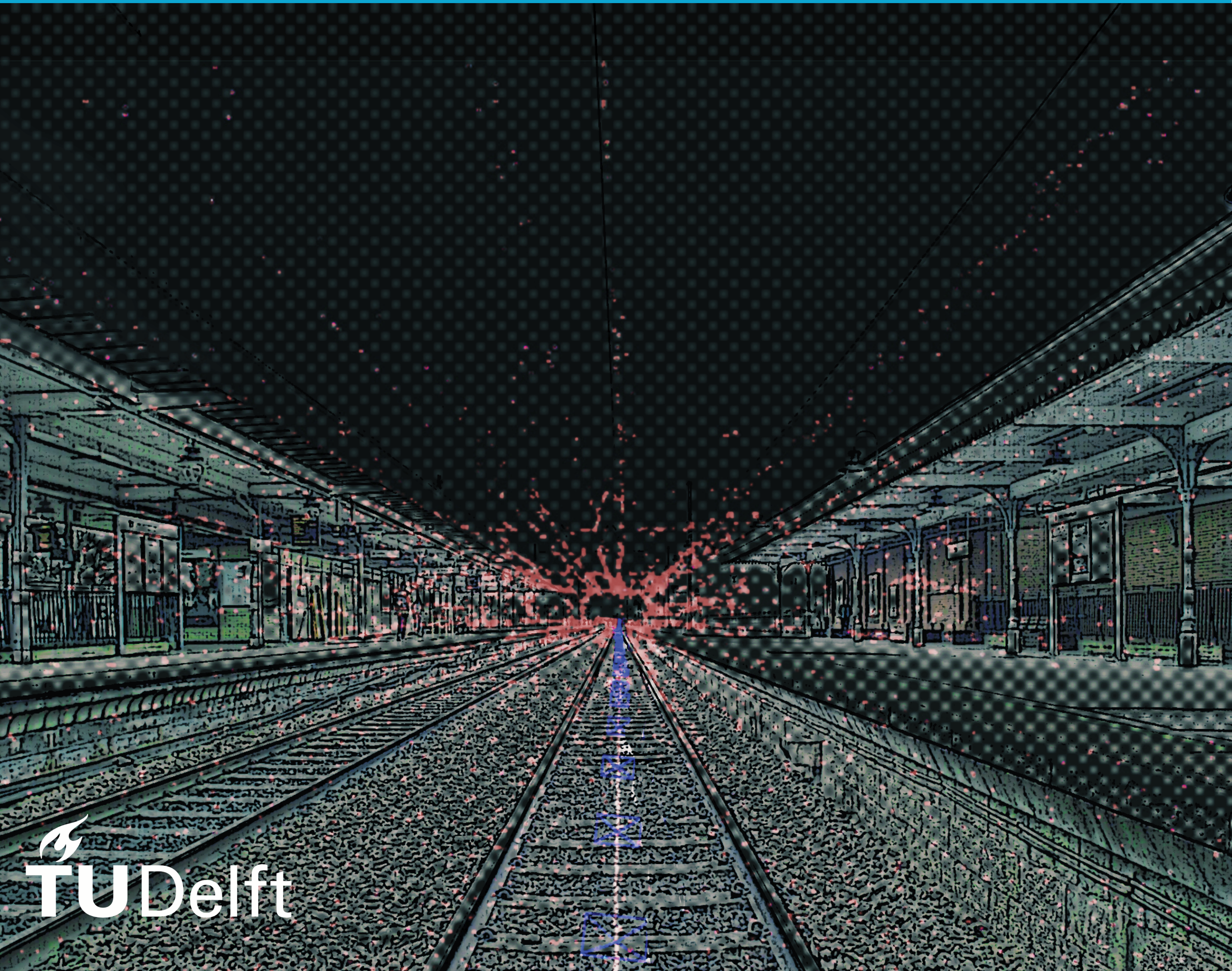
Trajectory Improvement of Railway Mobile Mapping System using Monocular Visual Inertial SLAM

Yashar Habibzadeh

MSc. Thesis Report - March 2023



TU Delft

Delft University of Technology
Faculty of Aerospace Engineering

Trajectory Improvement of Railway Mobile Mapping System using Monocular Visual Inertial SLAM

Yashar Habibzadeh

Thesis Committee:

Dr. ir. E.J.O Schrama	TU Delft, Chair
Dr. ir. W.van der Wal	TU Delft, Daily Supervisor
Dr. Neda Sepasian	Fugro, Supervisor
Dr. Alireza Amiri-Simkooei	TU Delft, External Committee Member

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Aerospace Engineering

Delft, March 2023

Abstract

Nowadays, Mobile Mapping System (MMS) have been widely used in railway, with integrated Inertial Navigation System (INS)/Global Navigation Satellite System (GNSS) system as the common approach. For environments where GNSS signals become unavailable, additional aiding sources need to be considered to preserve the quality of measurement. Typically, in most applications, a Distance Measurement Indicator (DMI) is combined with the INS/GNSS system to ensure the desired accuracy. Nevertheless, as this approach is physically less feasible for railway, alternative solutions would be preferred. To bridge this gap, we investigate the applicability of Simultaneous Localization and Mapping (SLAM) integration with INS/GNSS for the case of a railway MMS. In particular, we aim to propose a solution for adapting a Monocular Visual (Inertial) SLAM method for railway application and further evaluate our solution using real-world data based on the RILA system (Fugro's rail MMS). Accordingly, this dissertation consists of the following three research activities.

Firstly, we have conducted a literature review on the existing monocular Visual Inertial SLAM methods to identify the technique which fulfils the key requirements of the rail application. Considering the results of the study, we selected the ORB-SLAM3 method and proposed an end-to-end pipeline that covers all the phases to adapt it for RILA system.

Secondly, the impact of adapting ORB-SLAM3 technique on performance of the trajectory estimation has been evaluated using two criteria: Absolute Position Error (APE) and Relative Position Error (RPE). Accordingly, a case study using RILA dataset was developed for the experimental evaluations. In this case study, we have simulated a scenario in which the train entered the station, stayed there stationary for 1 minute, and then left the station slowly and manually inserted GNSS blockages before and after the station. Furthermore, a ground-truth trajectory was generated to evaluate the quality of the estimated SLAM-based trajectory. The results revealed that both APE and RPE increases with significant fluctuations in the first 15 seconds due to the lack of SLAM initialization time. Therefore, we introduced an strategy for fine-tuning the accuracy by allocating sufficient time for SLAM initialization phase. The APE and RPE were significantly reduced after fine-tuning and the expected estimated error at each time was equal to 4.3% of the travelled path length.

Finally, we presented the effect of using Zero Velocity Update (ZUPT) signals as aiding information on the accuracy of the estimated trajectory in the areas with poor GNSS coverage. Consequently, we manually inserted the extracted ZUPT signals to the INS/GNSS integration process and then compared the positional accuracy of the generated trajectories with and without this information. The results show that the estimated positional accuracy was improved by 30% with only 51 seconds of stationary condition in our case study.

Overall, based on the obtained results from the evaluations, it is possible to claim that our proposed ORB-SLAM3 technique has great potential to improve the estimated positional accuracy and in particular can be used as a standalone ZUPT detector in railway application.

Contents

List of Figures

List of Tables

Acronyms

MMS	Mobile Mapping System	v
GNSS	Global Navigation Satellite System	v
INS	Inertial Navigation System	v
DMI	Distance Measurement Indicator	v
IMU	Inertial Measuremnet Unit	x
SLAM	Simultaneous Localization and Mapping	v
LIDAR	Light Detection and Ranging	x
RADAR	Radio Detection and Ranging	2
SONAR	Sound Navigation and Ranging	2
V-SLAM	Visual Simultaneous Localization And Mapping	2
VO	Visual Odometry	2
VI-SLAM	Visual Inertial Simultaneous Localization And Mapping	2
VIO	Visual Inertial Odometry	2
EKF	Extended Kalman Filter	13
FAST	Feature from Accelerated Segment	xiii
SIFT	Scale Invariant Feature Transform	21
BRIEF	Binary Robust Independent Elementary Features	xiii
ORB	Oriented Feature from Accelerated Segment (FAST) and Rotated Binary Robust Independent Elementary Features (BRIEF)	21
SURF	Speeded Up Robust Features	22
RANSAC	Random Sampling Consensus	24
BA	Bundle Adjustment	27
ZUPT	Zero Velocity Update	v
MSCKF	Multi State Constraint Kalman Filter	34
SWF	Sliding Window Filter	34
ROVIO	Robust Visual Inertial Odometry	34
ROVIOLI	Robust Visual Inertial Odometry with Localization Integration	34
MAPLAB	An open Framework for Research in Visual-inertial Mapping and Localization	34
OKVIS	Open Keyframe-based Visual-Inertial SLAM	35
ORB-SLAM	Oriented FAST and Rotated BRIEF Simultaneous Localization And Mapping	35
VI-ORB-SLAM	Visual Inertial Oriented FAST and Rotated BRIEF Simultaneous Localization And Mapping	35
ORB-SLAM3	Oriented FAST and Rotated BRIEF Simultaneous Localization And Mapping 3	x
MAP	Maximum a Posteriori	35
VI-DSO	Direct Sparse Visual-Inertial Odometry	39
VINS-Mono	A Robust and Versatile Monocular Visual-Inertial State Estimator	39
ROS	Robot Operating System	42
HEC	Hand Eye Calibration	43
RTS	Robotic Total Station	48
ENU	East-North-Up	45
deg	degree	47

1. Introduction

Over the last two decades, MMSs have become a rising trend in mapping applications due to their capabilities of providing fast, accurate, efficient, and complete data collection [63]. Any MMS consists of several sensors, for example, laser scanners, metric and panoramic cameras, Global Navigation Satellite System (GNSS), Inertial Navigation System (INS), and Distance Measurement Indicator (DMI) [5]. These systems measure 2D or 3D geometric information utilizing their sensors attached to a vehicle, for instance, a car, a train, an aircraft, or a boat, to generate a geo-referenced point cloud of all objects along their trajectory [51].

GNSS is the core element of any traditional vehicle navigation system that can provide centimeter-level accuracy. However, due to multipath effects and GNSS outages (over tunnels, large structures, and vegetation), it is practically difficult to sustain the GNSS signal during a complete survey. In order to address this problem, an Inertial Measurement Unit (IMU) is fused with GNSS data. An IMU consists of three-axis accelerometers and gyroscopes that provide self-contained data about the vehicle's instant position, velocity, and attitude. It should be notified that, because of its dead reckoning (DR) working principle (new positions are calculated using the earlier ones), the IMU errors are grown over time, causing drift. These errors are due to the noise and biases existing in the inertial sensor measurements [51]. For this reason, in most MMSs, GNSS data is used to provide periodic corrections to IMU outputs. Furthermore, a DMI is also fused with the GNSS/IMU as an additional aiding device in most of these systems. DMI is a cost-effective and robust sensor to measure rotational data. Its measured data can be further processed into position and velocity to be integrated with INS/GNSS output [51].

Recently, MMS with INS/GNSS/DMI and mapping sensors have been extensively developed. In most cases, a Kalman filter is used in these systems to combine INS/GNSS/DMI measurements to produce optimum navigation information. Nevertheless, the navigation outputs in the current MMS still have errors, mainly over GNSS challenging areas where GNSS data is not available (due to GNSS outages) or not accurate enough (due to multipath errors) to be fused with IMU data. Therefore, another aiding system is required to provide a robust, precise, and stable navigation solution, especially in long-term GNSS-blocked environments [51]. In order to achieve this goal, one possible solution is the image-assisted Simultaneous Localization and Mapping (SLAM) system to be used as an additional aiding algorithm in combination with the INS/GNSS tool, and thus improving the MMS trajectory estimation.

Simultaneous Localization and Mapping (SLAM) has become a well-known study area as a promising approach, recently solving most of the issues linked to the autonomous robot applications [31]. A fully autonomous robot should have the capability of exploring the surrounding environment, building a complete map, and locating itself on the map without any user's support needed. Mainly if GNSS data is unavailable or does not have adequate accuracy, the robot must determine on its own, which reference points have to be used to create a proper map and simultaneously estimate its trajectory [29].

There are two main types of SLAM systems: visual SLAM and non-visual SLAM. Visual SLAM systems use camera data as input, while non-visual SLAM systems rely on sensors such as

LIDAR, Sound Navigation and Ranging (SONAR), or Radio Detection and Ranging (RADAR). Recently, there has been a growing interest in developing algorithms for visual SLAM due to the abundance of visual information that can be extracted from cameras, as well as their simplicity and cost-effectiveness when compared to LIDAR sensors. However, one of the major limitations of visual SLAM is its dependence on lighting conditions. This means that sufficient light is required to capture high-quality images that can be used as input for the visual SLAM algorithm.

Visual Simultaneous Localization And Mapping (V-SLAM) comprises three principal components. The first part is the Visual Odometry (Visual Odometry (VO)), which is the process of estimating the robot's trajectory by just utilizing its on-board camera observations [60]. The second component is recognizing an already visited environment, known as the loop closing phase. The third step is applying optimization techniques to compute a global, uniform, and complete map. It is worth stating that, in contrast to VO, which aims to estimate the robot path incrementally and only cares about its local consistency, V-SLAM's key objective is to acquire a global, consistent estimation of the trajectory. In order to do this, V-SLAM needs to create and sustain the environment map even for the cases where localization is the only goal [60].

Most of the V-SLAM algorithms are very sensitive to motion blur, occlusions and lighting changes. In addition, for the monocular (single camera) V-SLAM case, the obtained result is up to the scale. In contrast, the inertial sensors can provide the motion's absolute scale and are robust against the aggressive movement, whereas their output data is noisy, which diverges very quickly. Therefore, it is possible to significantly boost both the algorithm's robustness and accuracy by fusing the inertial sensor measurements to V-SLAM in a tightly coupled manner [53]. When an IMU is integrated with the visual sensor under the concept of SLAM/Odometery, the algorithms are usually known as Visual Inertial SLAM (Visual Inertial Simultaneous Localization And Mapping (VI-SLAM)) and Visual Inertial Odometry (Visual Inertial Odometry (VIO)).

Interestingly, different research have been done on integrating the INS/GNSS/SLAM for various applications over the last few years. However, up to the author's knowledge, none of them examined the V-SLAM/VI-SLAM system's applicability for the case of a railway MMS attached on a train traveling at medium speeds in a dynamic environment, such as station areas or urban canyons. Moreover, in contrast to typical ground robots, the railway MMS has one distinctive characteristic: its motion is constrained to the rail paths. This specific constraint implies some benefits and drawbacks to the quality of motion estimation, which need to be analyzed precisely. Therefore, this research thesis investigates how the V-SLAM/VI-SLAM algorithms can be used as part of the RILA system to enhance its trajectory in GNSS-challenging conditions, with the main focus on train station environments.

The rest of this chapter is structured as follows. First, we present the concepts of SLAM. Furthermore, we introduce an overview to the RILA setup. Then we declare the problem and our research questions and, finally, we address the structure for the rest of this dissertation.

1.1. Simultaneous Localization and Mapping (SLAM) Concept

Over the last decade, SLAM has become a significant study area as an assuring method to solve most of the issues linked to the autonomous robot applications [31]. A fully autonomous robot should have the ability to explore its surrounding environment without any assistance, construct an accurate map and locate itself on it. Mainly in the case of GNSS unavailability the robot must decide, on its own, what are the proper reference points on which to create a map and estimate its trajectory at the same time [29].

In the following subsections, initially, the concepts of localization and mapping are explained independently. Next, the SLAM problem is defined, and its mathematical representation is addressed.

1.1.1. Localization

With the purpose of making autonomous robots move correctly in unknown environments, having prior information of their surrounding is necessary. In the localization step, the robot's position is estimated based on the known location of the landmarks around it. The robot passes through the environment and identifies different landmarks using its sensor data. After that, the robot can determine its location relative to these landmarks. Figure 1.1 shows an illustration of the robot's localization step within a known map. As it can be seen here, the robot can initially recognize the landmarks precisely. However, as it moves further along its path, it wrongly decides to turn left instead of turning right, possibly due to various environmental factors. At this stage, the robot uses its sensor observations to discover the landmarks within the map and correct its location accordingly. Moreover, it is crucial to notice that, during this step, the correctness of the localization relies directly on the accuracy of sensor data and the motion performance quality [72].

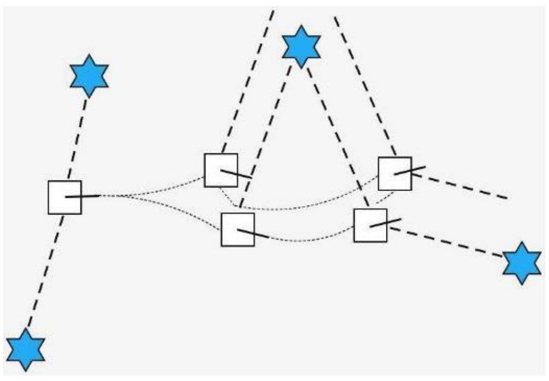


Figure 1.1.: An example of robot's localization when the environment map is given [72].

1.1.2. Mapping

Besides the localization step, the other part of the SLAM approach is the mapping phase. In this phase, the robot's position is known, and the landmarks' location needs to be computed using the robot sensors. When a robot moves through the environment, its on-board sensors collect observations from the surrounding landmarks. Using these observations, the robot can construct a map that depicts the location of each landmark. As illustrated in Figure 1.2, the estimated location of the landmarks is not definite due to the presence of noise in the sensor data. These uncertainties are shown clearly by the ellipse around the landmarks. In order to determine the precise position of the landmarks within a specific bound, these uncertainties need to be estimated beforehand using the Gaussian distribution, as mentioned in [72].

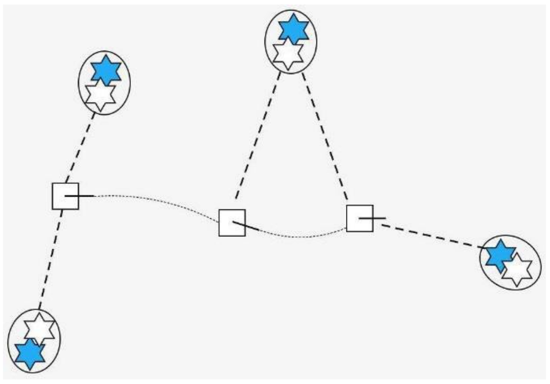


Figure 1.2.: Illustration of robot's mapping when its trajectory is known [72].

1.1.3. Localization and Mapping Simultaneously

After describing the localization and mapping phases, the SLAM process is defined briefly in this section. According to [7], the SLAM method's primary aim is to create a model of the environment (the map) while simultaneously calculating the robot's location within this map. In the SLAM process, the accuracy of the localization and the mapping construction are directly dependent. This means that, in order to precisely determine the location of the robot, a high-quality map is necessary. At the same time, an exact localization process is essential to create a precise map [72].

To explain the SLAM problem's mathematical basis, firstly, different variables involved in this approach need to be specified. Figure 1.3 exhibits the SLAM system's graphical model that represents the robot's trajectory and the sequence of sensor measurements. Additionally, in this model, the arrows are used to indicate the causal relations between these variables [66]. Here, for the remaining part of this section, we will mostly refer to the discussions, notations, and equations which are used by [66] unless otherwise stated.

To start with, the path of the robot can be described as follows:

$$X_T = \{x_0, x_1, x_2, ...x_t\} \tag{1.1}$$

Here, x_0 is the robot's known initial position and x_t denotes its location at time t. X_T is the sequence of the robot's locations, which shows its trajectory over the time interval T.

Secondly, the sequence of control commands or odometry information illustrated by Eq. (1.2) is used to characterize the robot's relative movement. The odometry measurement

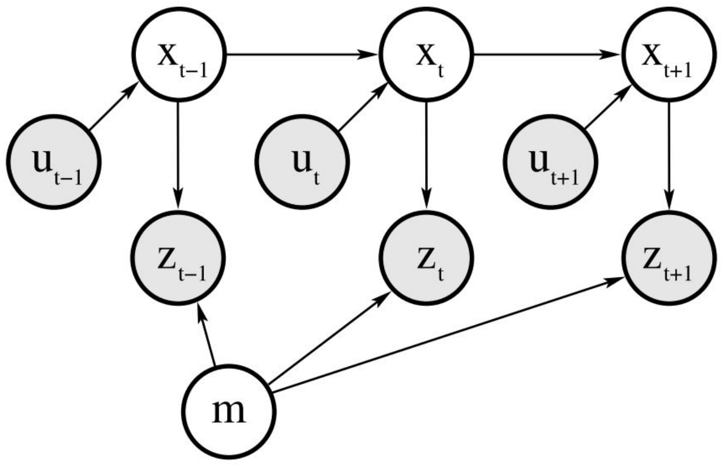


Figure 1.3.: Graphical illustration of the SLAM technique. In this model, shaded nodes are observable parameters to the robot, and non-shaded nodes are the variables that the robot tries to estimate. Arrows that connect these nodes are casual relationships. u_t is the odometry data which relates the robot's poses at time $t-1$ and t. In addition, z_t is the robot's sensor observation which can be used to relate the features in the environment map and the robot position at epoch t [66].

denoted by u_t, may be taken from wheel encoders or the controls command sent to the robot's motor, which describes the movement between epoch $t-1$ and t. In a perfect world, the entire trajectory X_T can be retrieved from the initial position x_0 using odometry data for noise-free movement. Nonetheless, these data are imperfect, and the trajectory integration eventually deviates from the truth.

$$U_T = \{u_1, u_2, u_3, ...u_t\} \tag{1.2}$$

Thirdly, the robot needs a series of observations at several points to establish information between the robot position x_t and the features in the world map m. Depending on the robot's sensors, its observation may be in the form of camera images [69] or LIDAR data [70] from a laser scanner, as mentioned in [72]. The sequence of observations can be defined as shown in Eq. (1.3), presuming that at each point in time, the robot performs only one observation.

$$Z_T = \{z_1, z_2, z_3, ...z_t\} \tag{1.3}$$

The SLAM problem is now the process of determining the trajectory of the robot and its surrounding environment map concurrently using the odometry and sensors measurements. From a probabilistic viewpoint, the SLAM problem may be classified into two different types. First, as [71] explained, the online SLAM requires determining the posterior over the current robot's position and the environment map. The mathematical representation of the online SLAM [66] is shown below:

$$p\left(x_t, m \;\middle|\; Z_T, U_T\right) \tag{1.4}$$

Where x_t shows the robot position at time t, m is the map of the environment, Z_T indicates the observations, and U_T is the control input. It is worth mentioning that the online SLAM problems are often solved using filter-based algorithms, which are incremental, meaning that the earlier observations and control inputs are discarded once processed [71].

The second classification of the SLAM problem is the full (offline) SLAM. The objective of the full SLAM is to calculate the joint posterior probability over the robot's entire path X_T along with the map m from the observed data Z_T and control input U_T. The batch algorithms are usually addressed to solve the offline SLAM problems [66, 71].

$$p\left(X_T, m \;\middle|\; Z_T, U_T\right) \tag{1.5}$$

Furthermore, two other models that correspond to the arrows in Figure 1.3 need to be described for solving the SLAM problem. First, the motion model that relates two sequential locations of the robot using odometry data can be determined using Eq. (1.6). [72].

$$p\left(x_t \;\middle|\; x_{t-1}, u_t\right) \tag{1.6}$$

After that, the observation model showing the probability distribution of the robot observation while the robot's location x_t, and the environment map m is known, may be defined by [72] :

$$p\left(z_t \;\middle|\; x_t, m\right) \tag{1.7}$$

1.2. RILA Setup

This study focuses on one of Fugro's mobile mapping systems, RILA, which has been adapted for rail inspections. As demonstrated in Figure 1.4, the train-mounted system integrates a variety of technologies such as GNSS, IMU, LIDAR scanner, video cameras, and laser vision to measure railway track geometry. This method allows for conducting track surveys at high speeds of up to 200 km/h (125 mph) without requiring track possession [79]. RILA can be easily attached to any passenger train to gather precise engineering data on the rail corridor. This eliminates the need for surveyors to work on live railways, making it a much safer alternative.

The data collected by RILA can be used to derive a wide range of valuable information related to rail track geometry and safety parameters. This includes parameters such as track gauge and cant, as well as identifying rail wear and the condition of switches and crossings. The data can also be utilized to determine ride comfort parameters, which is important for ensuring a smooth and safe ride for passengers. Furthermore, the collected data can contribute to Building Information Models (BIMs), which is a valuable resource for designers, engineers, and other stakeholders involved in the rail industry. Overall, the data provided by RILA is highly versatile and can be used for a wide range of applications, contributing to the overall safety and efficiency of rail transportation systems.

Railway track geometry measurement can be classified into two types: absolute and relative. Absolute track geometry measures the track position and is important during the design, construction, and as-built stages. Surveyors typically perform this measurement using various techniques such as GNSS, total station surveys, static and mobile laser scanning. The measured parameters include radius and gradients, which are critical for the track layout. In contrast, relative track geometry measures the track quality, and hence it is essential for planning maintenance and repairs. It includes parameters such as track gauge, longitudinal level, alignment, cant (cross level), and twist [79].

