2018.

[29] Y. Cheng and G. Y. Wang, "Mobile robot navigation based on lidar," in *2018 Chinese Control And Decision Conference (CCDC)*, IEEE, 2018, pp. 1243–1246.

[30] R. K. Megalingam, C. R. Teja, S. Sreekanth, and A. Raj, "Ros based autonomous indoor navigation simulation using slam algorithm," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 7, pp. 199–205, 2018.

[31] S. Saat, W. Abd Rashid, M. Tumari, and M. Saealal, "Hectorslam 2d mapping for simultaneous localization and mapping (slam)," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1529, 2020, p. 042 032.

[32] X. Zhang, J. Lai, D. Xu, H. Li, and M. Fu, "2d lidar-based slam and path planning for indoor rescue using mobile robots," *Journal of Advanced Transportation*, vol. 2020, 2020.

[33] J. M. Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2d slam techniques available in robot operating system," in *2013 IEEE international symposium on safety, security, and rescue robotics (SSRR)*, IEEE, 2013, pp. 1–6.

[34] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, IEEE, Nov. 2011.

[35] K. Tertychny, D. Krivolapov, S. Karpov, and A. Khoperskov, "Slam method: Reconstruction and modeling of environment with moving objects using an rgbd camera," in *CEUR Workshop Proceedings*, vol. 2254, 2018, pp. 274–281.

[36] H. Alismail, L. D. Baker, and B. Browning, "Continuous trajectory estimation for 3d slam from actuated lidar," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 6096–6101.

[37] D. Droeschel and S. Behnke, "Efficient continuous-time slam for 3d lidar-based online mapping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 5000–5007.

[38] N. Friedman and J. Y. Halpern, "A qualitative markov assumption and its implications for belief change," *arXiv preprint arXiv:1302.3578*, 2013.

[39] A. P. Dawid, "Conditional independence in statistical theory," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 41, no. 1, pp. 1–15, 1979.

[40] D. Zwillinger and S. Kokoska, *CRC standard probability and statistics tables and formulae*. Crc Press, 1999.

[41] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[42] D. Orbea, J. Moposita, W. G. Aguilar, M. Paredes, G. León, and A. Jara-Olmedo, "Math model of uav multi rotor prototype with fixed wing aerodynamic structure for a flight simulator," in *International conference on augmented reality, virtual reality and computer graphics*, Springer, 2017, pp. 199–211.

[43] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[44] N. Sünderhauf and P. Protzel, "Towards a robust back-end for pose graph slam," in *2012 IEEE international conference on robotics and automation*, IEEE, 2012, pp. 1254–1261.

[45] D. Fox, W. Burgard, and S. Thrun, "Markov localization for reliable robot navigation and people detection," in *Sensor Based Intelligent Robots*, Springer, 1999, pp. 1–20.

[46] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[47] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002.

[48] L. Zhao, S. Huang, and G. Dissanayake, "Linear slam: A linear solution to the feature-based and pose graph slam based on submap joining," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 24–30.

[49] M. Kaneko, K. Iwami, T. Ogawa, T. Yamasaki, and K. Aizawa, "Mask-slam: Robust feature-based monocular slam by masking using semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 258–266.

[50] G. Wang and Y. Chen, "Robust feature matching using guided local outlier factor," *Pattern Recognition*, vol. 117, p. 107 986, 2021.

[51] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme," in *2010 ieee intelligent vehicles symposium*, IEEE, 2010, pp. 486–492.

[52] M. Brown, R. Szeliski, and S. Winder, "Multi-image matching using multi-scale oriented patches," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, IEEE, vol. 1, 2005, pp. 510–517.

[53] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous obstacle avoidance and maneuvering on a vision-guided mav using on-board processing," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 2472–2477.

[54] S. Weiss, M. Achtelik, L. Kneip, D. Scaramuzza, and R. Siegwart, "Intuitive 3d maps for mav terrain exploration and obstacle avoidance," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1, pp. 473–493, 2011.

[55] H. Alvarez, L. M. Paz, J. Sturm, and D. Cremers, "Collision avoidance for quadrotors with a monocular camera," in *Experimental Robotics*, Springer, 2016, pp. 195–209.

[56] R. Ait-Jellal and A. Zell, "Outdoor obstacle avoidance based on hybrid visual stereo slam for an autonomous quadrotor mav," in *2017 European Conference on Mobile Robots (ECMR)*, IEEE, 2017, pp. 1–8.

[57] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proceedings. 1985 IEEE international conference on robotics and automation*, IEEE, vol. 2, 1985, pp. 116–121.

[58] H. Tibebu, J. Roche, V. De Silva, and A. Kondoz, "Lidar-based glass detection for improved occupancy grid mapping," *Sensors*, vol. 21, no. 7, p. 2263, 2021.

[59] K. M. Wurm, C. Stachniss, and G. Grisetti, "Bridging the gap between feature-and grid-based slam," *Robotics and Autonomous Systems*, vol. 58, no. 2, pp. 140–148, 2010.

[60] J. Stillwell, *Naive lie theory*. Springer Science & Business Media, 2008.

[61] R. Howe, "Very basic lie theory," *The American Mathematical Monthly*, vol. 90, no. 9, pp. 600–623, 1983.

[62] J. Sola, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018.

[63] T. D. Barfoot, *State estimation for robotics*, 2021.

[64] E. Eade, "Lie groups for 2d and 3d transformations," *URL http://ethaneade. com/lie. pdf, revised Dec*, vol. 117, p. 118, 2013.

[65] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[66] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 1: Classical Results and Geometric Methods*. Springer Science & Business Media, 2009.

[67] L. M. Paz, J. E. Guivant, J. D. Tardós, and J. Neira, "Data association in o (n) for divide and conquer slam.," in *Robotics: Science and Systems*, vol. 3, 2007, p. 281.

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

[69] J. Duchi, "Properties of the trace and matrix derivatives," *Available electronically at https://web. stanford. edu/~ jduchi/projects/matrix_prop. pdf*, 2007.

[70] V. Ricquebourg, M. Delafosse, L. Delahoche, B. Marhic, A. Jolly-Desodt, and D. Menga, "Fault detection by combining redundant sensors: A conflict approach within the tbm framework," *Cognitive Systems with Interactive Sensors, COGIS*, 2007.

[71] A. M. Flynn, "Redundant sensors for mobile robot navigation," 1985.