In RILA system , the collected absolute track geometry data meets the stringent accuracy requirements necessary for track design and is already in use in the Netherlands and the United Kingdom [79]. Moreover, the computed relative track geometry parameters satisfy the accuracy standards set by the European standard EN 13848-1. The methodology used to obtain these measurements is validated through comparison with results obtained using conventional approaches, such as track recording vehicles.

1.2.1. Existing Approaches and Constraints

This section discusses the importance of accurate trajectory estimation for the RILA system and the limitations of existing methods. Generating reliable point cloud data requires precise trajectory estimation. Any errors or drifts during the estimation process can cause the resulting data to have incorrect offsets and rotations. This can lead to discrepancies when comparing multiple surveys of the same area, affecting the quality and accuracy of the data. Therefore, accurate trajectory estimation is essential for maintaining consistency and repeatability in point cloud data, ultimately enhancing the reliability and usefulness of the generated data for various applications.

The RILA system integrates INS and GNSS sensors, providing reliable trajectory estimation under normal conditions. As reported in [79], the system's standard deviation in the horizontal and vertical directions are less than 8 mm and 12 mm, respectively, whereas the geo-referenced point cloud accuracy is typically around 10 mm and 15 mm, respectively. However, the accuracy of the trajectory estimation may decrease when GNSS environmental conditions become challenging. For instance, when a train travels through a partially covered station, the accuracy of GNSS data is impacted. This is because of less-than-optimal Position Dilution Of Precision (PDOP) values resulting from satellite geometry, and significant multi-path conditions caused by nearby objects close to the GNSS antenna. These factors significantly impact the accuracy of the GNSS data, reducing its reliability for trajectory estimation. Therefore, there is a need to improve the accuracy of the trajectory estimation to ensure the consistency and reliability of the generated point cloud data, particularly in challenging environmental conditions.

Currently, to address the challenges in trajectory estimation, the RILA system employs a weighted average method. The method involves conducting multiple runs over the same track and averaging the post-processing results to increase the reliability and accuracy of the final trajectory estimation. The automated weighting process reduces the impact of runs that differ greatly from the mean. However, this method is effective only if each survey possesses an acceptable absolute accuracy. If not, the averaged values may lack accuracy, resulting in discrepancies from the actual true trajectory values. Therefore, integrating SLAM into RILA system can be a valuable solution to improve the absolute accuracy of each survey, especially in challenging areas. In other words, SLAM can enhance trajectory accuracy without requiring additional field measurements. By improving the accuracy of individual surveys, the overall accuracy of the weighted average method can be significantly enhanced.

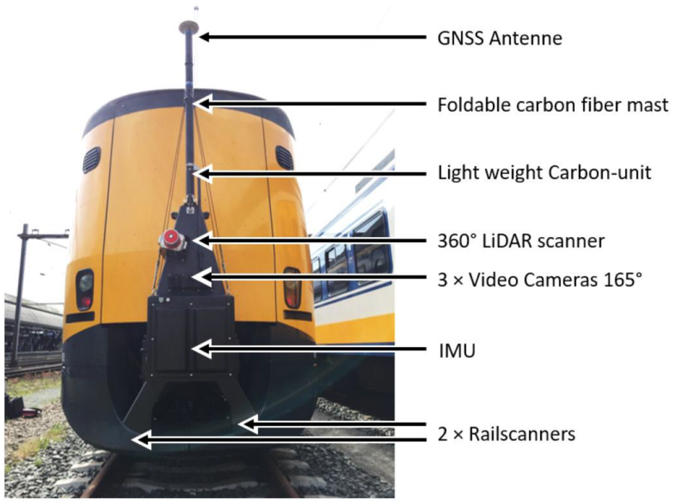


Figure 1.4.: RILA System Overview

1.3. Problem Statement

Having identified the potential advantages of using V-SLAM/VI-SLAM algorithms to improve RILA trajectory estimation, the first research question is formulated as follow:

RQ1. *How Monocular Visual (Inertial) SLAM methods can be implemented on the RILA system to enhance the positional accuracy of trajectory measurements over GNSS challenging sites, such as train stations?*

To answer this research question, Firstly, a literature study is conducted to compare the available Visual (Inertial) SLAM methods and subsequently, select the suitable candidate according to the key requirements of railway applications. Secondly, the trajectory is generated using the selected VI-SLAM method. To this end, the selected method requires modification based on our case study to be examined using the collected data from RILA system.

Multiple aiding information can be extracted from the generated SLAM-based trajectory, including Zero Velocity Update (ZUPT) which can be used to improve our INS/GNSS integration result. Hence, the first sub-research question is formulated as follow:

SUB-RQ1. *In which way, the Zero Velocity Update (ZUPT) information extracted from the SLAM solution can be used during post-processing to limit RILA's IMU drifts?*

Having developed the SLAM-based solution for RILA system, it needs to be validated using a corresponding ground-truth trajectory. Therefore, the second research question is defined as follow:

RQ2. *How the generated Visual Inertial SLAM solution can be validated using INS-GNSS data collected by RILA system?*

To answer this research question, it is assumed that the output trajectory of the RILA in open sky condition where the satellite coverage is adequate can be considered as our ground truth. Furthermore, a problematic trajectory is required to be generated by blocking the GNSS data for a certain period of time. Finally, the SLAM-based trajectory over the problematic area is compared with the ground truth.

In the next step, towards our research goal, the identified SLAM-based solution in the first research question 1.3 is required to be evaluated using different criteria. To this end, the SLAM-based method is mainly examined in the railway station area, where the condition is repetitive, low texture and dynamic, to investigate the maximum achievable accuracy.

SUB-RQ2. *What is the achievable accuracy of the proposed SLAM approach in dynamic, low texture, and repetitive corridor-like mapping environments such as Dutch railway stations?*

1.4. Thesis Contribution

This work contributes to the SLAM research domain by (i) providing an extensive overview of existing techniques, (ii) highlighting the current limitations and challenges of the techniques for the railway application, (iii) developing a pipeline to adapt the selected SLAM technique based on RILA system requirement, and (iv) evaluating the proposed technique using a real-world industrial use case. In summary, the main contribution of this dissertation can be categorized into the following two subjects:

Contribution 1. *Proposing a SLAM-based technique to improve the trajectory measurements in GNSS challenging areas for the railway application*

The following three steps are taken place to achieve the first contribution:

1. Selecting a SLAM technique to improve the positional accuracy of the estimated trajectories in GNSS challenging areas for the railway application.
2. Proposing a comprehensive pipeline to implement the chosen SLAM method for RILA system.
3. Fine-tuning the result of the execution of the pipeline to enhance the precision and the repeatability.

Contribution 2. *Evaluating the performance of the proposed SLAM-based technique by defining relevant Key Performance Indicator (KPI) such as accuracy and error values*

To achieve the second contribution, the following three steps are considered:

1. Analysing the performance using experimental approaches based on the relevant metrics.
2. Exploring the achievable accuracy of the proposed technique considering the railway environment limitations and challenges.
3. Investigating the impact of the aiding information, extracted from the SLAM result, on correcting the estimated trajectory.

1.5. Thesis Outline

The remainder of this thesis, as illustrated in Figure 1.5, is structured as follows. Chapter 2 introduces basics of SLAM and V-SLAM and additionally covers the carried out related work. Chapter 3 seeks to find an answer for the first research question (RQ1) of this study by implementing a VI-SLAM method for RILA system. Subsequently, Chapter 4 answers the second research question (RQ2) as well as the first sub-question by investigating the impact of ZUPT on trajectory improvement. Furthermore, it addresses the second sub-question of this study by analysing the selected SLAM-based method, using a set of performance metrics and establishing experiment. Finally, Chapter 5 concludes the thesis and suggests some research directions for future work.

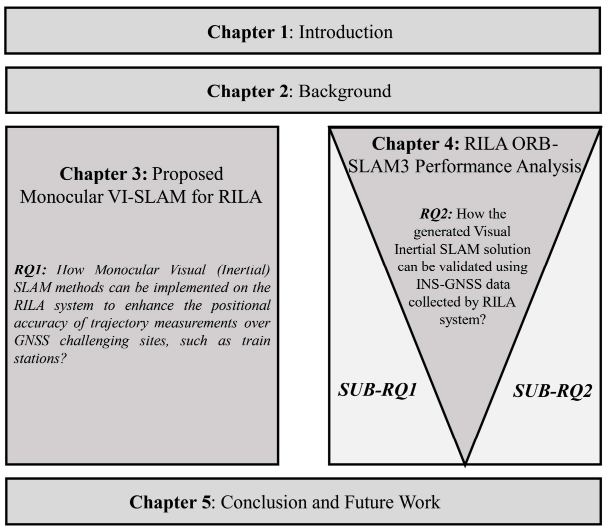

Chapter 1: Introduction
Chapter 2: Background
Chapter 3: Proposed Monocular VI-SLAM for RILA
RQ1: How Monocular Visual (Inertial) SLAM methods can be implemented on the RILA system to enhance the positional accuracy of trajectory measurements over GNSS challenging sites, such as train stations?
Chapter 4: RILA ORB-SLAM3 Performance Analysis
RQ2: How the generated Visual Inertial SLAM solution can be validated using INS-GNSS data collected by RILA system?
SUB-RQ1
SUB-RQ2
Chapter 5: Conclusion and Future Work

Figure 1.5.: A proposed dissertation outline

2. Background

This chapter aims to give a brief overview of SLAM by defining the terminologies and the concepts relevant to the work. The relevant concepts are further elaborated to gain the required understanding of the existing SLAM methods and to select the proper method considering the limitations and challenges of the railway application. Section 2.1 broadly outlines the three main methods for solving the SLAM problem. Section 2.2 presents the V-SLAM method and introduces the VO and its different configurations. In section 2.3, the different techniques as well as the processing methods for INS-GNSS integration are stated. Finally, Section 2.4 reviews the existing works in the area of VI-SLAM and highlights the suitable method for our study.

2.1. SLAM Paradigms

The SLAM problem has been a significant research focus in robotics for the last few decades. Various techniques have been proposed to solve the SLAM problem, and these techniques can be broadly classified into three main paradigms: Extended Kalman Filter (EKF)-based, Particle Filter-based, and Graph-based SLAM [72].

The EKF-based SLAM approach is the oldest and the most well-known method for SLAM. It is an extension of the Kalman Filter and is computationally efficient. The EKF-based SLAM uses a linearized approximation of the nonlinear observation and state models to estimate the robot's pose and the map. The Particle Filter-based SLAM method is a non-parametric approach that uses a set of particles to represent the robot's state and the map. The Particle Filter is a popular online SLAM method that can handle non-linear observation and state models.

In recent years, the Graph-based SLAM has gained popularity as a promising solution to the SLAM problem. Graph-based SLAM methods represent the environment and the robot's trajectory as a graph, where the nodes of the graph correspond to the robot's poses and the map features, and the edges correspond to the constraints between them. The graph-based SLAM approach uses a nonlinear optimization technique to solve the graph's constraints, which results in an optimal estimate of the robot's trajectory and the map. In the next section, we will focus on the Graph-based SLAM paradigm and its different optimization techniques used for solving the SLAM problem [66, 72].

2.1.1. Graph-based SLAM

The graph-based approach uses the nonlinear sparse optimization technique to solve the SLAM problem. The main idea of graph-based technique can be explained as follows [66]:

- The robot poses and landmarks locations are considered as some nodes in a graph.

- Each consecutive pair of positions x_t, x_{t-1} is connected by an arc which indicates the observation obtained from the odometry measurement u_t at time t.
- The robot pose x_t, is connected by some other arcs to all sensed landmarks m_i at time epoch t.
- These steps will be done consecutively for the entire trajectory to construct the graph.
- In this imaginary graph, all arcs can be thought of some soft constraints. The best estimate of the robot's trajectory and the environment map can be determined then by relaxing these constraints.

To clarify the technique mentioned above, the construction of such a graph can be explained using the following figures. Initially, imagine the robot observes the landmark m_1 at the first epoch. At this point, an arc needs to be added in the graph to connect x_1 and m_1. To illustrate these constraints in a matrix format, this step causes a value to be inserted in the elements between x_1 and m_1 as shown in Figure 2.1.

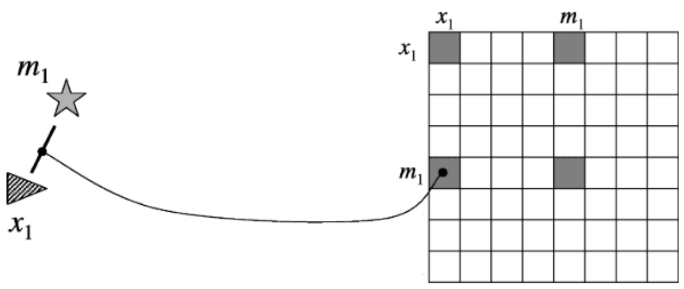


Figure 2.1.: Observing the first landmark m_1[66]

Now, let us assume the robot moves from its initial pose x_1 to x_2. This movement can be represented by an arc between poses x_1 and x_2 using the odometry measurement u_2. Figure 2.2 indicates this motion between x_1 and x_2 and its corresponding element in the matrix representation.

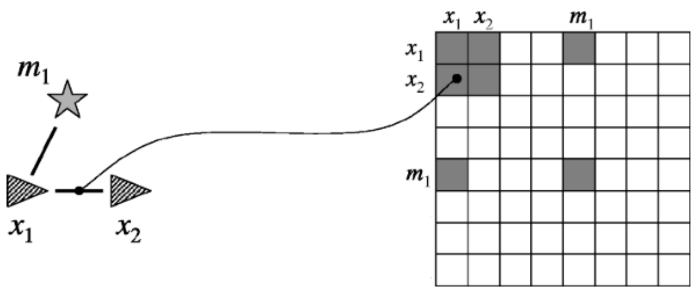


Figure 2.2.: Moving from x_1 to x_2 [66]

Similarly, as indicted in Figure 2.3, these two basic steps can be done successively to construct the graph.

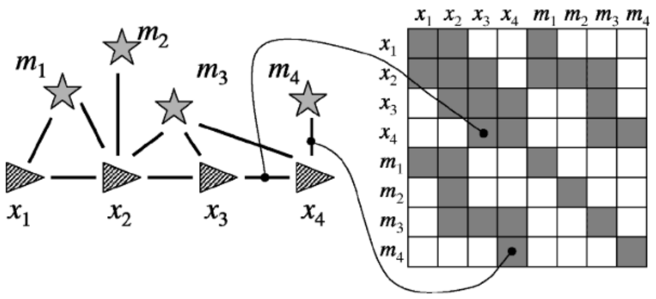


Figure 2.3.: Construction the graph consecutively, several step [66]

As it can be seen, the constructed graph is sparse, which means every specific node is just linked to a few numbers of other nodes. Now, let us think of this graph as a kind of spring-mass model. In this case, solving the SLAM problem can be considered as determining the minimum energy of this model. To show this, let us now represent the corresponding graph by the log-posterior of the SLAM problem as follows [66]:

$$\log p(X_T, m \mid Z_T, U_T) = const + \sum_t \log p(x_t \mid x_{t-1}, u_t) + \sum_t \log p(z_t \mid x_t, m) \tag{2.1}$$

It is worth noting that in the graph, two sorts of constraints exist. The first type is the constraint from each single motion event $\log p(x_t \mid x_{t-1}, u_t)$, while the second type is driven from a sensor observation $\log p(z_t \mid x_t, m)$. An arc represents each of these constraints in the graph. Finally, the SLAM problem is solved by determining the mode of the following equation [66]:

$$X_t^*, m^* = \underset{X^T, m}{\operatorname{argmin}} \log p(X_T, m \mid Z_T, U_T) \tag{2.2}$$

As shown in Eq. (2.2), the objective is to find the robot trajectory X_t^* and the map of the environment m^* with which the full SLAM argument $\log p(X_T, m \mid Z_T, U_T)$ is maximized. Several optimization techniques are available that can be used to solve this problem as presented in [66].

As mentioned earlier, scalability is one of the main difficulties for EKF SLAM, which makes it not an efficient method for large-scale mapping. The chief limiting factor of EKF is its continuously (and quadratically) growing covariance matrix. On the contrary, the graph-based techniques do not have such limitations. The necessary memory is linear, and the update time of the graph is fixed in this method. Moreover, most graph-based methods are naturally offline, which means they are used to optimize the robot's entire trajectory using the collected observations [66]. Since this research aims to improve Rila system's full trajectory in an offline manner (by batch processing of the observed data), graph-based techniques can be a suitable SLAM paradigm to be used considering the characteristics mentioned above.

2.2. Image-based SLAM

In recent years, there has been an increasing interest in exploring algorithms for image-assisted SLAM, also called Visual SLAM (V-SLAM), due to the intense visual information extracted from the camera, its simplicity, and cost-efficiency [84]. As [60] stated, the V-SLAM process's principal aim is computing a global, steady estimation of the robot trajectory. In general, the V-SLAM approach can be illustrated by the following expression [19]:

$$VSLAM = VO + loop\ closure + global\ map\ optimization \tag{2.3}$$

As it is presented in Eq. (2.3), V-SLAM comprises three main components. The first phase, VO, is the process of estimating the robot's trajectory utilizing its on-board camera observations [60]. The second phase consists in recognizing the already visited environment, known as loop closure. Finally, the third phase is about applying optimization techniques to compute a global, consistent map.

In this section, the V-SLAM approach will be discussed briefly. First, the camera modeling and calibration are illustrated in section 2.2.1. Next, section 2.2.2 describes the various camera configurations for VO approaches. Following that, section 2.2.3 explains the VO pipeline. Subsequently, the distinctions between the VO and V-SLAM methods are clarified in section 2.2.4. Finally, the Visual Inertial SLAM (VI-SLAM) approach is introduced briefly in section 2.2.5.

2.2.1. Camera Modeling and Calibration

Before describing the fundamentals of V-SLAM approaches, three different frames that are used in this study have to be specified:

1. The vision frame (v), is a local frame fixed concerning Earth and is also known as the world frame.
2. The camera frame (c), which defines the position and orientation of the camera.
3. The image frame (I), is a frame describing the pixel coordinates on the image plane.

Furthermore, in vision techniques, mainly the *Pinhole Projection Model* is used to compute the 3D motion of the camera from the 2D images. This model is illustrated by Figure 2.4. Additionally, in this section, most of the notations, equations, and discussions are referred to [2] unless specified otherwise.

As illustrated by Figure 2.4, a 3D point in the vision frame is denoted by p^v and its projection in the image plane by p^I. These two points can be related using the following expression:

$$k_c p_h^I = \Pi p_h^v \tag{2.4}$$

In Eq. (2.4), a 3D world point p^v and its corresponding 2D image point p^I are given in the homogeneous coordinates by p_h^I and p_h^v to represent the projection as a linear transformation. In addition, K_c is a scale factor introduced due to the unknown depth parameter using a single camera. The projection matrix is denoted by Π, which depends on the camera's intrinsic

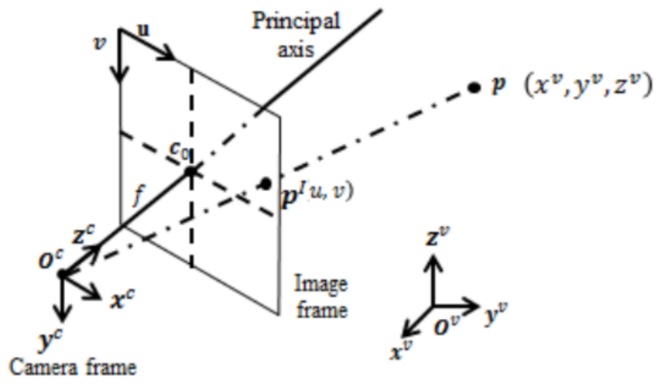


Figure 2.4.: Pinhole Camera Projection Model [2, P.36]

and extrinsic parameters. Furthermore, the principle point is indicated by $c_0 = (u_0, v_0)$ which gives the pixel coordinates of the image plane and the optical axis intersection.

Next, the calibration matrix that describes the transformation among the camera and the image frames can be defined:

$$K_c = \begin{pmatrix} K_u f & s_\gamma & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

where the focal length and the skew factor are denoted by f and s_γ, respectively. k_u and k_v indicate the number of pixels per unit distance in different directions, and (u_0, v_0) is the principle point coordinates in the image frame. As [60] mentioned, there are several open-source camera calibration toolboxes developed, which can be used to estimate the above-defined calibration matrix K_c.

Now, considering a position vector P_c^v and a rotation matrix C_{c2v} that indicates the rotation from the camera to vision frame, the relation between camera and vision coordinates for a physical 3D point may be formulated as:

$$p^v = C_{c2v} p^c + p_c^v \tag{2.6}$$

Knowing that C_{c2v} is an orthogonal matrix ($C_{c2v}^{-1} = C_{c2v}^T$), to project a 3D point from a vision into an image frame by using a pinhole camera model can be expressed as follows:

$$\Pi = K_c [C_{c2v}^T - C_{c2v}^T p_c^v] \tag{2.7}$$

2.2.2. Monocular, Stereo and RGB-D Setup

A VO system's primary purpose is to estimate an agent's trajectory on which a camera is rigidly mounted. Depending on the application's key requirements, different camera setups for VO implementation may be utilized. Monocular, Stereo, and RGB-D setups, which are the most popular VO approaches in the literature, are presented here. Moreover, it should be stated that, various V-SLAM algorithms are developed recently using different camera models such as monocular [14, 42], stereo [65, 43], RGB-D [25, 30], Omni-directional [32, 55] cameras.

Monocular

The monocular configuration is the most straightforward approach because just one camera is necessary to perform VO. In this method, the 3D scene is projected on the image plane, causing the loss of depth data. This means the obtained results with monocular implementation have the scale ambiguity. However, it is possible to determine the poses and the 3D environment's model by combining several frames from a single camera [19]. In the monocular approach, a minimum of three consecutive images is required to estimate the motion. In the first frame, some features need to be detected. Next, those features are re-observed, matched, and 3D points are triangulated in the second frame. Finally, the motion of the camera can be determined in the third frame. Moreover, the absolute scale can be obtained by fusing another aiding sensor, such as an IMU, LIDAR sensor, or a predefined item in the scene[19]. Figure 2.5 shows an illustration of a monocular VO system.

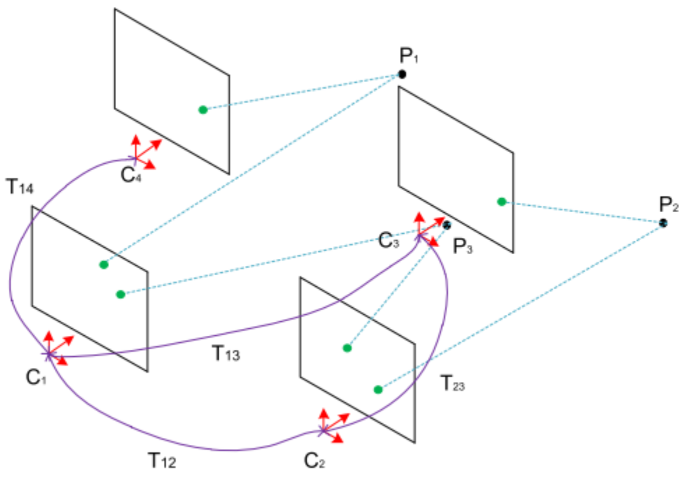


Figure 2.5.: An illustration of monocular VO system. P indicates the physical 3D points in vision frame, C gives the camera poses and T denotes the relative transformation between consecutive epochs. Furthermore, the purple line shows the trajectory of the camera. [84]
.

Stereo

Second, the stereo systems comprise two cameras. These cameras are separated by a fixed span identified as the baseline. In this setup, the depth information may be determined by triangulation at each epoch, using the same observed features by the left and right cameras. Thus, only two sequential frames are needed to evaluate the motion with stereo systems. In the first frame, the depth is computed by triangulation, and then the movement is calculated in the second one [19]. An illustration of a simple stereo geometry is given by Figure 2.6. Here, f indicates the focal length, Z is the depth. In addition, C_1 and C_2 are the center of lenses, x_1 and x_2 are the image coordinates in the left and right cameras, respectively [62, P.112].

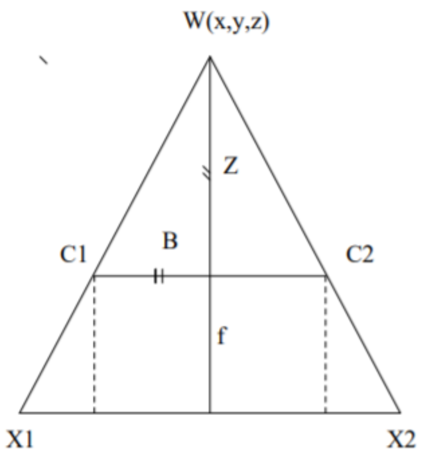


Figure 2.6.: An illustration of a simple stereo geometry [62, P.112]

The following expressions can be then derived, considering the equivalency condition between the triangles $\triangle W x_1 x_2$ and $\triangle W c_1 c_2$:

$$\frac{Z+f}{Z} = \frac{x_1 + x_2 + B}{B} \tag{2.8}$$

Rearranging Eq. (2.8) gives the depth Z :

$$Z = \frac{fB}{x_1 + x_2} \tag{2.9}$$

In Eq. (2.9), the disparity is denoted by $(x_1 + x_2)$. If the baseline distance B between two cameras and their focal length f are known for a given stereo model, the depth information Z can be calculated. It should be remarked that the depth and disparity are inversely proportional. This shows an object closer to the camera has a larger disparity compared to a farther one [62].

Several studies have been conducted on the basis of a stereo-based method. This method is prevalent and widely used in various applications since it can provide an absolute scale without the need for an additional aiding sensor to be integrated. Nevertheless, the stereo systems' performance directly depends on the baseline distance between the two cameras. This means, if the distance to the observed landmarks is much greater than the baseline, the triangulation between the two cameras cannot be performed precisely, and the stereo setup demotes to the monocular one. This can be considered as one of the main limitations of stereo VO configuration [60, 19].

RGB-D

RGB-D cameras are the sensors that collect RGB images together with their corresponding depth details. As [25] stated, RGB-D cameras rely on either active stereo or time-of-flight sensing to determine the depth. Namely, the Kinect, developed by Microsoft, is a well-known device that uses this principle to generate 3D point cloud data. Moreover, these cameras are capable of producing 2D and 3D information simultaneously, making them an appropriate sensor to be used to solve a robot's motion estimation problem. In recent years, RGB-D VO approaches have been meticulously investigated, which provide high-accurate assuring results. Additionally, these methods can produce RGB and depth images concurrently. Consequently, in contrast to monocular VO, they do not suffer from scale ambiguity. Nevertheless, one of the RGB-D VO methods' main concern is their reliability that prevents them from being used widely in fully autonomous robot applications. They may fail in various challenging environments since the sensor data is processed differently by these methods [18].

2.2.3. Visual Odometry (VO)

According to [60], VO is the method of calculating a robot's motion only using its on-board camera observations. The phrase, VO, is selected due to its relation to wheel odometry working principle, which, by integrating the number of its wheels turns, gradually calculates a vehicle's movement. Similarly, VO works by incrementally estimating the vehicle's position by following the changes generated by motion in its on-board cameras' images. It is worth noting that a semi-static well-textured scene with adequate lighting is essential for VO methods to work efficiently. Furthermore, in order to estimate the motion correctly, the subsequent images should have sufficient scene overlap.

VO typically consists of three main stages to calculate the robot trajectory based on a series of images captured by its on-board camera. The first stage defines matches among two consecutive images and during the second stage, outliers are eliminated. Finally, in the last step, the motion between the two captured images is estimated [2]. The illustration of these three stages is given by Figure 2.7, which is explained as follow:

Establishing Matches

In this stage, two consecutive images are compared by matching their identical characteristics. Whether the features detection and tracking are needed, this step may be categorized as either feature-based or direct approach [33].

Feature-based methods The feature-based methods, also known as indirect methods, detect and track a limited set of critical features between successive frames. In this way, by determining and minimizing sparse feature points' re-projection errors, the relative motion among two images is estimated. In other words, the main concept behind feature-based methods is to break the problem into two consecutive stages; firstly, from the image, a collection of features is detected. Secondly, the camera poses and scene geometry can be computed by using these feature observations [17].

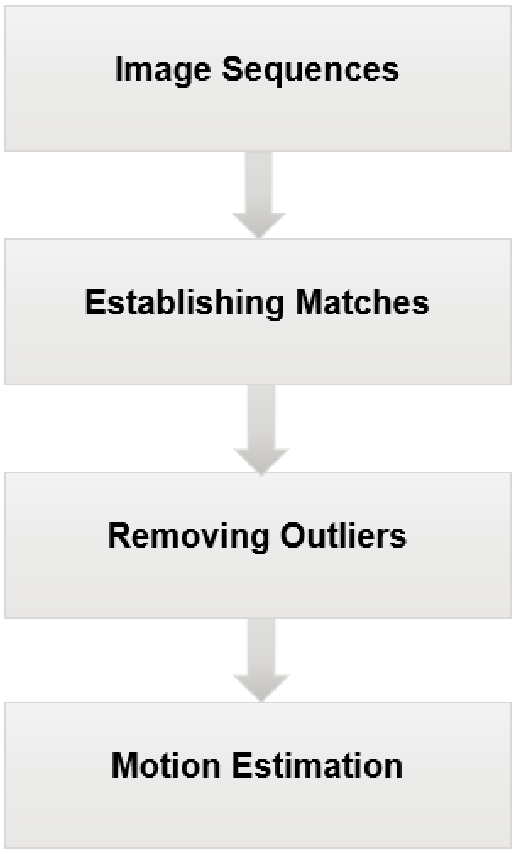


Figure 2.7.: A Block diagram showing the Visual Odometry main steps.

In this method, the feature detectors are used to search and select salient features in the image, like corners, edges, and blobs. In fact, depending on the requirements, a trade-off between the feature quality and the required computational resources needs to be made in order to choose suitable feature detectors [2]. For example, the Shi-Tomasi [64], Haris [21], and Feature from Accelerated Segment Test (FAST) [56] detectors are some of the most popular algorithms which are used broadly.

Consequently, after detecting the features in the image, they need to be uniquely distinguished using feature descriptors. Those descriptors that are invariant against various variations such as viewpoint, scale, illumination, and noise are known to be the most stable descriptors. There are several descriptor implementations where some of the most common are: the Scale Invariant Feature Transform (Scale Invariant Feature Transform (SIFT)) [37], Binary Robust Independent Elementary Features (BRIEF) [8] and (Oriented FAST and Rotated BRIEF (ORB)) [57]. Several implementations are based on different combinations of detectors and descriptors, which can have diverse performances. In other words, the variety of a detector-descriptor should be chosen such that they function together optimally considering the environment model and the application's requirements. [23] and [27] intensively compare and assess several varieties of feature descriptors and detectors [2, 19].

Finally, in this step, the features with similar characteristics need to be matched or tracked [2]. As [84] stated, the term feature tracking is used when the same features are followed among two adjacent images and is helpful when there are little changes in the motion. On the other hand, feature matching involves selecting features separately and matching them across several frames. It would be specifically beneficial if notable variations in the features' appearance occur due to tracking them over more extended sequences. An illustration of feature matching applying SIFT, Speeded Up Robust Features (SURF) and BRIEF descriptors is shown in Figure 2.8.

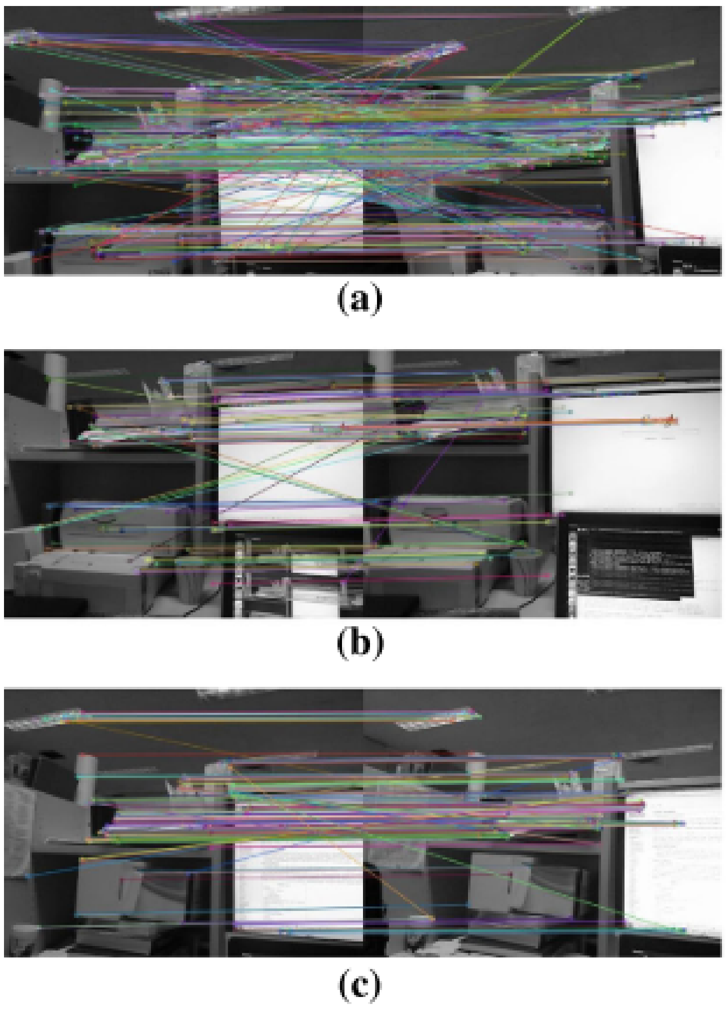


Figure 2.8.: An example of feature matching methods. (a) shows SIFT, (b) SURF and (c) BRIEF matches [84, P.306]

.

Direct methods: In contrast to feature-based methods, in the direct approaches, also known as appearance-based methods, the goal is to minimize a photo-metric error among corresponding pixels in consecutive images using only the sensor photo-metric observations [19]. Direct methods can be classified further into three different approaches as [83] mentioned—first, the dense methods [45] which utilize all image pixels information. Second, the semi-dense approaches [17] which use only pixels with sufficient gradient, and finally, in the sparse method [16], the sparsely selected pixels are considered. Furthermore, by minimizing the photo-metric error using non-linear optimization techniques, the pixel depths and camera positions can be computed [83].

It has to be pointed out that the optical flow technique is the principal theory behind the direct method. Within this technique, the intensities of neighboring pixels are used to determine the displacement of intensity patterns from one frame to another. In this way, by applying the optical flow technique, the relative motion between objects and a camera can be computed. In other words, the value of optical flow indicates the amount of pixel's intensity, which has moved in consecutive images [19]. Figure 2.9 shows the illustration of the optical flow matching technique between two consecutive images.

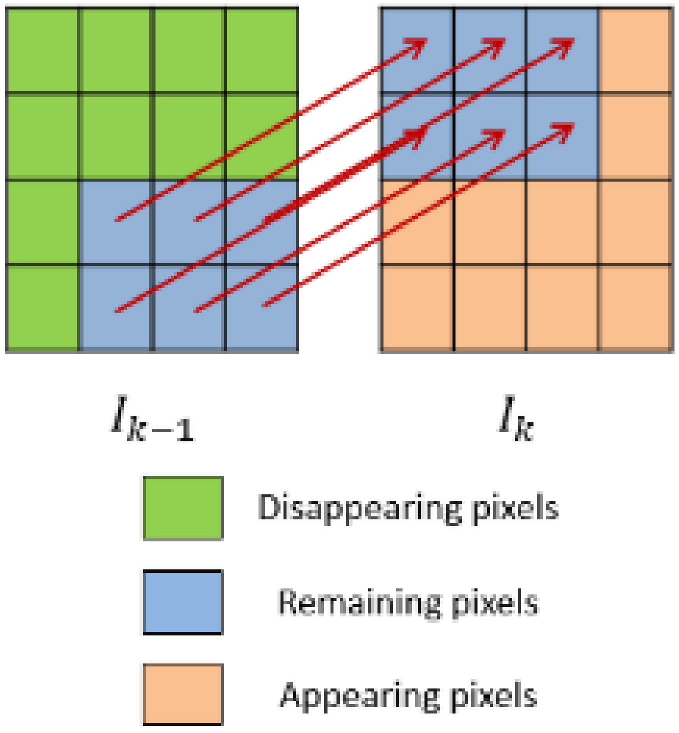


Figure 2.9.: Optical flow matching technique [2, P.39]

As [84] mentioned, the *Intensity Coherence* assumption is applied to determine the optical flow at all pixels. This assumption says that the intensity of a point projected in two consecutive images remains fixed (or roughly fixed). Considering this, the optical flow constraint equation may be derived:

$$\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} = 0 \tag{2.10}$$

In Eq. (2.10), the optical flow components are shown by v_x and v_y. Additionally, $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the image gradients along x, y and time respectively. [82] reviewed a number of various developed algorithms for solving the optical flow problem utilizing the described motion constraint equation [84].

In general, the feature-based VO is more robust for illumination and viewing changes between sequential images compared to the direct approaches due to their advanced feature descriptors. Nevertheless, detection and matching features require extra computational resources, which significantly restricts the number of retained features in the system. Therefore, the recreated 3D world models are relatively sparse and often can not be used directly in practical applications [83].

On the other hand, because it is possible to extract more information from the scene, the direct methods in a low-texture environment are more robust and can provide much denser 3D point clouds [83]. However, as [33] stated, because the direct methods utilize the entire image information, they are computationally quite expensive and much slower compared to the feature-based approaches. Furthermore, the direct methods are very susceptible to un-modeled artifacts such as camera auto exposure and rolling shutter effect because of the direct image adjustment formulation. More specifically, in practice, the principle of *Intensity Coherence* is not always valid, significantly decreasing the efficiency of the direct method when lighting conditions change quickly [83].

Outlier Removal

As can be seen in Figure 2.7, outlier removal is the second phase of the VO process. In this phase, incorrect matches, known as outliers, mostly because of the noisy sensors, changes in viewpoint and lighting, need to be excluded [84]. In order to address this, the Random Sampling Consensus (Random Sampling Consensus (RANSAC)) is frequently applied. An example of the RANSAC method is illustrated in Figure 2.10 and its step by step description can be found in [84]. Furthermore, the Epipolar geometry constraint can be used to implement the RANSAC algorithm for the outlier removal process as will be discussed further in this section [2].

Figure. 2.11 illustrates the geometrical Epipolar constraint. In this figure, p^I_{k-1} and p^I_k indicate the corresponding homogeneous image points seen from two consecutive locations of the camera at epochs $k-1$ and k. Therefore, as [22] addressed, c_{k-1}, c_k, p^I_{k-1}, p^I_k and the correspondence 3D point p^v need to be located in an identical plane known as the Epipolar plane. Considering this, the Epipolar constraint equation can be defined by [2]:

$${p^I_k}^T F^c_k \, p^I_{k-1} = 0 \tag{2.11}$$

In Eq. (2.11), the *Fundamental Matrix* at epoch k is denoted by F^c_k . This *Fundamental Matrix* represents the transformation of the camera between epochs $k-1$ and k, together with the

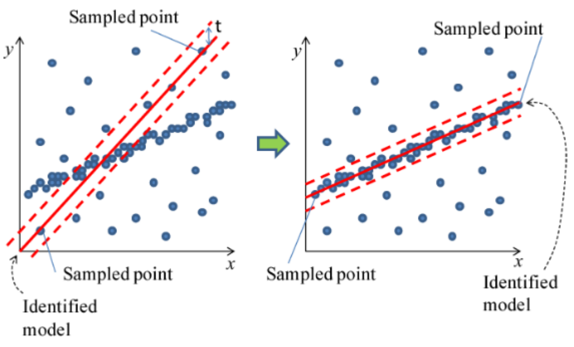


Figure 2.10.: An example of RANSAC algorithm. The outliers are indicated by the points which are located outside the dash lines [19].

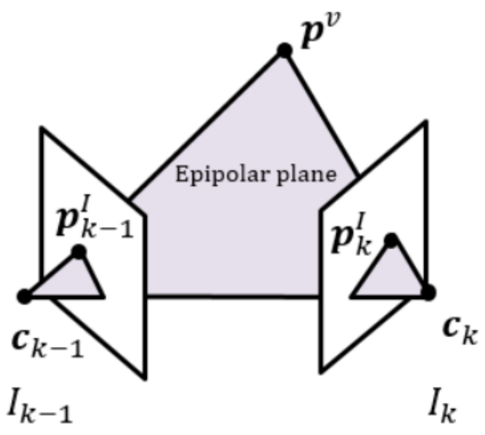


Figure 2.11.: An illustration of Epipolar geometry constraint. p^v and p^I deonte a physical 3D point in the vision frame and its projection in the image plane. $I_{(k-1)}$ and I_k are images at epochs $k-1$ and k respectively. c_{k-1} and c_k also show the camera locations at time $(k-1)$ and k [2, P.41]

internal camera calibration parameters. Besides, by each pair of homogeneous image points, a constraint can be added for the fundamental matrix calculation. According to [22], the normalized 8-point method may be implemented in combination with the RANSAC algorithm to determine F^c_k . To put it in another way, a minimum of 8 pairs of image points must be rightly matched to compute the fundamental matrix F^c_k. The other points that do not meet the geometrical Epipolar constraints defined by Eq. (2.11) can be counted as outliers and eliminated.

Similarly, as shown in [2, P.41], for the calibrated camera case, the geometrical Epipolar constraint can be rewritten as:

$$\overline{p}^{I\,T}_k E^c_k \overline{p}^I_{K-1} = 0 \tag{2.12}$$

Where,

- $\overline{p}_k^I = p_k^I (K_c^T)^{-1}$,
- $\overline{p}_{k-1}^I = K_c^{-1} p_{k-1}^I$,
- K_c is the known camera calibration matrix,
- And, E_k^c indicates the *Essential Matrix* which is a simplified version of the fundamental matrix for the case when the camera calibration is given.

Since the camera calibration matrix is known, in this case, at least, five pairs of correctly matched points are required to determine E_k^c. As [46] pointed out, the 5-point algorithm may be employed here for the essential matrix estimation. In other words, if a minimum of five correct matches is observed, E_k^c may be calculated directly. Furthermore, the system described by Eq. 2.12 is over-determined if more than five correct matches are obtainable. In this case, the least square minimization method can be applied to solve the problem, giving more robust results [2]. An illustration of 2D-to-2D motion is demonstrated by Fig 2.12.

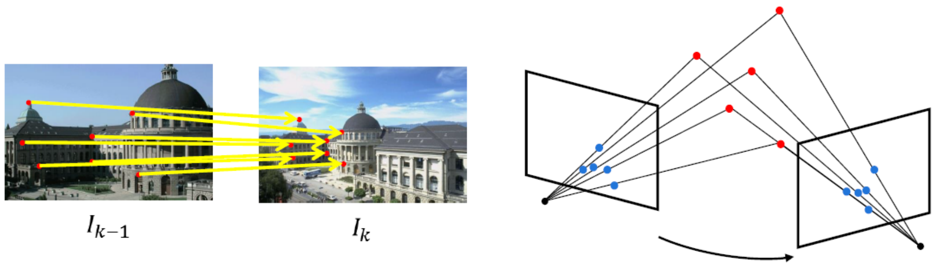


Figure 2.12.: Examples of 2D-to-2D motion [59].

Motion Estimation

According to [60], motion estimation is the main calculation phase conducted for each image in a VO approach. More specifically, after eliminating the outliers and estimating the essential matrix, the camera motion (2D to 2D) among every consecutive image may be computed using the earlier steps established matches. As mentioned before, for the calibrated camera case, the essential matrix may be determined by Eq. (2.12). Following that, as [22] described, this matrix may be decomposed into two parts; a rotation matrix and a displacement vector [2]:

$$E_k^c = \left(\left(p_{k-1,k}^{c_k} \right) \times \right) C_{k-1,k} \tag{2.13}$$

Where, the rotation matrix and the translation vector between epochs $k-1$ and k are denoted by $C_{k-1,k}$ and $p_{k-1,k}^{c_k}$ respectively. Moreover, $\left(\left(p_{k-1,k}^{c_k} \right) \times \right)$ expresses the skew-symmetric matrix of $p_{k-1,k}^{c_k}$.

Finally, as [60] stated, after concatenating all these single displacement vectors and rotation matrices, the trajectory of the camera may be estimated using Eq. (2.14) and Eq. (2.15) [2].

$$C_{c_k 2v} = C_{c_{k-1} 2v}\, C_{k-1,k}^{T} \tag{2.14}$$

$$p_{c_k}^{v} = p_{c_{k-1}}^{v} + C_{c_k 2v}\, p_{k-1,k}^{c_k} \tag{2.15}$$

Scale Estimation

As stated earlier, to determine an image sequence's trajectory, subsequent transformations need to be concatenated. In order to do this, the absolute scales are also necessary, which can be obtained using an extra aiding sensor such as an IMU. However, as [60] addressed, instead of computing the absolute scale, which is not possible from two images, the relative scales for successive transformations can be estimated using Eq. (2.16).

$$r = \frac{\left\| P_{k-1,i} - P_{k-1,j} \right\|}{\left\| P_{k,i} - P_{k,j} \right\|} \tag{2.16}$$

Where, $P_{k,i}$ and $P_{k,j}$ are two 3D points in the vision frame at time epoch k. Additionally, to achieve a more robust estimation, the scale ratios for several points are calculated, and the mean is considered alternatively as the relative scale. Following that, using the obtained distance ratio r, the translation vector can be scaled [19, 60].

Bundle Adjustment

Bundle Adjustment (Bundle Adjustment (BA)) is another important concept that should be explained in this section. As mentioned above, to reconstruct the trajectory during the VO process, successive transformations that each have their own uncertainties must be concatenated. As a result, the agent's pose uncertainty depends on the uncertainty introduced by all former transformations, as shown in Figure 2.13. To minimize these uncertainties, BA is a common approach that can be applied. In other words, BA is a nonlinear least-squares optimization used to optimize the camera poses and 3D features coordinates observed over various frames. The mathematical representation of minimizing the image re-projection error function in BA is shown below [60, 19]:

$$\underset{X^i, C_k}{\operatorname{argmin}} \sum_{i,k} \left\| p_k^i - g(X^i, C_k) \right\|^2 \tag{2.17}$$

Where,

- X^i indicates the ith 3D landmark.
- The ith 2D point of X^i measured in the kth image denoted by p_k^i.
- And, $g(X^i, C_k)$ shows the 3-D landmark X^i re-projection according to camera pose C_k.

BA is a critical process to optimize the obtained accuracy, mainly when the robot detects a loop closure (when returns to an earlier visited environment). This means if the robot's observed scene has features that have been seen before, the accumulated drifts of the trajectory can be eliminated by using the already known location of the 3D features [2].

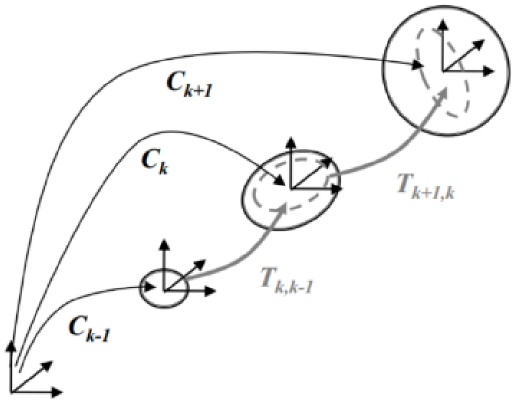


Figure 2.13.: The uncertainty in camera pose at epoch k directly related to its uncertatity at the previous epoch $k-1$ (shown as the black solid ellipse) plus the produced uncertainty by the transformation $T_{k,k-1}$ (indicated as the gray dashed ellipse) [19].

2.2.4. Visual Simultaneous Localization and Mapping (V-SLAM)

In contrast to VO, which aims to estimate the robot path incrementally and only cares about its local consistency, V-SLAM's principal objective is to acquire a global, consistent estimation of the robot trajectory. To do this, V-SLAM needs to build and maintain the environment map even for the case where localization is the only objective. In fact, keeping track of a map is necessary to detect loop closure. It has to be notified that loop closure is done by correlating the observed features and the robot positions, which can significantly reduce the drift in both the camera trajectory and environment map. To put it another way, by observing the features that have already been observed, V-SLAM integrates this constraint into the map to improve the camera path accuracy [60].

To reconstruct the 3D map from 2D features within the V-SLAM approach, the triangulation process is needed. As shown in Figure 2.14, within the triangulation process, the 3D location of a feature $\hat{p}^v$ needs to be computed by crossing back-projected rays from the corresponding 2D image points at two consecutive image frames ($\tilde{p}^I_{k-1}$ and $\tilde{p}^I_k$) [2]. According to [60], in absolute conditions, these back-projected rays should intersect each other in a single point $\hat{p}^v$; though, they almost never crossed due to calibration errors, camera model, the processing errors, and the noise in the image. As a result, from all intersecting rays, the point at a minimal distance in the least square senses can be chosen as a solution.

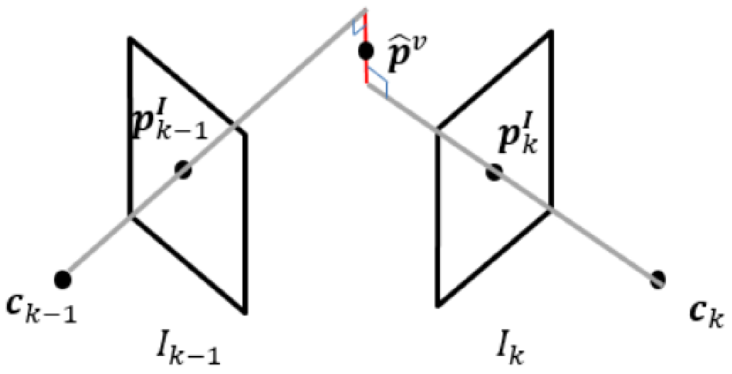


Figure 2.14.: Illustration of the triangulation process within V-SLAM [2]

Keyframe-based

Generally, V-SLAM methods can be classified into two distinct implementations. The first implementation is the filtering-based V-SLAM, which is centered on creating a probabilistic 3D feature map that represents the estimation of the camera's current position and the reconstructed 3D feature map in the vision frame. This method uses Bayesian filters, for example the Kalman filter or the Particle Filter, to estimate the state variables (the camera pose and 3D features) as the camera moves [2].

The second implementation is the non-filtering methods, also known as keyframe based methods. In this approach, the optimization of global BA is maintained for the selected keyframes. In other words, reconstructing the 3D map by choosing some selective frames is the central concept behind the keyframe based method. These frames need to be selected so that using the minimum number of correspondences between two images, the 3D map can be reconstructed. In this way, the reconstruction is just performed if there is a significant shift in the scene. The main advantage of keyframe based V-SLAM is that its computational expense decreases significantly because only a small number of frames are processed during the map reconstruction stage. The second advantage of this method is that it assures an adequate baseline for triangulating among the two selected images. To make this more precise, in the case of straight movement, when there is no distinguished variation in the observed scene, the time interval between two selected keyframes is longer. On the other hand, if the camera turns, the observed scene would change rapidly, and this would require a quick change of keyframes to consider the new features observed by the camera [60, 2].

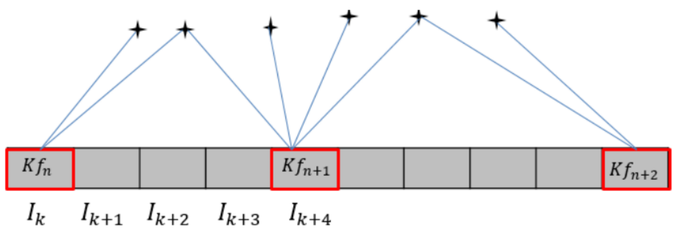


Figure 2.15.: An example of Keyframe selection process [2]

Figure 2.15 shows an example of the keyframe selection process. In Figure 2.15 , the n^{th} keyframe is shown by Kf_n. Additionally, I_k depicts the captured image at time epoch k. During this process, after selecting Kf_n as a keyframe, the number of corresponding features between Kf_n and I_{k+1} is determined. If the number of correspondences is less than a predefined threshold, I_{k+1} would be chosen as the next keyframe; otherwise, the next captured image needs to be examined. It is worth mentioning that, although only keyframes are used for 3D map reconstruction, the camera trajectory is estimated for all the frames given the location of the previously reconstructed 3D features. Because of this, BA, which is already explained in section 2.2.3 is usually implemented as a refinement process.

It has to be noted, the matching step in V-SLAM, in contrast to VO, where the matching is only established between 2D features, includes the 3D reconstructed features. This means that aside from 2D to 2D correspondences introduced earlier for VO, the V-SLAM algorithm utilizes the 3D to 3D and the 3D to 2D matching methods [2]. For 3D to 3D correspondences, the camera motion is determined by estimating the aligning transformation of the two sets of 3D features. This can be done by using Eq. (2.18) [60]:

$$\underset{T_k}{\text{argmin}} \sum_i \left\| \tilde{X}_k^i - T_k \tilde{X}_{k-1}^i \right\| \tag{2.18}$$

where T_k is the transformation, the i^{th} feature is shown by subscript i . The homogeneous coordinates of the 3D features are given by $\tilde{X}_k^i$ and $\tilde{X}_{k-1}^i$.

On the other hand, for 3D to 2D motion estimation, the transformation T_k needs to be determined using the 3D correspondence X_{k-1} and the 2D feature p_k. To estimate the transformation T_k that minimizes the image reprojection the following equation can be used [60]:

$$\underset{T_k}{\text{argmin}} \sum_i \left\| p_k^i - \hat{p}_{k-1}^i \right\|^2 \tag{2.19}$$

In Eq. (2.19), T_k is the transformation, the reprojection of the 3D feature X_{k-1}^i into the image I_k using the transformation T_k is denoted by $\hat{p}_{k-1}^i$. This problem is known as *perspective from n points*(PnP) and several solutions are available in the literature which can be used to solve it [60].

In general, as addressed by [46], the 3D to 2D motion estimation that minimizes the image reprojection error has higher accuracy compared to 3D to 3D methods, which is based on minimizing the feature position error. This makes it the most popular approach for the motion estimation process [60].

2.2.5. Visual Inertial Simultaneous Localization and Mapping (VI-SLAM)

As described earlier, in visually distinguishable areas, V-SLAM can produce an accurate trajectory concurrently with the environment's 3D map. However, most of the V-SLAM algorithms are very sensitive to occlusions, motion blur, and illumination changes. Additionally, for the monocular V-SLAM case, the obtained result is up to the scale. On the other hand, inertial sensors, such as IMU, can give high-frequency self-motion data. Inertial sensors are robust against the aggressive motion and can provide the motion's absolute scale, but their output data is noisy, diverging very quickly. Considering the characteristics of V-SLAM and inertial sensors, it is possible to significantly boost both the algorithm's robustness and accuracy by fusing the inertial sensor measurements to V-SLAM in a tightly coupled manner [53].

The benefit of integrating the visual and inertial sensors is most apparent in a monocular V-SLAM system, which can solve the scale ambiguity issue and may reduce the accumulated drifts over time. In other words, an inertial sensor, like IMU, may be used to determine the absolute scale for the monocular V-SLAM algorithm or even to improve the accuracy of the system by fusing the obtained data from both the camera and the IMU [19]. Since this research's main objective is to enhance the Rila trajectory accuracy in the GNSS challenging area using one of its attached cameras in a monocular setup, the VI-SLAM algorithm is chosen as the main pipeline for this study. Therefore, a literature study is conducted to higlights the most popular and suitable monocular VI-SLAM algorithms for railway applications in section 2.4.

2.3. INS-GNSS Integration Methods

There are various integration strategies used for INS-GNSS integration, each with its own advantages and limitations. One significant advantage of the integration of INS and GNSS is their complementary nature [3, 41]. According to [3], GNSS systems are more stable in the long-term, meaning their errors are effectively time-invariant with homogeneous accuracy. On the other hand, INS short-term errors are relatively small, but they degrade rapidly and are unbounded, making external aiding necessary. Therefore, GNSS systems can be used as external aiding to bound INS errors. In other words, INS sensors provide accurate short-term data with a high rate and can be used to interpolate GNSS trajectory. Additionally, INS sensors supply data with continuity, while GNSS is subject to outages caused by signal blocking or interference. High-precision inertial sensors such as tactical or superior grade can be used to bridge GNSS outages. Moreover, INS systems provide a complete navigation state, including position, velocity, and attitude, while a single GNSS receiver cannot supply angular information.

To integrate GNSS and INS systems, there are three commonly used integration strategies: loosely coupled integration, tightly coupled integration, and ultra-tight integration. However, as stated in [41], ultra-tight integration is implemented only by hardware manufacturers. Therefore, this section focuses on the other two techniques which are briefly explained in the next sub-sections.

2.3.1. Loosely Coupled Kalman Filter

The Loosely Coupled Kalman Filter (LCKF) is a common approach for integrating GNSS and INS data. In this approach, the GNSS and INS data are processed independently and then combined in a Kalman filter. The Kalman filter estimates the state of the system (position, velocity, attitude) based on the measurements from both the GNSS and INS systems. The LCKF has been widely studied and applied in many applications, including airborne, ground vehicle, and pedestrian navigation systems [81, 3, 41]. Figure 2.16 represents the integration scheme of this approach.

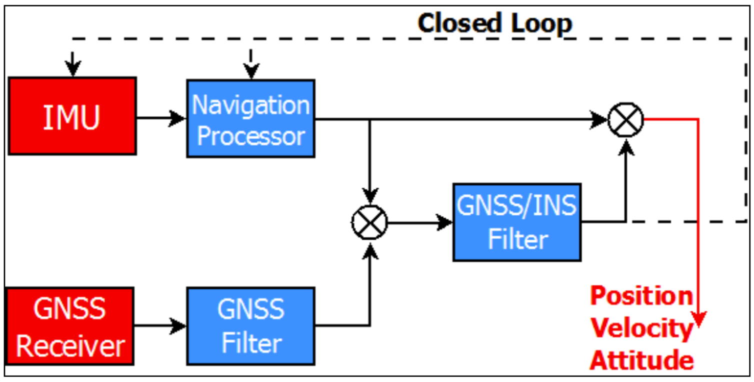


Figure 2.16.: The Loosely Coupled Integration Scheme [3]

One advantage of the LCKF is that it can handle different measurement update rates for GNSS and INS data, which can be useful for systems that require high update rates. Additionally, the LCKF allows for easy implementation and tuning of the filter parameters, making it a practical solution for real-time navigation systems. However, the LCKF has some limitations, including the need for accurate calibration of the sensor errors, and the fact that it does not fully exploit the complementary nature of GNSS and INS data. As a result, the accuracy of the LCKF can degrade over time, especially in challenging environments [81, 3, 41].

2.3.2. Tightly Coupled Kalman Filter

The Tightly Coupled Kalman Filter (TCKF) is another approach for integrating GNSS and INS data. In contrast to the loosely coupled method, the tightly coupled method directly incorporates the raw measurements from GNSS into the filter, rather than using the GNSS receiver's position solutions as input to the INS. Figure 2.17 shows the integration scheme of this approach. It can be seen that the raw measurements from GNSS are not processed in a separate filter but instead are directly integrated into a single filter. This allows for a more efficient and accurate integration, as the GNSS measurements are incorporated more directly into the filter, leading to better tracking of the system's state. The tightly coupled approach has been shown to provide higher accuracy than the loosely coupled method, particularly in challenging environments with high levels of multi-path and signal interference [49, 50, 3, 41].

One of the challenges with the tightly coupled approach is the increased computational complexity of the algorithm due to the direct use of the raw GNSS measurements. This can make the algorithm more difficult to implement in real-time applications with limited processing power. To overcome this, various techniques have been proposed, such as the use of adaptive filtering methods, which adjust the filter parameters based on the level of GNSS signal strength and quality [49]. Additionally, the use of efficient hardware and software architectures, such as Field Programmable Gate Arrays (FPGAs) and graphics processing units (GPUs), can help to accelerate the processing speed of the algorithm [50]. Overall, the tightly coupled Kalman Filter provides a powerful and accurate method for integrating GNSS and INS data, although it requires careful consideration of computational complexity and hardware resources in order to implement in real-time applications.

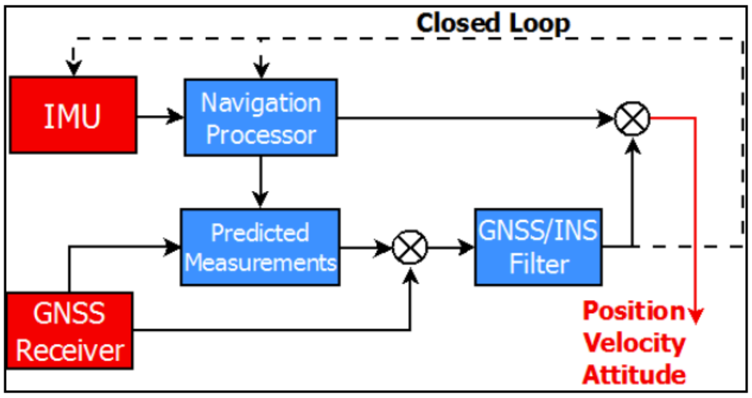


Figure 2.17.: The Tightly Coupled Integration Scheme [3]

2.3.3. Processing Methods and Directions

Different methods exist to process GNSS data, such as differential GNSS and Precise Point Positioning. In the RILA system, the latter technique is utilized. According to [20], differential GNSS is a technique that reduces the error in GPS-derived positions by using additional data from a reference GNSS station at a known position.

Furthermore, in RILA system during post-processing it is possible to run GNSS data both forwards and backwards in time, and independent forward and backward solutions are automatically combined to maximize accuracy. This combined method of processing achieves different solution types, such as fixed or float solutions, for different parts of the survey, depending on factors like baseline length, satellite geometry, and the number of satellites available. In the combined solution, inverse variance weighting is applied to ensure that the direction with the lower estimated errors receives the most weight in the combined trajectory.

2.4. Literature Review

VI-SLAM that integrates the visual and inertial data for localization and mapping has become more attractive for different applications. As mentioned earlier, to improve the RILA trajectory accuracy in the GNSS challenging areas, which is the main objective of this research, VI-SLAM is selected as the principal pipeline. Generally, VI-SLAM algorithms may be classified into two categories: filtering-based and optimization-based approaches. [12] is listed and shown the state-of-the-art VI-SLAM approaches, including the fusion types, camera setups, and their possible applications. In this section, some of the most popular state-of-the-art VI-SLAM methods are reviewed. Finally, some of the railway application related works under the SLAM framework are presented.

2.4.1. Filtering Based Methods

Considering the sensor fusion methods, filtering-based VI-SLAM algorithms may be categorized as loosely coupled and tightly coupled. In the loosely coupled approaches, the inertial sensor data and the observed images are processed separately before being integrated [13]. In contrast, in the tightly coupled algorithms, the camera's state is fused with IMU and inserted into the motion and observation equations to compute the full state. The tightly coupled method has more computational complexity; however, due to computer technology developments, the tightly coupled processes currently form the main research focus and are more popular [12].

One of the classical filtering based VI-SLAM system is the **Multi-State Constraint Kalman Filter (Multi State Constraint Kalman Filter (MSCKF))** [40]. In this method, a measurement model is derived to represent the geometric constraints when a static feature is detected from several camera positions. In MSCKF algorithms, the SIFT feature is extracted, and 30 camera poses are maintained in the filter state. In this method, the 3D feature position is not inserted in the state vector, causing the computational complexity to be linear in the number of features. Due to this, it is suitable for large-scale real-time mapping applications. In [8], MSCKF and the Sliding Window Filter (Sliding Window Filter (SWF)) are compared and it is proved that the latter is more precise and less sensitive to the tuning parameters [12, 13].

Robust Visual Inertial Odometry (ROVIO) is a monocular visual-inertial algorithm presented in [6]. In this approach, image patches' pixel intensity errors are directly utilized to obtain precise tracking performance together with a high level of robustness. In this model, the FAST corner is used for establishing feature matches. After matching, the multilevel patch features' tracking is linked to the EKF during the update phase, using intensity errors. It should be mentioned that within this algorithm, a pure robocentric method is used to estimate the location of 3D landmarks concerning the current position of the camera. In addition, there is no need for any initialization procedure in this framework because of the robocentric, inverse-distance landmark parameterization, making it an effective power-up-and-go state estimation system. In [6], the authors show how this algorithm can be applied for highly dynamic hand-held experiments and they evidence its efficient functionality in the control loop of a multi-rotor unmanned aerial vehicle.

In robotics, one of the principal challenges is achieving a robust and accurate state estimation. Through obtaining precise pose estimation using a prior map, the system applicability can be significantly improved [12]. In order to attain this goal and achieve an exact and drift-free motion estimation, [61] developed a robust and accurate visual-inertial system to localize against a prior map. This presented algorithm in [61], called **An open Framework for Research in Visual-inertial Mapping and Localization (MAPLAB)**, is an open-source visual-inertial pipeline to create, process and handle multi-session maps. This algorithm is a ready-to-use VI-SLAM system employed as a tool for visual-inertial batch optimization, loop closing, and map merging.

In addition, within the MAPLAB framework, a new tool called Robust Visual Inertial Odometry with Localization Integration (ROVIOLI) has been developed. It consists mainly of ROVIO with localization integration, applied as a front-end SLAM tool. Furthermore, MAPLAB is one of the first VI-SLAM frameworks that can be utilized for a broad diversity of applications in a single system. It is tested extensively for several cases, such as micro aerial vehicles, autonomous cars, walking robots, and autonomous underwater vehicles.

2.4.2. Optimization Based Methods

The optimization-based VI-SLAM approach has recently become more popular due to computer technology developments. Within this approach, the entire SLAM framework is splitted into two components; the front-end SLAM that is responsible for constructing the map and the back-end SLAM that optimizes the pose estimates. For the back-end SLAM section, different optimization tools are usually used, namely g2o [34] (an open-source framework for graph-based nonlinear error functions optimization) and ceres-solver [26] (which is a C++ library to model and solve complex optimization problems). Below, some of the most popular state-of-the-arts optimization-based VI-SLAM are briefly reviewed [12].

Open Keyframe-based Visual-Inertial SLAM (OKVIS) [35] is a keyframe optimization-based sliding window VI-SLAM system that integrates the IMU and landmark re-projection error terms into a cost function to be optimized. In this algorithm, the former keyframes are partly marginalized to keep a limited-sized optimization window in order to make a real-time operation possible. The authors showed that the global coherence of the gravity direction and robust outlier removal for the IMU motion model could be achieved by using their proposed method. Nonetheless, all the advantages of keyframe based nonlinear optimization are also achievable within their approach. Furthermore, by analyzing the acquired data from a stereo visual-inertial system, it is shown that their method can provide more accurate and robust real-time results compared to vision-only or loosely coupled approaches.

[44] proposed a novel algorithm, known as **Visual Inertial Oriented FAST and Rotated BRIEF Simultaneous Localization And Mapping (VI-ORB-SLAM)**. This algorithm integrates the monocular camera with the inertial sensor in a tightly coupled manner. This algorithm contains ORB sparse as a front-end SLAM component, whereas its back-end SLAM is based on graph optimization techniques. The proposed approach can also close the loop for trajectory estimation if the sensor revisits the same environment and hence, this allows achieving zero-drift localization in contrast to VO methods where drift grows unlimited. As mentioned earlier, the developed algorithm focuses on the monocular camera setup to recover the well-known scale ambiguity issue. A novel IMU initialization approach is employed in their proposed VI-SLAM system to estimate the scale, velocity, gravity direction, and the accelerometer and gyroscope biases in a few seconds accurately.

[44] showed that the VI-ORB-SLAM system is able to reach a standard scale factor error of 1% with centimeter precision on several micro-aerial vehicle public data-sets. Furthermore, by comparing their method with some of the state-of-the-art visual-inertial systems, it is proven that VI-ORB-SLAM has greater accuracy because of map reuse and no drift accumulation.

As [10] stated, **ORB-SLAM3** is the system which can perform V-SLAM, VI-SLAM, and multi-map-SLAM for several camera setups (monocular, stereo, and RGB-D) using different lens models (pinhole and fisheye). ORB-SLAM3 is a tightly-coupled feature-based VI-SLAM method entirely built upon *Maximum a Posteriori* (Maximum a Posteriori (MAP)) estimation, even within the IMU initialization stage. According to the authors, their developed system can be used robustly in real-time for small and large scale applications, achieving 2 to 5 times more accurate result compared to the previous cutting-edge methods.

It is worth mentioning that ORB-SLAM3 is built on Oriented FAST and Rotated BRIEF Simultaneous Localization And Mapping (ORB-SLAM) [42, 43] and VI-ORB-SLAM [44] which are the first visual and visual-inertial approaches that used the full profit of data association to reach drift-free localization in previously mapped environments. In comparison with those methods, one of the main novelties of ORB-SLAM3 is the possibility of multiple-map data association, making it also a suitable system for railway applications. Thus, ORB-SLAM3 is able to match and use the element coming from previous mapping runs in the bundle adjustment phase. In other words, ORB-SLAM3 has the capability to reach the actual aim of a SLAM system: creating a map of the environment to be saved and used later (not even necessarily from the same mapping season) for obtaining accurate localization.

2.4.3. Railway Application Related Works

[24] introduced a SLAM framework for path constrained motion. A novel SLAM strategy for moving platforms in case of tightly curve-constrained, i.e., 1-D motion. Their proposed filter is verified using real-world railway survey data to confirm that it is capable of acquiring the result with a position error of less than $1m$ and a precision of around $10cm$ for iterative mapping of railway tracks. In [80], visual track recognition is performed based on edge extraction with a known track gauge width. In their approach, only the direction that a train turns at switches is determined and not a continuous position estimate of the vehicle. The system is examined with real data from test runs with different weather conditions in various places and it is confirmed to be very robust for track-selective self-localization of railroad vehicles [75].

Moreover, [36] proposed a monocular vision-based localization algorithm for railway applications with different speed conditions. This approach derives keyframes and reference frames and uses only a few useful features to represent each keyframe. Their test outcomes confirm that their proposed method can reach 88.8% precision with 100% recall under an acceptable range of deviation from the ground truth, which outperforms SeqSLAM [39], a localization and mapping algorithm benchmark. In [4] a complete vision-based odometry system for rail vehicles is developed. This method converts a front-facing camera image into a birds-eye view of the track to determine the two successive frames' correspondences. The drawback of this framework is that in low-texture environments or repeated pattern conditions, which is frequently the case for railway environments, this approach might fail [75].

In addition, [75] reviewed the applicability, challenges, and constraints of the cutting-edge visual and visual-inertial odometry using different camera configurations for railway applications. Multiple data-sets recorded in industrial, urban, and forest environments are used and evaluated in their review. Based on their analyzed results, the stereo visual-inertial odometry outperforms other methods for rail vehicle's applications, as it can provide an accurate and robust motion estimation due to its complementing sensor modalities.

3. Proposed Monocular VI-SLAM for RILA System

This chapter aims at addressing the first research question of this study. This question enquires to define key requirements for railway application. These key requirements are further used to choose the most suitable SLAM method from the previously reviewed algorithms in chapter 2. Furthermore, a pipeline is presented to implement the selected method for RILA system to improve the positional accuracy of the estimated trajectory in area with poor GNSS coverage.

To this end, section 3.1 introduces the key requirement and the selected method. Section 3.2 discusses some preliminaries regarding the frame definition and the system calibration of the method. Section 3.3 presents the different phases of the pipeline that we used to implement the chosen method for RILA system. Finally, Section 3.4 implements the selected method based on the pipeline for RILA system.

3.1. Algorithm Selection

Various VI-SLAM algorithms are often developed for different applications. This makes it quite challenging to compare them with each other comprehensively. However, all of these systems have their own capabilities that can be considered to select the most appropriate method to achieve our research objectives. In this section, the critical requirements for improving RILA trajectory in GNSS challenging areas are discussed. Furthermore, the most suitable algorithm which can fulfill these essential requirements is selected for this research.

3.1.1. Railway application key requirements

As stated before, the RILA system consists of a GNSS receiver, INS sensors, video cameras, and laser scanners mounted on a train to perform real-time measurements related to absolute and relative track position and geometry with engineering level accuracy. For RILA trajectory estimation, currently, the GNSS/INS data is integrated with an external software package, which sometimes drifts remarkably in GNSS-denied areas. Therefore, the fusion of the SLAM-based aiding algorithm in RILA system is necessary in order to reach its desired accuracy. Regarding the selection of the most suitable image-assisted SLAM algorithm, first, the main challenges, limitations, and requirements for motion estimation in railway environments, particularly for RILA case, were described below.

In general, for railway applications, only a few restrictions exist regarding the weight, size, and required power of the localization system. This makes it possible to openly select suitable sensor modalities and estimation algorithms from the wide range of available V-SLAM solutions [75]. However, since RILA is a stand-alone MMS attached to a train, some additional limitations exist that need to be considered during the selection of the motion estimation approach:

- Due to weight restrictions, adding extra sensors in the RILA unit is not possible without following a complicated certifying process.
- Installing a DMI sensor for aiding INS/GNSS integration algorithm is not feasible due to the RILA installation procedure. RILA system is independent of its mounting platform and should have the possibility to be installed on any passenger train in less than three minutes. Figure 3.1 shows RILA sub-systems and the way it should be installed on any trains as fast as possible.
- The position and orientation of the three video-cameras placed on the RILA unit are fixed and can not be modified easily.

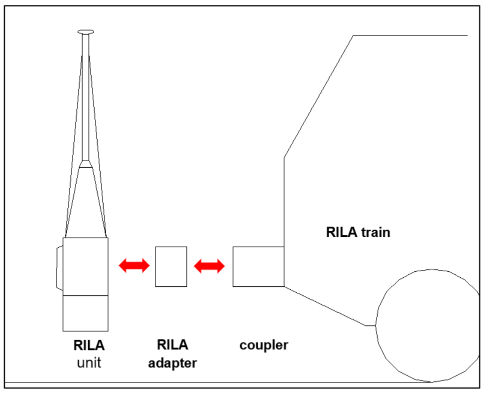


Figure 3.1.: RILA subsystems.

3.1.2. Selected method

In this section, considering the RILA application's key requirements, the most suitable method for this research was selected step by step:

Camera setup selection

Based on the literature, the stereo visual approach is more robust than the monocular configuration due to its faster initialization process. Also, in the stereo method, the real scale can be directly estimated, which is another significant advantage of this method compared to the monocular one [10]. However, in the RILA system, the orientation of the two adjacent cameras results in not enough overlap between them (around 15%). In addition, the current distance (baseline) between two adjacent cameras is much shorter than the standard stereo systems. Based on these two hardware-related limitations, it is expected that the stereo approach is not able to perform efficiently and robustly. Therefore, during this research study, the monocular visual SLAM method would be used.

Fusing inertial data to V-SLAM algorithm

As described earlier, for the monocular V-SLAM case, the obtained result is up to the scale. The inertial sensor, such as IMU that provides the motion's absolute scale, needs to be fused with the V-SLAM system to solve this scale ambiguity issue. In this way, it is possible to boost both the algorithm's robustness and accuracy significantly. Based on this, the monocular VI-SLAM algorithm is selected as the main framework for this study. According to [10], for applications where stereo camera setup is not attainable, the monocular-inertial can be employed without missing much in terms of robustness and accuracy.

Dynamic environments

In general, the feature-based VO is more robust to illumination, occlusions, scale, and viewing changes between consecutive images compared to the direct approaches, because of their advanced feature descriptors. In contrast, the direct methods, also known as appearance-based methods, utilize the pixels' intensities and exploit more visual information, make them more robust for motion estimation even in texture-less areas [19]. Nevertheless, considering this study's primary focus, which is improving the RILA trajectory in the station areas, the feature-based methods are more suitable for motion estimation than the appearance-based ones, as they are more invariant to dynamic situations.

Multi-Map Capability

One of the objectives of this research study is to investigate if the processing of the same track's multi-temporal data, collected by the RILA system during its four different runs, can improve the trajectory's positional accuracy. To be able to achieve this goal, during our final algorithm selection for this research project, a monocular VI-SLAM system with multi-map capability is chosen.

Accuracy and Robustness

RILA acquisition system aims to measure absolute and relative track position with engineering-level accuracy. To assist the RILA system in achieving this goal, particularly in GNSS challenging areas, it is necessary to choose an accurate and robust algorithm. In [10], Carlos Campos et al. compared the performance of some of the most common state-of-the-art V-SLAM/VI-SLAM using EuRoC data-set. Their evaluation shows that the ORB-SLAM3 is as robust as the best systems available in the literature. In addition, ORB-SLAM3 achieves more accurate results in all sensor configurations compared to the most accurate methods by a wide margin. Notably, in monocular-inertial configuration, ORB-SLAM3 can perform twice more accurately in comparison with Direct Sparse Visual-Inertial Odometry (VI-DSO) [77] and A Robust and Versatile Monocular Visual-Inertial State Estimator (VINS-Mono) [52], underlying the advantages of mid-term and long-term data association again.

Finally, after considering all the criteria mentioned above, ORB-SLAM3 was chosen as our monocular visual-inertial SLAM algorithm to be used during this research project.

3.2. ORB-SLAM3

To address our research objectives, we selected ORB-SLAM3. An overview on this method was previously presented in chapter 2. However, in this section we present some preliminaries regarding the different frame definitions, and system calibration methods of ORB-SLAM3.

3.2.1. ORB-SLAM3 Frames Definitions

As shown in Figure 3.2, the ORB-SLAM3 uses three different reference frames [11, 48] which are described as follow:

- World (W): W frame is a fixed reference system. In this frame, the z_W axis is pointing up in opposite of the gravity vector direction, i.e., g_W as illustrated in Figure 3.2. In ORB-SLAM3, the first camera position defines the the world reference frame.
- Body (B): B frame is an optimizable reference system. This frame is aligned with the IMU coordinate system. An important assumption here is that the IMU's gyroscopes and accelerometers have fully aligned reference system.
- Cameras (C_1 and C_2): C_1 and C_2 frames define the camera coordinate systems. In these frames, the z_C axis is pointing forward aligned with the optical axis of the camera. The other two axes, namely x_C and y_C are pointing to the right and down, aligned with the image vectors u and v, respectively.

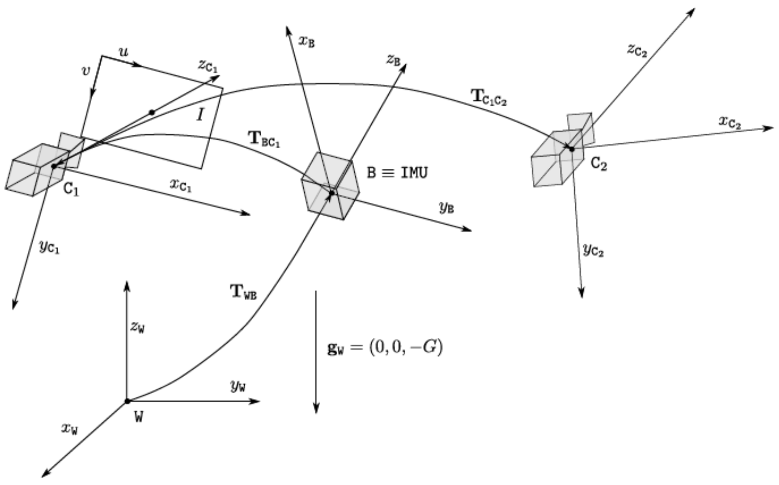


Figure 3.2.: ORB-SLAM3 reference frames as illusterated in [48]

3.2.2. ORB-SLAM3 System Calibration

System calibration is divided into two stages; extrinsic and intrinsic calibration [48]. During extrinsic calibration the geometrical relation between different sensors are defined using a set of parameters. Whereas, intrinsic calibration defines the set of parameters related to each sensor itself.

Extrinsic parameters are used to transform the C_1 frame to the B frame. Equation 3.1 formulates this relation for monocular inertial case:

$$T_{WC_1} = T_{WB}T_{BC_1} \tag{3.1}$$

where, T_{WC_1} and T_{WB} are the homogeneous matrices which transform $C_1$and B frames to W frame, respectively. In addition, T_{BC_1} denotes the matrix to transform points in C_1 frame to B frame.

Intrinsic parameters, as stated earlier, is defined for each sensor. For our case study, the intrinsic calibration for visual and inertial sensors are described separately.

Depending on camera set-up, different calibration parameters were provided. For Pinhole camera, the intrinsic parameters are the camera focal length (f_x,f_y) and central point (c_x, c_y) in pixel. Moreover, radial-tangential distortion model is the other intrinsic parameter defined as k_1, k_2, k_3. The last parameters are p_1, p_2 which address the tangential distortion coefficients.

Furthermore, *Gyroscope Noise Density*, *Accelerometer Noise Density*, *Gyroscope Random Walk*, and *Accelerometer Random Walk* defines the IMU intrinsic parameters. These parameters are available in the datasheet of the IMU sensors. The mathematical representation of these parameters can be found in [48].

3.3. RILA ORB-SLAM3 Pipeline

In this section, the adaption of ORB-SLAM3 method for RILA system is presented. To this end, a pipeline for RILA ORB-SLAM3 is depicted in Figure 3.3. This pipeline includes five main phases: **SLAM Tool Installation**, **Input Data Preparation**, **SLAM Tool Execution**, **Validation**, and **Trajectory Correction**. In the following subsections, each phase with its used techniques is described.

3.3.1. Phase (1): SLAM Tool Installation

In the initial phase, ORB-SLAM3 tool together with several C++ open source libraries were installed on a Linux-based platform. For this study, Ubuntu 18.04 was used as our build environment which was run on a powerful machine (Intel(R) Core(TM) i9-9980HK CPU @ 2.40GHz , 32 GB RAM) to ensure a real time performance with stable and accurate results. The dependencies for the ORB-SLAM3 are as follow:

- OpenCV: An open source library which provides real-time optimized computer vision tools and functions. In ORB-SLAM3 framework, this library is used mainly for image processing and feature extraction [47].
- Pangolin: A lightweight and portable platform which provides a set of libraries to prototype and visualize the video based data. ORB-SLAM3 uses Pangolin as a visualization and user interface [67].
- g2o: An open source framework to optimize nonlinear problems based on graph theory. Since ORB-SLAM3 is a graph-based SLAM, this framework is used for optimization of the nonlinear error functions [54].

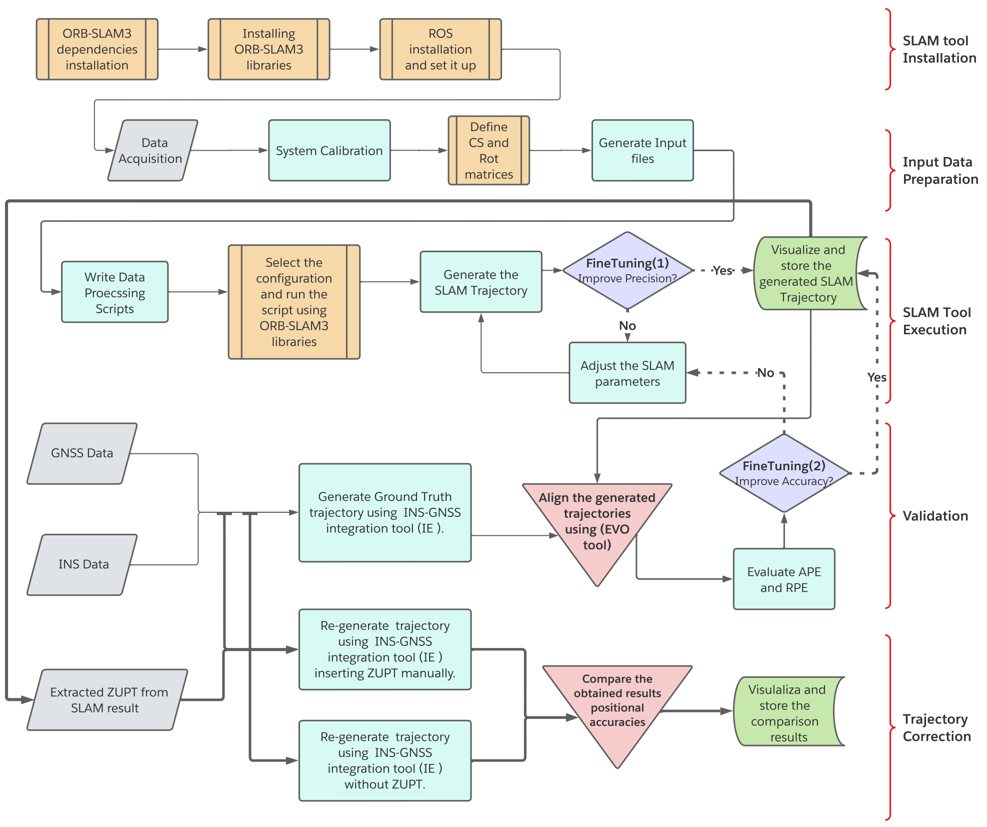

ORB-SLAM3 dependencies installation
Installing ORB-SLAM3 libraries
ROS installation and set it up
SLAM tool Installation
Data Acquisition
System Calibration
Define CS and Rot matrices
Generate Input files
Input Data Preparation
Write Data Proecssing Scripts
Select the configuration and run the script using ORB-SLAM3 libraries
Generate the SLAM Trajectory
FineTuning(1) Improve Precision?
Yes
Visualize and store the generated SLAM Trajectory
No
Adjust the SLAM parameters
No
Yes
SLAM Tool Execution
GNSS Data
INS Data
Generate Ground Truth trajectory using INS-GNSS integration tool (IE).
Align the generated trajectories using (EVO tool)
FineTuning(2) Improve Accuracy?
Evaluate APE and RPE
Validation
Extracted ZUPT from SLAM result
Re-generate trajectory using INS-GNSS integration tool (IE) inserting ZUPT manually.
Re-generate trajectory using INS-GNSS integration tool (IE) without ZUPT.
Compare the obtained results positional accuracies
Visualiza and store the comparison results
Trajectory Correction

Figure 3.3.: RILA ORB-SLAM3 Pipeline.

- Eigen3: A C++ library which is used to solve linear algebra related problems[38]. ORB-SLAM3 utilizes this library as part of g2o framework.
- DBoW2: An open source C++ library which provides an image database to enhance making queries and feature comparisons [15]. Place recognition in ORB-SLAM3 employs this library.

After setting up the build environment, the ORB-SLAM3 library was generated and installed on the machine. As the final step, Robot Operating System (ROS) was set up to further organize the build environment and make easier the access to different SLAM related tools and configuration.

To automate all these stages, a bash script was written by the author to fully install the ORB-SLAM3 which can be found in appendix A.

3.3.2. Phase (2): Input Data Preparation

The input data preparation phase should effectively calibrate the system, define the different coordinate systems and rotation matrices to enable transformation from one coordinate system to another according to the ORB-SLAM3 requirements. The outcome of this phase is a set of input files which are later used by the ORB-SLAM3 tool.

Figure 3.4 illustrates in more details the input data preparation stages. Initially, the camera and IMU sensor need to be calibrated using the collected calibration datasets.To this end, a Fugro custom-made calibration tool, Hand Eye Calibration (HEC), was used to calibrate the camera and estimate the intrinsic parameters such as focal length (f_x, f_y) and optical centers (c_x, c_y) as well as distortion coefficients $(k_1, k_2, k_3, p_1, p_2)$. Moreover, for IMU calibration, kalibr_allan tool was used to verify the provided accelerometer and gyroscope noise values from the datasheet. After verifying the intrinsic parameters of our sensors, the HEC tool was used for determining the extrinsic parameters and generating the corresponding rotation matrix.

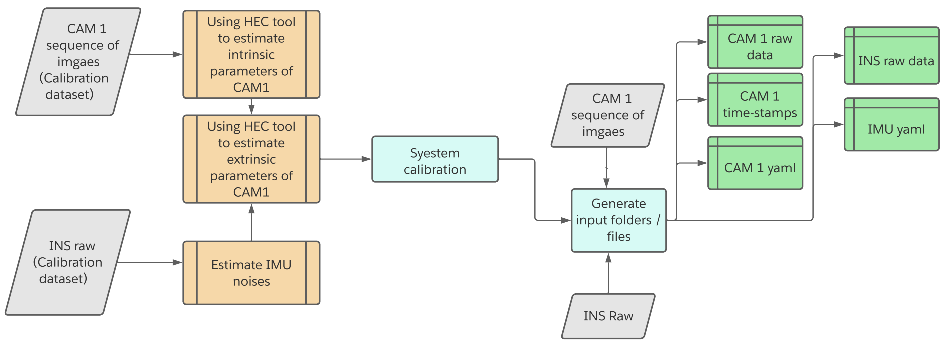

CAM 1 sequence of imgaes (Calibration dataset)
Using HEC tool to estimate intrinsic parameters of CAM1
Using HEC tool to estimate extrinsic parameters of CAM1
INS raw (Calibration dataset)
Estimate IMU noises
Syestem calibration
CAM 1 sequence of imgaes
Generate input folders / files
INS Raw
CAM 1 raw data
CAM 1 time-stamps
CAM 1 yaml
INS raw data
IMU yaml

Figure 3.4.: ORB-SLAM3 Input Data Preparation Phase.

Ultimately, the computed calibration parameters along with the sequence of images captured by the camera and the IMU sensor data were used to generate the set of input files for the ORB-SLAM3 tool. These input files are presented in green boxes in Figure 3.4.

3.3.3. Phase (3): SLAM Tool Execution

The SLAM tool execution phase is the heart of this pipeline. Within this phase, the aim is to produce a SLAM based trajectory using the previously generated input files. In this step, the proper configuration for our platform was chosen and subsequently the ORB-SLAM3 script was executed according to this configuration.

In the script, first, the path to vocabulary was defined which contains a vocabulary-tree of visual words, i.e., Bag of Words. This hierarchical visual vocabulary is commonly used for place recognition and loop closure. The vocabulary was built offline from a huge training collection of images and their features. For each feature in the image a visual word was assigned which has a weight according to the frequency of appearance in the training images. The higher the frequency, the lower the assigned weight is since the word is less discriminative. In this way, each image has a bag-of-word vector, i.e., a sparse numerical vector. To check the similarity between the images, their vectors are compared.

Secondly, the path to settings file was passed to the script. The setting file contains the camera intrinsic calibration parameters, the resolution, and the frame rate. It also includes the IMU intrinsic parameters and frequency as well as the rotation matrix from camera to

IMU frames and the color order of the images (such as BGR, RGB,...). Finally, the ORB binary descriptor parameters were specified in the settings file that can be optionally fine-tuned.

Eventually, after executing of the script, the SLAM based trajectory was generated which includes the position and orientation of the vehicle in each epochs.

Furthermore, the generated trajectory needs to be fine-tuned. This happened in two steps: Firstly, to ensure the precision and repeatability of our method the two key input parameters, (1) ORB-extractor, (2) IMU noise model, require optimization. The ORB-extractor is used to define the number of features per image which are used during the motion estimation. Moreover, the IMU noise model has an important effect on the behavior and the stability of our SLAM tool. If the estimated values are too optimistic, too much weight on IMU is placed which makes the system unstable.

Secondly, the accuracy of the SLAM trajectory needs to be improved. In general, all the SLAM tools ask for a certain time for initialization. During this initialization process, the vehicle needs to be stationary as much as possible and only has limited rotational movements. However, in our case study, the train enters the station area with medium to high speed. Therefore, there is not enough stationary time for the initialization of our SLAM tool. Hence, there would be some outlier data which would affect on the quality of the final trajectory. To reduce this effect, a fine-tunning method is used as explained in section 4.4.

3.3.4. Phase (4): Validation

Validation phase is required to evaluate the quality of the generated SLAM based trajectory. This phase is described in detail in section 4.

3.3.5. Phase (5): Trajectory Correction

Multiple aiding information can be extracted from the generated SLAM trajectory. In correction phase, the impact of these information on improving the accuracy of RILA trajectory in the GNSS challenging area is examined. Section 4.5 discusses this phase in detail.

3.4. Implementation of RILA ORB-SLAM3

This section describes how the ORB-SLAM3 can be implemented for RILA system. First, the RILA's reference frames are stated precisely. Next, it is addressed how RILA system is calibrated according to ORB-SLAM3 requirements and the generated calibration file is presented. After that, the selected RILA dataset for this study is used and the first result utilizing RILA ORB-SLAM3 is generated. Finally, after evaluating RILA ORB-SLAM3's result, the first phase of fine-tuning for precision and repeatability is implemented to make the system more stable.

3.4.1. RILA Frames Definitions

For RILA system we define three different frames, namely body, IMU and camera frames.

Body frame is defined to be aligned with the vehicle reference frame (when mounted on the rear-end of a train, 'facing' backward). This frame is represented using East-North-Up (ENU) convention:

- Origin is at the center of gravity of the IMU.
- Positive X axis is pointing toward right.
- Positive Y axis is pointing forward (toward the train).
- Positive Z axis is pointing up

Figure 3.5 shows the RILA's sensors and visualizes its body frame.

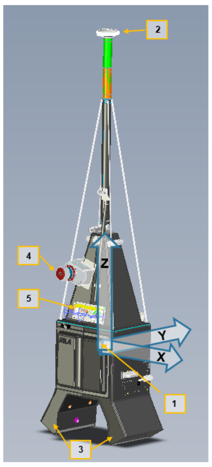


Figure 3.5.: RILA body frame: (1) IMU, (2) GNSS antenna, (3) Track scanners, (4) LIDAR scanners, (5) cameras

IMU frame as illustrated in figure 3.6 is defined as:

- Positive X axis is downward.
- Positive Y axis is toward right.
- Positive Z axis is toward rear (away from train).

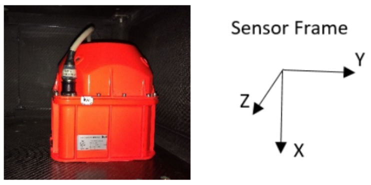


Figure 3.6.: The IMU frame mounted in RILA system.

Camera frame uses OpenCV convention and is coincident with the optical center of the camera. As shown in Figure 3.7, the camera frame in RILA system is defined as follows:

- Positive X axis is toward right.
- Positive Y axis is downward.
- Positive Z axis is pointing forward along the optical axis.

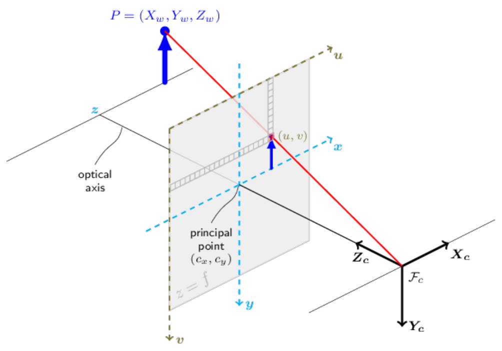


Figure 3.7.: Camera frame definition by OpenCV convention [9].

3.4.2. RILA ORB-SLAM3 System Calibration

The main objectives of system calibration is to determine the position and orientation of the various sensors to a common reference frame, i.e., the body frame. To be able to use ORB-SLAM3 tool, the RILA system should be calibrated according to the described requirements in 3.2.2. To do so, firstly the FUGRO HEC tool was employed to compute the intrinsic and extrinsic parameters of RILA's installed sensors. After that, the computed parameters are used to generate the calibration YAML file which later ORB-SLAM3 tool utilizes as input files.

As mentioned in section 3.2.2, the intrinsic parameters of the camera were estimated using the HEC tool. Moreover, these parameters for the IMU sensore were provided by the manufacturer, i.e. iMAR. Table 3.2 and Table 3.1 summarize the estimated intrinsic parameters for the IMU and camera, respectively.

Table 3.1.: RILA's IMU Intrinsic Parameters

Intrinsic Parameter	**Value**
Gyroscope Noise Density $[rad/s^{0.5}]$	0.00029888
Accelerometer Noise Density $[m/s^{1.5}]$	0.0001000000
Gyroscope Random Walk $[rad/s^{1.5}]$	0.0000000001
Accelerometer Random Walk $[m/s^{2.5}]$	0.0000031620

Table 3.2.: RILA's Camera Intrinsic Parameters

Intrinsic Parameter	Value
f_x	1499.84104007175
f_y	1499.8475095955
c_x	1014.58125785923
c_y	1021.26199168825
k_1	-0.171918629792281
k_2	0.118149760774721
k_3	-0.0233476114366727
p_1	-0.000448848328024488
p_2	-2.29280010419978E-05

At this stage, to correctly relate measured accelerations and gyroscope in the IMU frame to attitude and position changes in the real world a relation between the sensor and our defined body frames needs to be established. This can be done by rotating the body frame (ENU convention) step by step in the following order:

1. Rotate Yaw over body Z-axis
2. Rotate Pitch over the (rotated) body X-axis
3. Rotate Roll over the (twice rotated) body Y-axis

By applying the above-mentioned rotation steps, the Euler angles is computed as follows:

1. **Yaw**: Rotate -90 [degree (deg)] over body Z-axis.

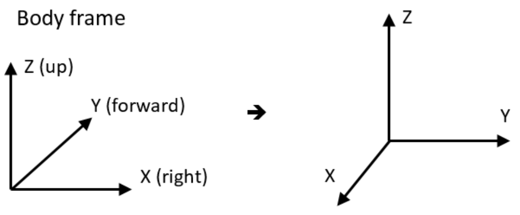


Figure 3.8.: IMU- Body alignment (step 1).

2. **Pitch**: No rotation over body X-axis (0 [deg], as Y-Axis is already pointing in right direction).
3. **Roll**: Rotate $+90$ [deg] over the body Y-axis, this makes the body frame aligned with IMU sensor frame as shown in Figure 3.9.

Utilizing the obtained Euler angles (Roll, Pitch and Yaw) and considering the proper rotation orders for ENU frames (ZXY), the rotation matrix for aligning IMU and body frames was computed and shown by Eq. (3.2). Assuming the accelometers and gyroscopes use the same

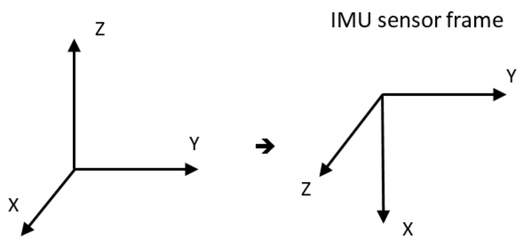


Figure 3.9.: IMU- Body alignment (step 3).

reference frames, these sensors' raw observations are multiplied by the computed R_{BI} matrix to transform the IMU measurements into our defined body frame.

$$R_{BI} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix} \tag{3.2}$$

After aligning the IMU and body frames, next, RILA camera's extrinsic parameters were generated. To do so, firstly, the HEC tool was utilized to estimate the boresight angles which are used to align the IMU and the camera frames. Secondly, by using Robotic Total Station (RTS) the offsets between the camera and the IMU origins were measured and computed. Table 3.3 presents these six parameters.

Table 3.3.: RILA Camera's Extrinsic Parameters

Boresight Angles [deg]	Roll	-0.1595
	Pitch	0.03
	Yaw	-179.943
Offsets [m]	Right	-0.1770
	Front	-0.1968
	Up	0.2777

Subsequently, the transformation matrix, T_{BC}, is computed. This matrix is used later by our SLAM tool to transform points expressed in the camera frame to the body/IMU frame. To compute T_{BC}, first, the body frame (ENU) was rotated to be aligned with our defined OpenCV camera frame using the rotation matrix, $R_{BC_{OpenCV}}$, as shown by Eq. (3.3).

$$R_{BC_{(OpenCV)}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.3}$$

Secondly, the camera to body rotation matrix was corrected utilizing the estimated borsight angles shown in Table 3.3 using the following equation:

$$R_{BC} = R_{BC_{(OpenCV)}} . R_{BC_{(BA)}} \tag{3.4}$$

Where R_{BC} is the rotation matrix to move from RILA camera frame to the body/IMU frame, $R_{BC_{(OpenCV)}}$ is the rotation matrix to move from our defined OpenCV camera frame to RILA body frame and $R_{BC_{(BA)}}$ is the computed boresignt angles rotation matrix.

Finally, Eq. (3.3), Eq. (3.4), and the computed extrinsic parameters shown in Table 3.3 were used to generate the transformation matrix, T_{BC}, presented in Eq (3.5).

$$T_{BC} = \begin{bmatrix} -0.999995631831873 & -0.002783274290178 & 0.000994837373168 & -0.1770 \\ -0.000993376063089 & -0.000526365888985 & -0.999999368071274 & -0.1968 \\ 0.002783796179806 & -0.999995988153541 & 0.000523598751674 & 0.2777 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

Furthermore, to generate the calibration file according to ORB-SLAM3 requirements some other parameters should be defined, namely, the camera frame rate, the camera resolution, the color order of the images (BGR or RGB or ignored if grayscale), and the IMU frequency. Table 3.4 summarizes these parameters for RILA setup. At this stage, by computing all the parameters, the settings file for the execution of ORB-SLAM3 tool is prepared.

Table 3.4.: RILA's Sensors Extra Parameters

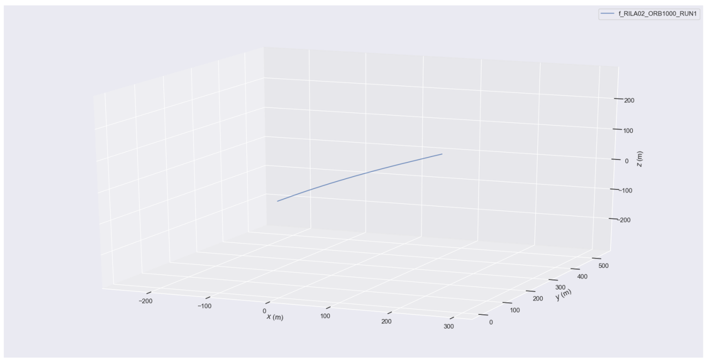
Sensors Parameter	**Value**
Camera Frame Rate [Hertz]	15
Camera Resolution	2016 × 2016
Color Order	RGB
IMU Frequency [Hertz]	300

3.4.3. Execution of RILA ORB-SLAM3

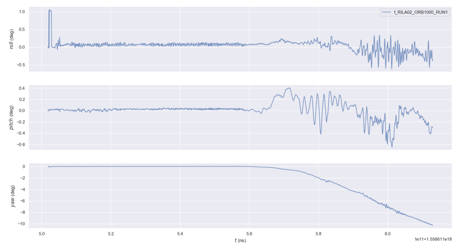
After calibration of RILA ORB-SLAM3, in this section first it is explained how the RILA dataset was used as input data for our SLAM tool. Next, using the initial defined setting parameters, RILA ORB-SLAM3 is executed and the results are presented. Furthermore, the necessity of fine-tuning at this stage is addressed.

As stated in section 3.3.2, the sequence of images captured by RILA's camera in combination with raw IMU data are used as inputs for our SLAM tool. Moreover, the default parameters shown in Tables 3.1, 3.2 and 3.4 together with the computed transformation matrix by Eq. (3.5) are used to generate the calibration input file which is presented in appendix C. It should be noted that for the ORB-extractor parameter at this stage, the recommended value by [11] was used, i.e., 1000.

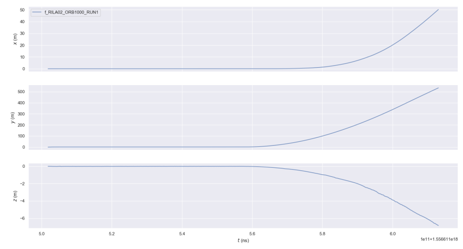
Figure 3.10 illustrates the generated RILA ORB-SLAM3-based trajectory utilizing the above mentioned parameters. In Figure 3.10a, the 3D view of this trajectory is shown while Figures 3.10b and 3.10c represent the orientations (Roll, Pitch and Yaw) and the positions (*xyz*) at each epochs, respectively.



(a) The 3D view



(b) The XYZ vs. time



(c) The RPY vs. time

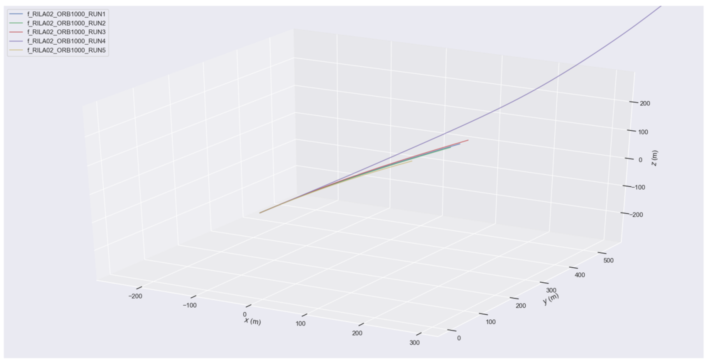
Figure 3.10.: RILA ORB-SLAM3 trajectory

To investigate the consistency of the results, the experiment was executed 5 times with the default parameters. The results are illustrated in Figure 3.11.

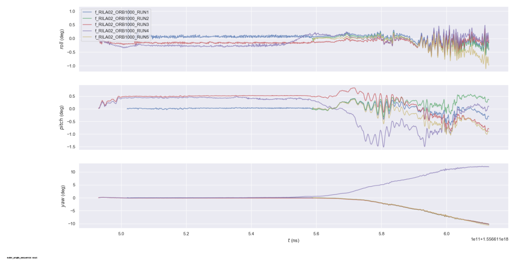
As shown in Figure 3.11a, after 5 runs the 3D trajectories differ significantly, indicating the instability in the system. Moreover, the initialization period between different executions are noticeably unpredictable as can be seen in Figure 3.11b and 3.11c. Additionally, during the execution several warnings appeared in SLAM tool which indicates the unreliability of the system using the default parameters. Consequently, the following section 3.4.4 states a proper fine tuning method to further stabilize the system for repeatability.

3.4.4. RILA ORB-SLAM3 Fine-tuning (1): Improving Precision and Repeatability

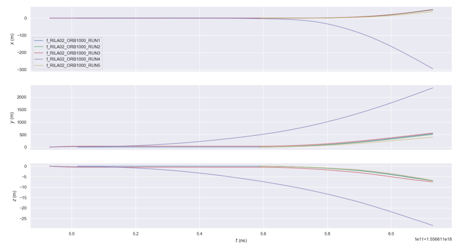
To improve the precision and repeatability of our SLAM tool, mainly two parameters are considered to be optimized, namely (1) IMU noise model and (2) ORB Extractor. The IMU noise model defines how much weight should be placed on IMU measurements during the motion estimation. In fact, if this model is too optimistic, the system relies too much on



(a) The 3D view of RILA ORB-SLAM3 trajectory



(b) The XYZ vs. time



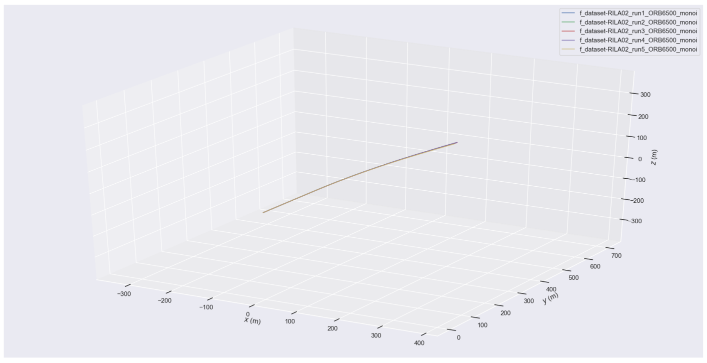
(c) The RPY vs. time

Figure 3.11.: RILA ORB-SLAM3 Trajectory

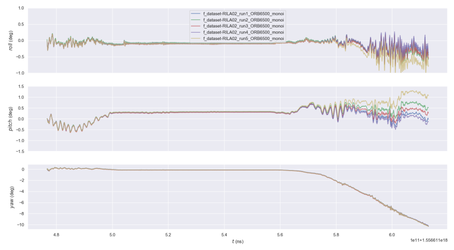
IMU which leads to instability. It is worth mentioning that, the IMU noise models are derived from a sensor that is static and at a constant temperature. Therefore, to consider unmodelled effects, it is advised to inflate the random walk parameters by 10 which improve the IMU initialization convergence [48].

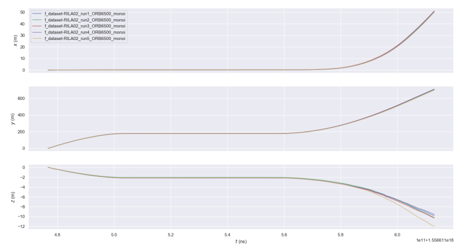
Furthermore, as mentioned earlier, the ORB-extractor parameter indicates the number of extracted features in each frame. By increasing this value, a denser point cloud is generated, and therefore with greater probability the neighbour frames are matched. As a result, the SLAM tool is able to track itself properly and the chance of loosing the track is reduced. This noticeably impacts the overall stability of the system as well as improving the speed of the initialization process. To find the optimal value, different experiments were conducted and eventually, by analysing the obtained results, we reached to the range of 6000 to 6500. Values above 6500 are not recommended as they highly affects the performance of the SLAM tool by increasing the feature extraction process time.

Figure 3.12 depicts the outcome of 5 runs using the fine-tuned IMU noise model and ORB-extractor parameters. The 3D view of the trajectories, presented in Figure 3.12a, shows that the output of the 5 runs are perfectly match, in contrast to Figure 3.11a. Moreover, as can be seen in Figure 3.12b and 3.12c, the overall trend of these 5 runs are highly aligned. Additionally, the initialization periods are remarkably similar between all the execution rounds.



(a) The 3D view of RILA ORB-SLAM3 trajectory





(b) The XYZ vs. time

(c) The RPY vs. time

Figure 3.12.: RILA ORB-SLAM3 Trajectory

4. RILA ORB-SLAM3 Performance Analysis

After implementing RILA ORB-SLAM3 and producing the SLAM-based trajectories using RILA dataset, in this chapter it is shown how we conducted the performance analysis for RILA ORB-SLAM3. To this end, Figure 4.1 depicts the workflow for the validation and trajectory correction phases.

For the validation, firstly, the Inertial Explorer (IE) tool used the raw data collected by RILA's GNSS and INS sensors to generate the ground-truth trajectory for our case study. As previously stated, it is assumed that the output trajectory of the RILA in open sky condition with adequate satellite coverage is accurate enough to be considered as the ground-truth. Next, we used EVO tool that is a python package to handle, evaluate, and compare the generated trajectories. Using this tool, first the SLAM-based trajectory (generated in previous phase 3.3.3) was aligned with the ground-truth employing the Umeyama's method [76]. Eventually, the defined performance metrics were computed by EVO tool to undertake the performance analysis.

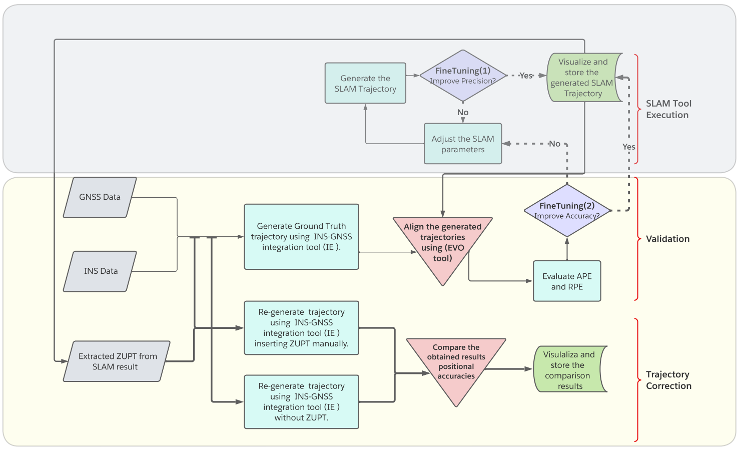

Generate the SLAM Trajectory
FineTuning(1) Improve Precision?
Yes
Visualize and store the generated SLAM Trajectory
No
Adjust the SLAM parameters
No
Yes
SLAM Tool Execution
GNSS Data
INS Data
Generate Ground Truth trajectory using INS-GNSS integration tool (IE).
Align the generated trajectories using (EVO tool)
FineTuning(2) Improve Accuracy?
Evaluate APE and RPE
Validation
Extracted ZUPT from SLAM result
Re-generate trajectory using INS-GNSS integration tool (IE) inserting ZUPT manually.
Re-generate trajectory using INS-GNSS integration tool (IE) without ZUPT.
Compare the obtained results positional accuracies
Visulaliza and store the comparison results
Trajectory Correction

Figure 4.1.: RILA ORB-SLAM3 Validation and Trajectory Correction Phase.

Subsequently, after generating the validated trajectory, we are interested in investigating which information can be extracted from this trajectory to assist INS-GNSS integration via IE tool. Furthermore, the effect of the selected information on improving the accuracy of RILA trajectory in the GNSS challenging area is examined in the trajectory correction phase. Section 4.5 discusses this phase in detail.

The rest of this chapter is structured as follow: firstly, two relevant performance metrics are defined. Next, our experimental setup used in validation and trajectory correction phases are briefly described. Following that, the obtained results are evaluated using our defined performance metrics. Afterwards, it is addressed how our SLAM tool was fine tuned for this case study. Finally, the trajectory correction using the extracted information is discussed.

4.1. Performance Metrics

To evaluate the quality of the estimated SLAM-based trajectory, it is common to compare it with a ground-truth trajectory acquired using a motion caption system or derived from high quality GNSS data. However, comparing these trajectories is not always straightforward. There are mainly two difficulties which need to be tackled. First, the generated SLAM-based and the ground-truth trajectories are often not expressed in the same reference frames, and, hence comparing them directly is not possible. To solve this first problem, a so called trajectory alignment is an essential step to transform the estimated SLAM-based trajectory into the ground-truth's reference frame. Since a trajectory is high-dimensional data which comprises the states at many different epochs, summarizing all its information into concise accuracy metrics is not an easy task. Thus this second problem can be solved, by defining some meaningful performance metrics and utilizing them to properly evaluate the obtained results [85, 68].

Figure 4.2 illustrates the process of trajectory evaluation. First off, the estimated SLAM-based trajectory (presented in blue) is aligned with the ground-truth (presented in black). Afterwards, the trajectory estimation error is computed using different performance metrics. In the following two subsections, we explain the chosen metrics in this research.

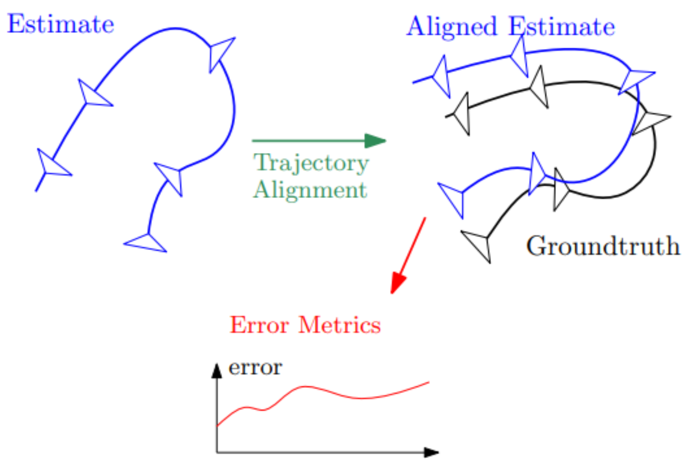


Figure 4.2.: The process of evaluating trajectory estimation errors [85].

4.1.1. Absolute Position Error (APE)

APE is one of the standard and commonly used metrics in robotic applications to evaluate the SLAM-based trajectory[68, 85]. APE compares the absolute distances between the SLAM-based and the ground-truth trajectories to evaluate the global consistency of the SLAM solution.

For computing APE, first the frames in which these two trajectories are expressed should be aligned [68]. This alignment can be performed in closed form using Umeyama's method which is described comprehensively in [76]. To this end, the absolute position error at epoch (i), i.e, F_i is computed using Eq. (4.1),

$$F_i = Q_i^{-1} S P_i \tag{4.1}$$

where P_i and Q_i are the estimated and the corresponding time-aligned ground-truth poses at epoch i, respectively. Moreover, S represents the rigid-body transformation to map the estimated poses $P_{1:n}$ onto the ground-truth poses $Q_{1:n}$ [68].

Assuming a sequence of n camera poses, the root mean squared error over all epochs of the translational parts is formulated according to Eq. (4.2).

$$RMSE(F_{1:n}) = \left(\frac{1}{n} \sum_{i=1}^{n} \| trans(F_i) \|^2 \right)^{\frac{1}{2}} \tag{4.2}$$

4.1.2. Relative Position Error (RPE)

The other standard and widely used metrics for evaluating the trajectory estimation in particular for visual odometry systems is RPE [68, 85, 58]. In contrast to APE which measures the global consistency of the SLAM systems, RPE assesses the local accuracy of the estimated trajectory by computing the difference between relative transformations over a fixed time interval Δ. The relative position error at epoch i, i.e, E_i is then defined using the following equation [68]:

$$E_i = \left(Q_i^{-1} Q_{i+\Delta} \right)^{-1} \left(P_i^{-1} P_{i+\Delta} \right) \tag{4.3}$$

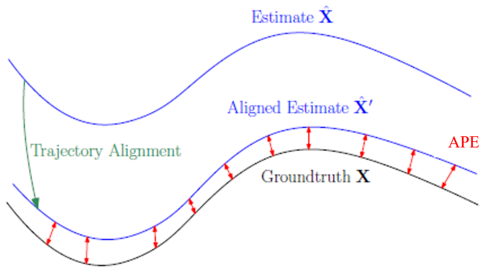
In Eq. (4.3), P_i and $P_{i+\Delta}$ are the estimated poses at epochs (i) and ($i + \Delta$) respectively. Moreover, the corresponding ground-truth poses at time indices (i) and ($i + \Delta$) are represented by Q_i and $Q_{i+\Delta}$.

Furthermore, Eq. (4.2) computes the root mean squared error over all epochs of the translation parts [68].

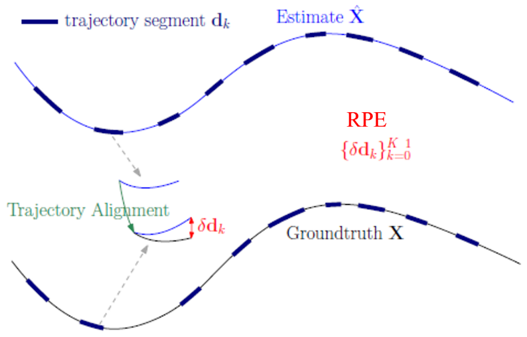
$$RMSE(E_{1:n}, \Delta) = \left(\frac{1}{m} \sum_{i=1}^{m} \| trans(E_i) \|^2 \right)^{\frac{1}{2}} \tag{4.4}$$

where n is the number of camera poses in a sequence; Δ is a fixed time interval over which the relative pose errors are computed. Moreover, the number of individual relative pose errors along the trajectory is represented by $m = n - \Delta$. Ultimately, $trans(E_i)$ indicates the translational components of the error E_i.

Figure 4.3 illustrates the process of generating APE and RPE graphically. As can be seen in Figure 4.3a, the estimated trajectory is first aligned with the ground-truth. After that the absolute differences between the aligned estimation and the ground-truth are calculated over all epochs to compute APE.


Estimate $\hat{\mathbf{X}}$
Aligned Estimate $\hat{\mathbf{X}}'$
Trajectory Alignment
APE
Groundtruth $\mathbf{X}$

(a) APE


trajectory segment $\mathbf{d}_k$
Estimate $\hat{\mathbf{X}}$
RPE
$\{\delta \mathbf{d}_k\}_{k=0}^{K-1}$
Trajectory Alignment
$\delta \mathbf{d}_k$
Groundtruth $\mathbf{X}$

(b) RPE

Figure 4.3.: Graphical representations of Absolute and Relative Position Errors [85]

Moreover, as illustrated in Figure 4.3b, to estimate RPE, first the sub-trajectories of length d are selected. Each sub-trajectory is then aligned with the corresponding part in the ground-truth trajectory. Next, the error of the end state for each sub-trajectory, i.e., δd_k, is calculated. This should be done for all the sub-trajectories and can be repeated for different lengths d to compute RPE.

4.2. Experimental Setup

This section presents the experimental setup used in the validation and trajectory correction phases. As discussed in the problem statement, we aim to evaluate the impact of utilizing ORB-SLAM3 in areas where GNSS coverage is poor, such as fully covered train stations for example Delft station where full GNSS blockages may occur. In these situations, accurate estimation of the train's position can be challenging.

To this end, for our experiments, we simulated the scenario where the train enters the station, stays there stationary for 1 minute, and then leaves the station slowly. In this scenario, the total duration that the train spent in the station without any GNSS signal is approximately 2 minutes.

To simulate this real-life example, we manually inserted 90 seconds of GNSS blockages before and after the station in our experiments. This 5-minute period with GNSS blockages, which we refer to as the *GNSS challenging area*, is illustrated in Figure 4.4. However, it is worth noting that in other semi-covered stations, such as Utrecht or Rotterdam stations for example, the GNSS blockage may be partial rather than full.

Since the RILA system has a high-grade IMU and the trajectory is normally processed in two different directions (forward and reverse), the positional accuracy in the central part of the trajectory, where the train is inside the station, is worse compared to the intervals outside the station. Therefore, we focused on the duration of 109 seconds, which corresponds to the central part of the GNSS challenging area, and refer to it as the *area of interest*, as shown with the blue box in Figure 4.4. We executed the developed ORB-SLAM3 tool for this area of interest to evaluate the generated trajectory against the ground-truth.

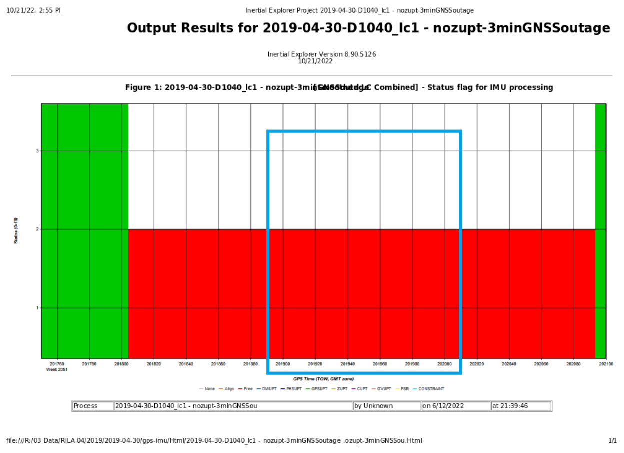


Figure 4.4.: The 5-minute GNSS challenging area including the area of interest highlighted by the blue box.

In the context of this study, the scenario where the train enters the station and stays there for a period of time represents a challenging situation for the RILA system. During this period, the train is subject to complete GNSS blockages, meaning that no satellite signals are received by the GNSS receiver onboard the train.

In real-life train stations, some stations can be particularly challenging in terms of GNSS reception due to their physical layout. For example, some stations are constructed like tunnels or are located underground, making it difficult for GNSS signals to penetrate and reach the receiver. In these cases, complete GNSS blockages can occur, similar to what is simulated in the current study.

When complete GNSS blockages occur, the system has no external reference for position and orientation, and must rely solely on internal sensors such as the IMU and visual sensors. This can result in a degradation of accuracy and reliability in the estimated position and orientation of the train. Therefore, the scenario of a complete GNSS blockage, particularly in the area of the station, is considered the worst case scenario for the RILA system, and is the focus of this study.

4.2.1. Effect of Motion Parameters

The drift in both the INS-GNSS system and the SLAM system is influenced by various factors, including time, speed, and distance traveled, but the relationships between these factors differ for each system.

In the INS-GNSS system, the generated drift is directly influenced by the speed of the train. The INS sensor readings accumulate errors over time, and the slower the train moves, the longer it spends in the area without GNSS coverage, which allows more error accumulation. For example, if a train travels a distance of 10 kilometers with 1 kilometer having no GNSS coverage and an INS sensor error rate of 0.1% per hour, the INS drift increases as the speed of the train decreases. At a speed of 100 km/h, the train spends 36 seconds in the area, accumulating an error of 0.6 meters, while at 25 km/h, it spends 144 seconds and accumulates an error of 2.4 meters.

In contrast, the drift in the SLAM system is not directly related to time and speed but is primarily influenced by the distance traveled. For a fixed distance, if the train speed changes, the SLAM drift remains approximately constant. However, if the traveled distance increases, the SLAM drift also increases. For instance, assuming a train travels a distance of 100 meters at a constant speed of 10 m/s while performing SLAM, the drift after traveling 100 meters is approximately 1 meter. If the same train were to travel 200 meters at the same speed, the drift in the SLAM system would increase to approximately 2 meters. However, if the train were to travel the same 100 meters at a different speed (e.g., 5 m/s or 20 m/s), the drift would be approximately the same.

It is worth noting that the quality of the sensors used, as well as their calibration and synchronization, can affect the drift in both systems. A high-quality IMU with proper calibration and synchronization can reduce the drift in the INS-GNSS system, while a high-quality camera with proper calibration and synchronization can reduce the drift in the SLAM system. In summary, the INS-GNSS system and the SLAM system are subject to drift, and their relationship with motion parameters varies. It is essential to consider the specific characteristics of each system and the conditions in which they are utilized to ensure accurate and reliable performance.

4.2.2. Ground-truth Generation

As mentioned previously, to generate the ground-truth for our experiments, we used the output trajectory of RILA utilizing only its GNSS/INS data, in open sky condition. In fact, we assumed that if the satellite reception is high, the generated trajectory of RILA is accurate enough to be considered as our ground-truth.

To generate the ground-truth trajectory, firstly the virtual reference stations and the RILA GNSS raw data were processed using IE tool. Consequently, this processed GNSS data was combined with the IMU data using the loosely coupled Kalman filter to generate a trajectory. IE tool supports multiple formats to export this trajectory, such as Google Earth, DXF, RIGEL POF, and Smoothed Best Estimate of Trajectory (SBET) formats, from which we chose the latter for our experiments. The SBET file logs various information including position, rotation, velocity, and acceleration at corresponding timestamps. However, to be able to conduct the performance analysis, the SBET file was converted to the EVO tool supported format, i.e., TUM to be further used as our ground-truth.

For converting the SBET file to TUM format, two major steps are required as follow:

- First off, the position information in SBET file which is expressed in geodetic coordinate (latitude, longitude, altitude) was converted into local tangent plane ENU coordinate (East, North, Up).
- Secondly, the rotational information in SBET file is described using Euler Angles. However, TUM requires Quanternion convention. Therefore, an Euler Angles to Quanternion conversion was done.

A python script was developed to carry out the above-mentioned conversion and generate the ground-truth trajectory for the experiments, which can be found in appendix B.

4.2.3. Trajectories Alignment Methods

As mentioned in the section 4.1, the groundtruth and the SLAM-based trajectories are defined in two arbitrary frames. In monocular VI-SLAM, the system is capable of estimating the camera pose (i.e., position and orientation) using a combination of visual and inertial measurements. However, the estimated camera pose has four degrees of freedom that are unobservable, which are the global position and the yaw angle [86, 87]. As a result, the estimated trajectory cannot be directly compared to the ground truth.

To overcome this challenge, an alignment process is performed between the estimated trajectory and the ground truth. The alignment process involves two steps: the first step is an origin alignment, where the global position of the estimated trajectory is aligned with the ground truth by finding a translation vector that minimizes the difference between the estimated positions and the ground truth positions. The second step is a yaw-only rigid body transformation, where the rotation around the gravity is estimated by finding a rotation matrix that minimizes the difference between the estimated orientation and the ground truth orientation[86, 87].

To perform the yaw-only rigid body transformation, the Umeyama method is often used, which is a closed-form solution that minimizes the distance between two sets of points in different coordinate systems[28, 76]. By applying this method, the estimated trajectory can be transformed to match the ground truth in terms of global position and orientation up to the yaw angle. Once the alignment process is completed, the accuracy of the estimated trajectory can be evaluated by comparing it to the ground truth.

For our evaluation, we utilized the following methods provided by EVO tool:

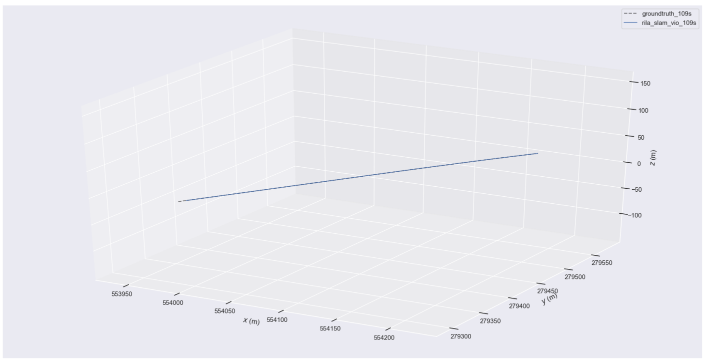
- ***The Special Euclidean Group in 3D (SE3)***: A rigid body transformation which consists of a translation and a rotation part. This method is mostly used for stereo SLAM cases where the scale is known.
- ***Similarity Transformation in 3D (Sim3)***: A transformation model with seven parameters(i.e., scale factor, three translation parameters and three rotation angles). This method is similar to SE3 with extra degree of freedom to estimate the unknown scale which is necessary for monocular SLAM.
- ***Scale Correction***: A scale factor estimation which used in monocular SLAM to correct the random scale without any translational or rotational alignment.

- ***Origin Alignment***: A simple method method which aligns only the origins of the two trajectories. This method is effective in drift estimation particularly for odometry cases.

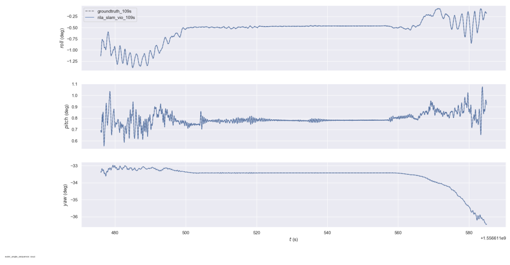
In the following sections, based on necessity, each of these alignment methods are used and mentioned explicitly.

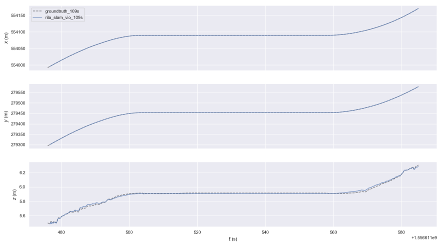
4.3. Validation Results

According to the validation workflow 4.1, the generated SLAM-based trajectory over the area of interest was compared to the ground-truth using Sim3 alignment method described briefly in the previous section. The obtained results from the EVO tool is presented in Figure 4.5.



(a) The 3D view





(b) The RPY vs. time

(c) The XYZ vs. time

Figure 4.5.: RILA ORB-SLAM3 trajectory over the area of interest vs the corresponding ground-truth trajectory

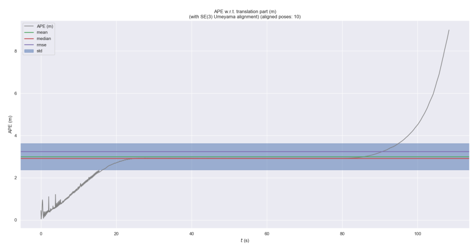
The 3D view of the trajectories, presented in Figure 4.5a, indicates that the generated SLAM result is perfectly match to the ground-truth. Moreover, as can be seen in Figures 4.5c and 4.5b, the translational and rotational parts of the trajectories are aligned with the same trends to a great extent.

In the next two subsections, we present the results of our evaluation for metrics defined in Section 4.1.

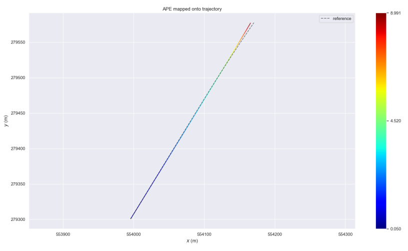
4.3.1. The Estimated APE Before Fine Tuning

To validate the obtained SLAM trajectory, we computed the APE with respect to the translational part using the EVO tool as explained in section 4.1.1. For aligning the trajectories, the SE3 method was used over the first 10 poses. Figure 4.6 shows the estimated APE for our case study.

Figure 4.6a presents the calculated APE for the period of 109 second, i.e., the area of interest inside the station, shown in Figure 4.4. The purple line and the blue bar address the Root Mean Squared Error (RMSE) and Standard Deviation (STD) values, respectively. Furthermore, to reduce the effect of outliers in the statistical analysis, the mean and median parameters are computed as well, shown in green and red lines. Table 4.1 summarizes all these values.



(a) The APE w.r.t. translational part in (m)



(b) The APE mapped onto trajectory

Figure 4.6.: APE RILA ORB-SLAM3 trajectory over the area of interest before fine tuning

Table 4.1.: APE related parameters w.r.t. translation part before fine tuning

Calculated Parameter [m]	**Value**
Maximum	8.990574
Mean	2.996627
Median	2.922912
Minimum	0.049612
RMSE	3.245827
STD	1.247246

As we observe, the APE increases with significant fluctuations in the first 15 seconds and then slightly stabilizes. The early fluctuations are as a result of SLAM initialization phase in which it uses different levels of optimizations. Moreover, the APE remains steady roughly between time 20 and 85 second, indicating the period in which the train is stationary. Eventually, while the train moves again to leave the station around time 85, a sharp rise occurs in APE value which is due to the accumulative nature of this metric.

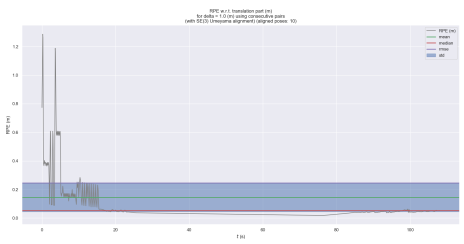
Figure 4.6b plots the 2-dimensional trajectory where APE is mapped onto it. The color bar shows the range of the errors between the SLAM-based and the ground-truth trajectories. In the initial part of the trajectory, the APE is within the acceptable range and aligned to the reference. However, by reaching to the end of the trajectory, it diverges from the reference and therefore, the error range increases from yellow towards red.

4.3.2. The Estimated RPE Before Fine Tuning

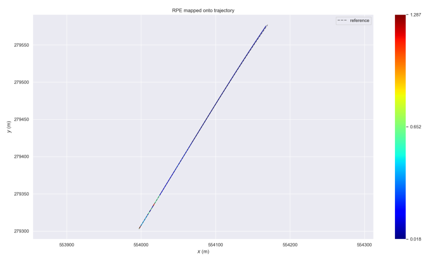
As stated in section 4.1.2, we considered the RPE to evaluate the local accuracy of the SLAM result. Figure 4.7 denotes the calculated RPE with respect to the translational part of the trajectory, where the Δ is chosen to be 1 meter. Similar to the APE estimation, the SE3 alignment method was applied over the first 10 poses.

In contrast to APE which considers the accumulated errors, RPE measures the errors in local intervals. Therefore, the maximum RPE value, i.e., 1.28 m, falls into the initial phase where SLAM is being initialized, as shown in Figure 4.7a. Furthermore, approximately after the first 25 seconds, the RPE remains zero for the period where the train is stationary. And finally, as it moves again to leave the station, RPE holds a steady value close to the median, i.e., 0.05 m. The summary of calculated RPE parameters for our case study is listed in Table 4.2.

In Figure 4.7b, the mapped RPE to the 2-dimensional trajectory emphasizes that, while the local accuracy along the path is acceptable, some peaks (between 0.4 to 1.3 meter) occurs in the early stage as a result of the SLAM initialization. Henceforth, this results in a RMSE value of 24 cm in a trajectory segment of 1 meter which is relatively large.



(a) The RPE w.r.t. translational part in (m)



(b) The RPE mapped onto trajectory

Figure 4.7.: RPE RILA ORB-SLAM3 trajectory over the area of interest before fine tuning

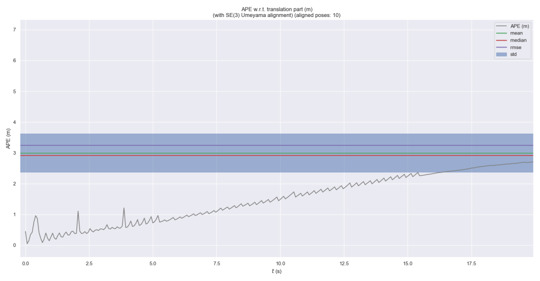
Table 4.2.: RPE related parameters w.r.t. translational part before fine tuning

Calculated Parameter [m]	Value
Maximum	1.286857
Mean	0.144211
Median	0.053447
Minimum	0.017912
RMSE	0.245927
STD	0.199206

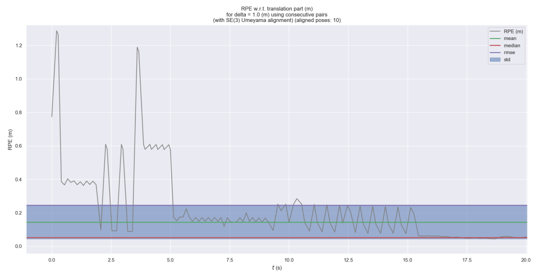
4.4. RILA ORB-SLAM3 Fine Tuning (2): Improving Accuracy

By evaluating the results obtained in Section 4.3, the impact of SLAM initialization phase is noticeable. Allocating sufficient time for initialization, is one of the key requirements for the SLAM tools. Additionally, it is advised to maintain the rotational movements as gentle as possible while the translational motion is also not too fast. This is the ideal situation to ensure the system has enough time to create an initial map and stabilize itself within that map. Nonetheless, considering our application, establishing this condition is relatively impossible while the train is approaching to the station. Hence, we decided to come up with an alternative solution to further fine-tune the generated SLAM-based trajectory for our use case.

Analyzing the RPE plots, reveals that the significant fluctuations were observed in the beginning of the trajectory, i.e., approximately the first 12 to 17 seconds as shown in Figures 4.8a and 4.8b. During this period, when the first major outliers happened, SLAM tool was creating a new map and initialized the IMU within it. Afterwards, several levels of the bundle adjustment were performed and eventually, to adjust the generated trajectory an scale reference estimation was done.



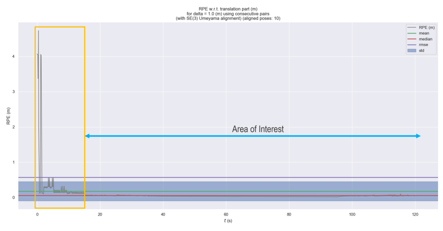
(a) Zoomed version of the APE plot



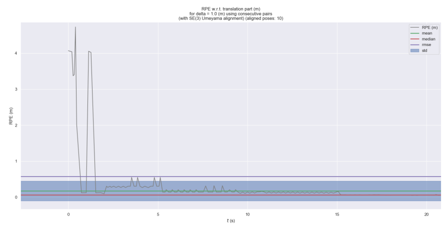
(b) Zoomed version of the RPE plot

Figure 4.8.: APE and RPE RILA ORB-SLAM3 trajectory over the area of interest before fine tuning zoomed on the first 20 seconds

Considering the results, we concluded that for our use case, a period of 15 seconds is required for the initialization. In this way, the SLAM system is able to generate more accurate results with the minimum amount of outliers. On the other hand, our area of interest, i.e., 109 seconds, is mainly the time that train spends inside the station. Therefore, we executed the SLAM tool over the duration of 124 seconds from which the first 15 seconds were considered as the initialization phase, and the rest 109 seconds was the area of interest over which the performance metrics were recomputed. In Figure 4.9a, RPE for this extended trajectory is calculated and shown. The orange box represents the initialization phase which is excluded for the performance metric evaluation.



(a) The RPE w.r.t. translational part in (m)



(b) Zoomed version of the RPE plot

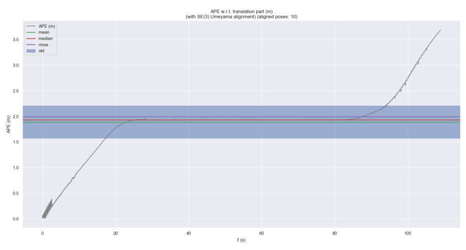
Figure 4.9.: RPE RILA ORB-SLAM3 trajectory over the extended dataset (124 seconds)

To better observe the behavior of the SLAM after this fine-tuning, Figure 4.9b depicts the results of the orange box in detail. Since the train speed in this initialization phase was higher compared to the previous experiment, we detect larger outliers in comparison to the results obtained before fine-tuning. However, as mentioned earlier, this phase is separated from the area of interest and therefore, does not have any influence on the final performance evaluation. In the following two sub-sections, the estimated APE and RPE for the area of interest after this fine-tuning is presented.

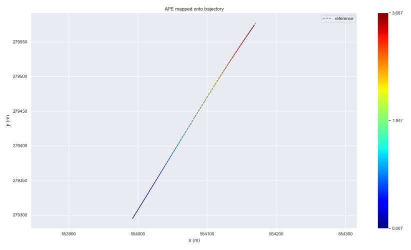
4.4.1. The estimated APE after fine-tuning

Figure 4.10, shows the APE results for the SLAM-based trajectory after fine-tuning over the area of interest. As can be seen, the maximum APE value is 3.7 m which is noticeably smaller comparing to the previous experiment. As expected, this maximum value still occurs at the end of the trajectory due to the accumulative nature of the APE. Since the fine-tuning reduced the initial error values, the maximum of APE significantly drops by 60%. Additionally, the rest of the APE parameters such as RMSE, STD decreased by 35% to 45%. Table 4.3 compares the statistical parameters before and after the fine-tuning.

According to the literature study, the obtained APE results are acceptable and expected for scenarios where loop closure is not possible, like the railway applications. In fact, V-SLAM framework without loop closure is downgraded to the VO and similar to any odometery systems, the drifts along the trajectory are accumulated. Therefore, the APE value grows quickly as the traveled distance increases.



(a) The APE w.r.t. translational part in (m)



(b) The APE mapped onto trajectory

Figure 4.10.: Fine-tuned APE RILA ORB-SLAM3 trajectory

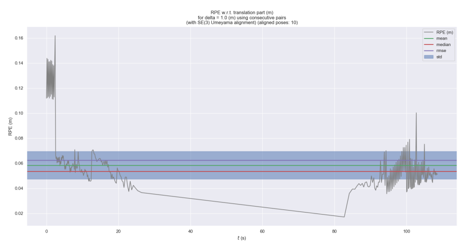
Table 4.3.: Comparing APE related parameters w.r.t translation part before and after fine-tuning

Calculated Parameter [m]	**Value Before Fine-Tuning**	**Value After Fine-Tuning**	**Improvement in Percentage [%]**
Maximum	8.990574	3.687226	59
Mean	2.996627	1.891725	36.9
Median	2.922912	1.932055	33.9
Minimum	0.049612	0.007321	85.2
RMSE	3.245827	1.994206	38.6
STD	1.247246	0.631060	49.4

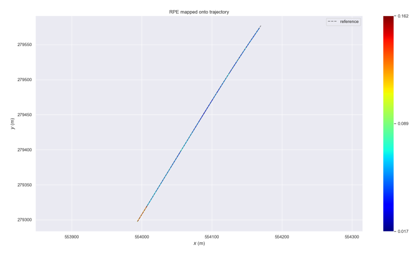
4.4.2. The estimated RPE after fine-tuning

The results for RPE after fine-tuning is presented in Figure 4.11. Overall, the fine-tuning enhanced the RPE value for all the parameters as expected. The improvement in the RMSE is remarkable as it reached to 0.06 m. Moreover, as can be seen, the RPE results in a downward trend while the train was entering the station because of the reduction in speed. On the other hand, it ascends when the train was accelerating to leave the station. This is expected, since the relative error between two corresponding trajectory segments directly depends on the speed value. A summary of comparison between all the RPE parameters before and after fine-tuning is given in Table 4.4.

By fine-tuning the initialization phase, the RPE results considerably improved. Additionally, since the remaining outliers still appear in the beginning of the trajectory, it is possible to further improve the result by choosing a more precise initialization time. To reduce the impact of these remaining outliers in our evaluation, Median is the best statistical parameter to take into consideration. As can be seen in Table 4.4, the Median value indicates that the the estimated relative error in 1 meter is about 5.2%. In other words, for our case study, the SLAM tool has 52 mm drift per meter.



(a) The RPE w.r.t. translational part in (m)



(b) The RPE mapped onto trajectory

Figure 4.11.: Fine-tuned RPE RILA ORB-SLAM3 trajectory

Table 4.4.: RPE related parameters w.r.t. translation part

Calculated Parameter [m]	**Value Before Fine-Tuning**	**Value After Fine-Tuning**	**Improvement in Percentage [%]**
Maximum	1.286857	0.161808	87.6
Mean	0.144211	0.058434	59.5
Median	0.053447	0.052914	1.0
Minimum	0.017912	0.017087	4.6
RMSE	0.245927	0.062430	74.6
STD	0.199206	0.021977	89

Furthermore, to deal with the unbounded errors in odometry as proposed in [74], the trajectory is divided into different segment lengths by choosing several Δ values including: 1 m, 10 m, 20 m, and 50 m. To test the evolution of RPE error, the RMSE and median in each case is computed and summerized in Table 4.5.

Table 4.5.: RPE RMSE and median w.r.t. translation part for multiple *Delta*

Delta **[m]**	**RMSE [%]**	**Median[%]**
1	6.2	5.3
10	4.3	4.3
20	4.3	4.3
50	4.3	4.3

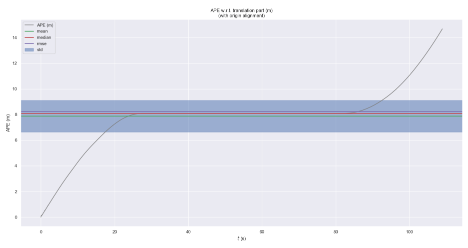
The different values for Δ shows that the RPE converged to 4.3%. Considering the prior researches [74] in railway applications, the obtained result is reasonable for monocular VI-SLAM systems without loop closure. However, comparing to other applications the estimated error is slightly larger. This is mainly due to the fact that the train movement is constrained to one direction with the minimum heading changes which results in unobservable IMU biases that causes scale drifts. Additionally, comparing RILA to indoor robotic systems, the velocity is higher while the camera frame rate is lower. Therefore, the distance between two consecutive frames is longer which leads to a larger relative error.

4.4.3. RPE and APE Evaluation in Conjunction

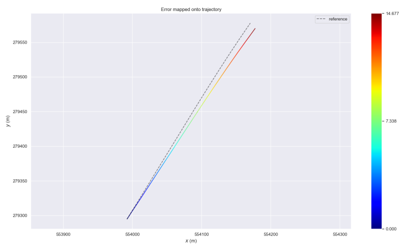
According to the obtained RPE, the expected error for the total path length is approximately 15 meter, i.e., 4.3% of the 350 meter (the length of the area of interest over which the SLAM trajectory is generated). However, the estimated absolute error, presented in section 4.4.1 is substantially less (around 11 m) than this expected value.

The cause of this noticeable difference highly depends on the selected method for the trajectory alignment. As mentioned in section 4.1, SE3 is one of the most widely used methods in the literature to align the translation and rotation parts of the trajectories. However, all these studies aimed to provide a benchmark on different SLAM algorithms. Therefore, the applied alignment method is a constant condition which has no impact on their obtained comparison results. Nonetheless, in our research, we are interested in estimating the achievable accuracy itself rather than performing comparison. Accordingly, we computed the APE by aligning only the origins of the trajectories without any further correction using the similarity transformation which reduces the estimated error.

Figure 4.12 illustrates the APE result with the origin alignment. As expected the estimated APE follows the relation to the computed RPE, in other words the APE is equal to the 4.3% of the travelled path length at each time. Furthermore, the Figure 4.12b shows the drift between the SLAM-based trajectory and the ground-truth.



(a) The APE w.r.t. translational part in (m)



(b) The APE mapped onto trajectory

Figure 4.12.: APE RILA ORB-SLAM3 trajectory over the area of interest using only origin alignment

4.5. RILA Trajectory Improvement Using the Extracted Zero Velocity Update (ZUPT)

Several information can be extracted from the SLAM-based trajectory to improve the accuracy of the RILA output in the GNSS challenging area. In our use-case, the IE tool can accept three aiding information, namely (1) Position, Velocity, Attitude relative updates (PVA) (2) DMI input (3) ZUPT. To be able to utilize the first two aiding information for RILA system, certain modification in the IE tool is required which can be done only by the manufacturer which is out of the scope of this research study. Therefore, ZUPT is selected as our aiding information to evaluate its impact on enhancing the positional accuracy of RILA.

ZUPT plays a vital role in reducing the uncertainty in generating the trajectory where the vehicle is stationary. In fact, by employing the ZUPT information, the accumulated IMU drifts are limited [73], [78].

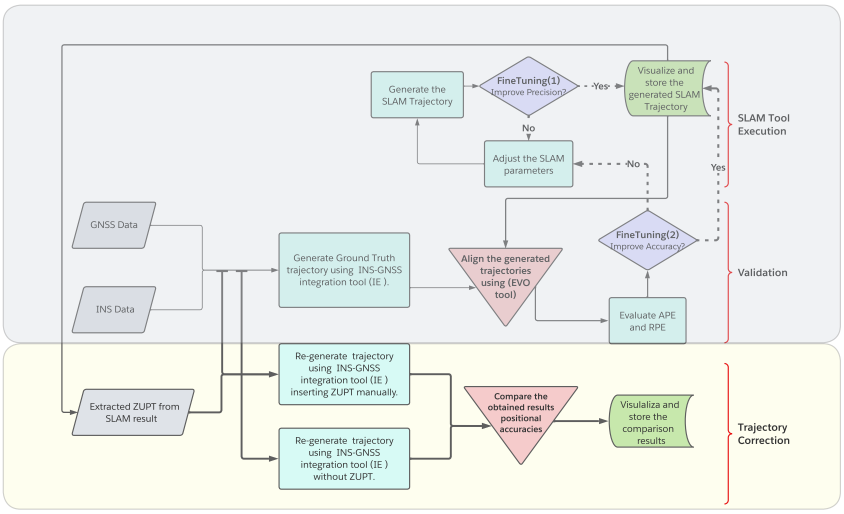

Generate the SLAM Trajectory
FineTuning(1) Improve Precision?
Yes
Visualize and store the generated SLAM Trajectory
No
Adjust the SLAM parameters
No
Yes
SLAM Tool Execution
GNSS Data
INS Data
Generate Ground Truth trajectory using INS-GNSS integration tool (IE).
Align the generated trajectories using (EVO tool)
FineTuning(2) Improve Accuracy?
Evaluate APE and RPE
Validation
Re-generate trajectory using INS-GNSS integration tool (IE) inserting ZUPT manually.
Extracted ZUPT from SLAM result
Re-generate trajectory using INS-GNSS integration tool (IE) without ZUPT.
Compare the obtained results positional accuracies
Visulaliza and store the comparison results
Trajectory Correction

Figure 4.13.: RILA ORB-SLAM3 Trajectory Correction Phase.

Figure 4.13 presents the steps of the trajectory correction phase using the ZUPT signals. Initially, the ZUPT information was extracted from the generated fine-tuned trajectory using a custom script. Then, to evaluate the impact of ZUPT on the positional accuracy of the trajectory, we manually input ZUPT signals to the IE tool and generated trajectories with and without ZUPT information over the 5-minute GNSS challenging area, as shown in Figure 4.4. Finally, we used the positional accuracy plots, computed by IE, to compare the impact of the ZUPT signals. These plots illustrate the standard deviation of the east, north, and up directions over time which enable us to understand how accurate the generated trajectories are. It is noted in [1], that the estimated error plot does not consider the possible systematic errors, such as problems in base stations. In the following section, the obtained results are illustrated and analyzed.

4.5.1. ZUPT Correction Results Analysis

Figure 4.14 depicts the estimated position accuracy plot over the 5-minute GNSS blocked area without any aiding information. As can be seen, the shape of the estimated plot has a smooth curve which is due to the fact that our processing is performed in both direction, (i.e., forward and reverse processing) also different filtering and smoothing steps are done by IE. It is clearly apparent that the maximum error occurs in the middle of the curve where the train is stationary inside the station. Table 4.6 summarizes the maximum estimated error for each direction as well as the calculated overall maximum estimated error for our case study where we have 5 minutes of the fully GNSS blockages.

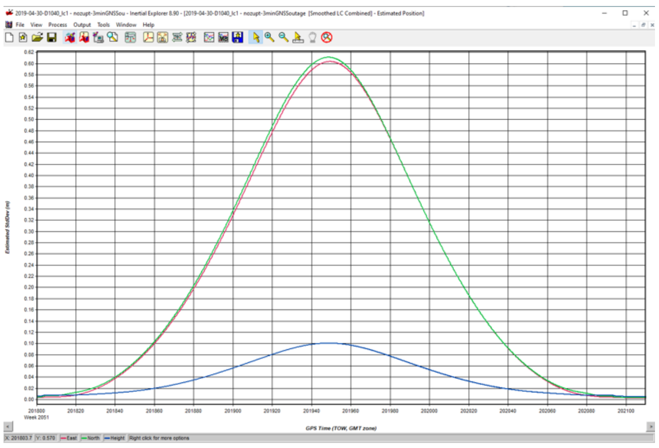


Figure 4.14.: Estimated Positional accuracy without utilization of ZUPT signals. Red, green, and blue lines shows the estimated error for east, north and height directions, respectively

Table 4.6.: Maximum standard deviation error in the absence of ZUPT information

North [cm]	East [cm]	Height [cm]	Overall [cm]
61	60	10	86

Figure 4.15 highlights how the extracted ZUPT information improves the positional accuracy in our case study. As can be seen, while entering the blocking GNSS area from both direction, the estimated error increases gradually. However, from the moment that the train is stationary, this growing trend is stopped due to the fact that the ZUPT can help to mitigate the IMU drifts. The maximum estimated errors for all the directions along with the computed overall error is listed in Table 4.7.

Table 4.7.: Maximum standard deviation error using the ZUPT information

North [cm]	East [cm]	Height [cm]	Overall [cm]
36	34	22	63

Comparing the overall estimated errors for both cases, around 30% of improvement is achieved with only 51 seconds of stationary condition in our case study. It is noticeable that the East and North error curves are significantly suppressed after applying the ZUPT information. Although the trend of height error curve is aligned with the other directions, the estimated error values for this direction slightly rose up.

In the beginning, the reason of this unexpected behavior was not clear to the author. However, after further investigation, we realized that in the complete dataset (from which the

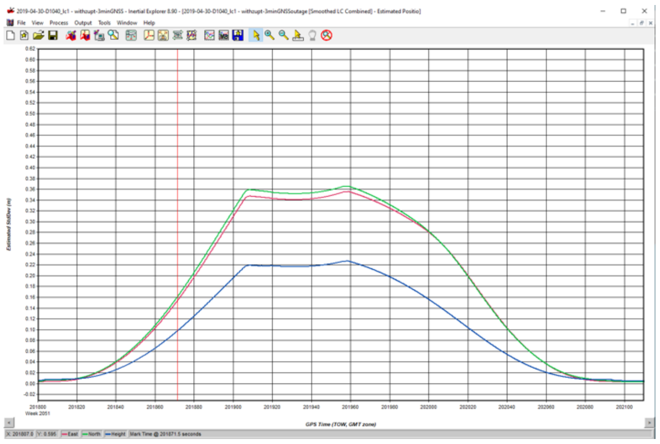


Figure 4.15.: Estimated Positional accuracy with utilization of ZUPT signals. Red, green, and blue lines shows the estimated error for east, north and height directions, respectively

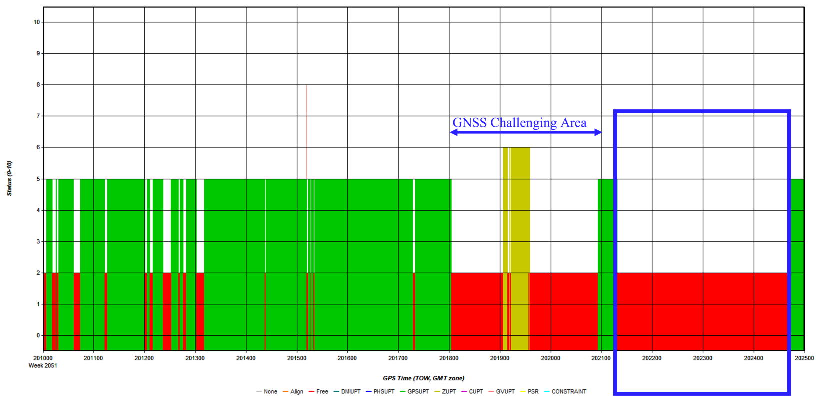


Figure 4.16.: The complete dataset. The vertical yellow bars in the 5-minute GNSS challenging area represents the ZUPT signals inserted manually. The blue box shows the problematic area with no base station.

5-minute GNSS challenging area was derived) one of the base stations immediately after the area of our case study was absent. In Figure 4.16, this problematic area is shown with the blue box. Therefore, having outliers in the dataset could be one of the reasons that leads to misbehavior of the height direction.

5. Conclusion and Future Work

This chapter presents the conclusions of the project. The first part focuses on providing answers to the research questions. In the second part, we propose some suggestions for the possible future work to further continue this research.

5.1. Conclusion

This dissertation set out to propose a solution to adapt a monocular VI-SLAM method for railway application. The primary motivation behind this work was to improve the accuracy of the trajectory estimation in the GNSS challenging area.

Most of the existing works in the domain of SLAM fusion with the INS/GNSS system has significantly focused on other applications such as ground robotic rather than mobile mapping systems for railway. However, in this study, we have aimed at integrating the SLAM framework for the railway application. Such an application has a distinctive characteristic of that the motion is limited to the rail path in contrast to other applications where the vehicle can move freely in arbitrary directions. Therefore, we have further evaluated the applicability, challenges and limitations of the proposed SLAM-based solution using real-world case study based on the RILA system. The obtained results have demonstrated the effectiveness of our proposed solution.

To perfectly be utilized in railway application, initially we have formulated the SLAM problem mathematically in order to understand its important concepts. Furthermore, we have reviewed studies on the existing SLAM paradigms and outlined their advantages and disadvantages for different use cases. Later, we classified the SLAM methods based on the sensor's input data into two main categorizes, namely visual, and non-visual SLAM and addressed their pros and cons according to the railway constraints. Overall, the result has shown that considering the challenges including high speeds, constrained motion, and dynamic environment the V-SLAM methods were suggested due to the possibility to extract intense visual information from the images, their simplicity and cost-efficiency.

In order to answer the first research question of this work 1.3, firstly, a comprehensive literature study was conducted to identify, classify and analyze the monocular VI-SLAM methods considering the railway application constraints. Consequently, a set of key requirements based on the RILA system was defined to assess the applicability of the methods and select the most suitable solution for our case study. The result of this study revealed that ORB-SLAM3 technique fulfills the defined criteria and therefore was chosen as our monocular VI-SLAM algorithm during this research project.

After identifying the SLAM method, we have designed an end-to-end pipeline that covers all the phases to adapt the selected technique for RILA system. This pipeline consists of five phases, namely SLAM Tool Installation, Input Data Preparation, SLAM Tool Execution, Validation and Trajectory Correction as described in Chapter 3. To implement the selected method according to the pipeline, firstly we have calibrated sensors of the RILA system and generated the intrinsic and extrinsic parameters based on ORB-SLAM3 format. Subsequently, we executed 5 times the ORB-SLAM3 using the RILA dataset which comprised of images captured by the camera along with the raw IMU data and the default parameters. However, by comparing the generated trajectories of these runs, we noticed inconsistency in the results. To overcome this limitation, we have performed a fine-tuning for precision and repeatability to make the system stable. To this end, we optimised the IMU noise model to reduce the dependency on IMU measurements for the trajectory estimation. Additionally, we increased the ORB-extractor parameter to an optimal value to ensure the SLAM tool is able to track itself properly. The generated results after this fine-tuning show the overall trend of these 5 runs are highly aligned.

Finally, to answer the second research question (i.e., evaluate our proposed solution), we have implemented a case study using RILA dataset. In this case study, we have illustrated a scenario in which the train entered the station, stayed there stationary for 1 minute, and then left the station slowly. In particular, we manually inserted GNSS blockages before and after the station to evaluate the impact of utilizing ORB-SLAM3 in the areas where GNSS coverage is poor. Additionally, we generated a ground-truth trajectory for our case study and conducted several experiments to validate the obtained SLAM-based trajectory of the proposed solution according to two criteria. Then, considering the evaluation results we decided on the strategy for fine-tuning to improve the accuracy. Additional experiments were performed on the fine-tuned results to determine the extent to which our proposed solution is applicable. In our experiments, we measured the APE and RPE to evaluate the absolute and local estimated errors of our solution comparing to the ground-truth trajectory. Furthermore, we revealed that our proposed solution can achieve higher accuracy by allocating sufficient time for SLAM initialization. Having compared the results before and after the fine-tuning, we concluded that both APE and RPE are significantly reduced and the estimated error is around 4.3% of the travelled path. Last but not least, we assessed the impact of ZUPT information on INS/GNSS integration result. Having used the ZUPT signal as the aiding information, the estimated positional accuracy of the trajectory was improved by 30% with 51 seconds of stationary condition.

Having provided the research findings for the two questions, we can now argue that the proposed monocular VI-SLAM solution can integrate to INS/GNSS system as an aiding information algorithm and enhance the trajectory estimation for rail vehicles. In particular, in sites with GNSS blockage where the vehicle is stationary, the ZUPT information remarkably reduces the IMU drift and therefore enhances the trajectory estimation. Please note that although in our specific case study the stationary period was only 15% of the total GNSS problematic dataset, an overall 30% improvement in accuracy was achieved. Hence, it is expected that in real-life scenarios where the train stays much longer in the stations, the proposed solution improves the estimated positional accuracy more significantly.

By summarizing, our research brings the following scientific achievements:

- A complete pipeline to implement and validate the ORB-SLAM3 for RILA system.
- A truly test over a real-life case study to analyse the performance and explore the achievable accuracy of the proposed technique considering the railway application.
- A correction to the trajectory estimation using the ZUPT information.

5.2. Limitations and Future Work

Despite the promising findings provided in this dissertation, a number of research challenges are still open for further investigation. Firstly, since an specific motion with sufficient acceleration and rotation is not possible for train, precise IMU initialization is not possible which results in imprecise scale estimation, i.e., around 1% drift in our case study. Additionally, due to limited yaw angle movement, the bias in this angle is relatively larger than the rest and therefore increases the absolute error along the travelled path. One interesting solution to tackle this issue is adapting a stereo VI-SLAM method which provides an estimation for yaw separately.

Additionally, as the travelled distance increases, the odometer drifts grows exponentially due to the lack of loop closure for rail application. It is possible to reduce this effect by applying specific modifications in RILA setup and the SLAM pipeline:

- Use camera with higher frame rate: Higher frame-rates improve the rapid motion tracking and feature extraction which enhance the system accuracy.
- Modify RILA camera setup: The position and orientation of the RILA's camera should be adjusted in such a way to enable adapting stereo SLAM. Providing a precise scale estimation for all scenes, the stereo SLAM method mitigates the scale drift issue.
- Implement the bi-directional processing of the SLAM solution: Similar to INS/GNSS integration tools, the SLAM solution should be processed in both forward and reverse directions to reach smoother trajectory results.
- Implement multi-map SLAM: The collected data from the typical 4 RILA's runs can be processed under the framework of multi-map SLAM. to imitate the loop closure condition and provide accurate localization.

As a closing note, it would be interesting to explore how deep learning can be used in the VI-SLAM pipeline. Specially, deep learning can improve the image feature extraction, enhance the camera pose and image depth estimation, and extract semantic information for data association.

A. ORB-SLAM3 Installation Bash Script

Listing A.1: bash version

```
#Install instructions for ORB-SLAM3
on a clean Ubuntu 18.04

RUN sudo sh -c 'echo
"deb http://packages.ros.org/ros/ubuntu
$(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
RUN sudo apt-get update
RUN sudo apt-get install curl
RUN sudo curl -s
https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc |
sudo apt-key add -

RUN sudo apt-get update && sudo apt-get install -y --no-install-recommends
apt-utils
RUN sudo apt-get install -y \
        cmake \
        build-essential \
        git \
        unzip \
        pkg-config \
        libgtk2.0-dev \
        python-dev \
        python-numpy \
        libgl1-mesa-dev \
        libglew-dev \
        libpython2.7-dev \
        libeigen3-dev \
        qt5-default \
        ros-melodic-cv-bridge \
        ros-melodic-image-geometry \
        ros-melodic-geometry \
        ros-melodic-image-pipeline \
    && apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

# Build OpenCV (3.0 or higher should be fine)
RUN cd /tmp && git clone https://github.com/opencv/opencv.git && \
    cd opencv && \
    git checkout 4.4.0 && \
    mkdir build && cd build && \
    cmake -DENABLE_PRECOMPILED_HEADERS=OFF -D WITH_QT=ON -D WITH_GTK=ON
    -D WITH_OPENGL=ON -D CMAKE_BUILD_TYPE=Release
    -D CMAKE_INSTALL_PREFIX=/usr/local .. && \
    make -j3 && sudo make install && \
    cd / && rm -rf /tmp/opencv

# Build Pangolin
RUN cd /tmp && git clone https://github.com/stevenlovegrove/Pangolin && \
    cd Pangolin && mkdir build && cd build && \
    cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_CXX_FLAGS=-std=c++11 .. && \
    make -j1 && sudo make install && \
    cd / && rm -rf /tmp/Pangolin

# Build ORB-SLAM3 for ROS
RUN git clone https://github.com/UZ-SLAMLab/ORB_SLAM3 ~/ORB_SLAM3

DO Modify
CHANGE Tracking.cc remove comment from "mpFrameDrawer->Update(this);"

DO Modify build.sh and build_ros.sh (edit all "make -j" to "make -j1").
   To edit the files use "sudo chmod -R o+rw bash.sh"
   and "sudo chmod -R o+rw bash_ros.sh" to give write permissions.

FOLLOW steps on http://wiki.ros.org/melodic/Installation/Ubuntu
RUN sudo ln -s /usr/include/eigen3/Eigen /usr/include/Eigen

RUN . /opt/ros/melodic/setup.sh && \
    export ROS_PACKAGE_PATH=${ROS_PACKAGE_PATH}:~/ORB_SLAM3/Examples/ROS
    && \
    cd ~/ORB_SLAM3/ && \
    sudo chmod +x build.sh && ./build.sh \
    sudo chmod +x build_ros.sh && ./build_ros.sh

# Download dataset
RUN cd ~ && \
    mkdir -p Datasets/EuRoc && \
    cd Datasets/EuRoc/ && \
    curl -O
    http://robotics.ethz.ch/~asl-datasets/ijrr_euroc_mav_dataset/
    machine_hall/MH_01_easy/MH_01_easy.zip && \
    mkdir MH01 && \
    unzip MH_01_easy.zip -d MH01/

#Examples
RUN cd ~/ORB_SLAM3/

# Mono
./Examples/Monocular/mono_euroc ./Vocabulary/ORBvoc.txt
./Examples/Monocular/EuRoC.yaml ~/Datasets/EuRoc/MH01
./Examples/Monocular/EuRoC_TimeStamps/MH01.txt dataset-MH01_mono
```

```
# Mono + Inertial
./Examples/Monocular-Inertial/mono_inertial_euroc
./Vocabulary/ORBvoc.txt ./Examples/Monocular-Inertial/EuRoC.yaml
~/Datasets/EuRoc/MH01
./Examples/Monocular-Inertial/EuRoC_TimeStamps/MH01.txt dataset-MH01_monoi

# Stereo
./Examples/Stereo/stereo_euroc ./Vocabulary/ORBvoc.txt
./Examples/Stereo/EuRoC.yaml ~/Datasets/EuRoc/MH01
./Examples/Stereo/EuRoC_TimeStamps/MH01.txt dataset-MH01_stereo

# Stereo + Inertial
./Examples/Stereo-Inertial/stereo_inertial_euroc ./Vocabulary/ORBvoc.txt
./Examples/Stereo-Inertial/EuRoC.yaml ~/Datasets/EuRoc/MH01
./Examples/Stereo-Inertial/EuRoC_TimeStamps/MH01.txt dataset-MH01_stereoi


#ROS examples
ROS datasets can be downloaded in
https://projects.asl.ethz.ch/datasets
/doku.php?id=kmavvisualinertialdatasets

#ROS Mono-Inertial
RUN cd ~/ORB_SLAM3/Examples/ROS/ORB_SLAM3 && \
    export ROS_PACKAGE_PATH=${ROS_PACKAGE_PATH}:
    ~/ORB_SLAM3/Examples/ROS && \
    roscore && \
    rosparam set use_sim_time true
ctrl + shift + t to open terminal tab
RUN export ROS_PACKAGE_PATH=${ROS_PACKAGE_PATH}:
~/ORB_SLAM3/Examples/ROS && \
    rosrun ORB_SLAM3 Mono ~/ORB_SLAM3/Vocabulary/ORBvoc.txt
    ~/ORB_SLAM3/Examples/Monocular-Inertial/EuRoC.yaml
ctrl + shift + t to open terminal tab
RUN export ROS_PACKAGE_PATH=${ROS_PACKAGE_PATH}:
~/ORB_SLAM3/Examples/ROS && \rosbag play --pause
~/Downloads/V1_02_medium.bag /cam0/image_raw:=/camera/image_raw /imu0:=/imu
PRESS space in rosbag tab

# Evaulation of Monocular-Inertial Example
cd ~/ORB_SLAM3/Examples
RUN ./Monocular-Inertial/mono_inertial_euroc
../Vocabulary/ORBvoc.txt ./Monocular-Inertial/EuRoC.yaml
~/Datasets/EuRoc/MH01 ./Monocular-Inertial/EuRoC_TimeStamps/MH01.txt
~/Datasets/EuRoc/MH01 ./Monocular-Inertial/EuRoC_TimeStamps/MH01.txt
~/Datasets/EuRoc/MH01 ./Monocular-Inertial/EuRoC_TimeStamps/MH01.txt
dataset-MH01_monoi
RUN python ../evaluation/evaluate_ate_scale.py
../evaluation/Ground_truth/EuRoC_left_cam/MH01_GT.txt
f_dataset-MH01_monoi.txt --plot MH01_monoinertial.pdf --verbose --verbose2
```

B. SBET to TUM converter python script

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import numpy as np
import pandas as pd
import pyproj

import cut_sbet_in_tum


def sbet_record_types():
    """Function sbet_record_types
       Get a list of the sbet record types
       Arguments:
       Returns: list of the data types in a sbet record
    """

    return [
        ('time_sbet', np.float64),
        ('lat', np.float64),
        ('lon', np.float64),
        ('alt', np.float64),
        ('xvel', np.float64),
        ('yvel', np.float64),
        ('zvel', np.float64),
        ('roll', np.float64),
        ('pitch', np.float64),
        ('heading', np.float64),
        ('wander', np.float64),
        ('xacc', np.float64),
        ('yacc', np.float64),
        ('zacc', np.float64),
        ('xang_rate', np.float64),
        ('yang_rate', np.float64),
        ('zang_rate', np.float64),
        ]


def con_rad_to_deg_sbet(df):
    """"Function con_rad_to_deg
        Read a dataframe and convert the variable from radians to degrees
    """

    df['lat'] = np.rad2deg(df['lat'])
    df['lon'] = np.rad2deg(df['lon'])

    return df


def readSbet(filename):
    """Function readSbet
       Read an sbet file into a np array.
       Arguments:
                 filename: string of filename to read into a np array
       Returns: 2-d np array of sbet data
    """

    if not isinstance(filename, str):
        raise TypeError('argument 1 to readSbet must be a string')
    np_data = np.fromfile(filename, dtype=np.dtype(sbet_record_types()))
    return pd.DataFrame(np_data)


if __name__ == '__main__':

    # print(pyproj.show_versions())

    filepath = \
        input('Enter the full filepath location and name for SBET file: \n'
              )
    sbet_data = readSbet(filepath)
    sbet_data = con_rad_to_deg_sbet(sbet_data)

    sbet_data.to_csv(r'SBET_lc1_ref.csv', index=False)

    print 'SBET to Text created successfully!'

df = pd.read_csv(r'SBET_lc1_ref.csv')
df.columns = [
    'GpsTimeOfWeek',
    'Latitude',
    'Longitude',
    'Altitude',
    'EWSpeed',
    'NSSpeed',
    'VertSpeed',
    'Roll',
    'Pitch',
    'Heading',
    'Wander',
    'EWAccuracy',
    'NSAccuracy',
    'VertAccuracy',
    'XAccuracy',
    'YAccuracy',
    'ZAccuracy',
    ]
print df[0:1]
```

```
df.iloc[:, 9] = np.multiply(df.iloc[:, 9], -1)

tempX = df['Latitude'].values
tempY = df['Longitude'].values
tempZ = df['Altitude'].values
concar = np.vstack((tempX, tempY, tempZ))
etrs89 = pyproj.Proj('epsg:4258')
ostn15 = pyproj.Proj(init='epsg:27700',
                     nadgrids=r'OSTN15_NTv2_ETRStoOSGB.gsb',
                     geoidgrids=r'OSGM15_Malin.gtx')
res = pyproj.transform(etrs89, ostn15, *concar)  # Converting from ETRS89 to OSGB

df['East'] = res[0]
df['North'] = res[1]
df['Height'] = res[2]
df['qx'] = np.sin(df['Roll'] / 2) * np.cos(df['Pitch'] / 2) \
    * np.cos(df['Heading'] / 2) - np.cos(df['Roll'] / 2) \
    * np.sin(df['Pitch'] / 2) * np.sin(df['Heading'] / 2)
df['qy'] = np.cos(df['Roll'] / 2) * np.sin(df['Pitch'] / 2) \
    * np.cos(df['Heading'] / 2) + np.sin(df['Roll'] / 2) \
    * np.cos(df['Pitch'] / 2) * np.sin(df['Heading'] / 2)
df['qz'] = np.cos(df['Roll'] / 2) * np.cos(df['Pitch'] / 2) \
    * np.sin(df['Heading'] / 2) - np.sin(df['Roll'] / 2) \
    * np.sin(df['Pitch'] / 2) * np.cos(df['Heading'] / 2)
df['qw'] = np.cos(df['Roll'] / 2) * np.cos(df['Pitch'] / 2) \
    * np.cos(df['Heading'] / 2) + np.sin(df['Roll'] / 2) \
    * np.sin(df['Pitch'] / 2) * np.sin(df['Heading'] / 2)
s(df['Heading'] / 2) * np.cos(df['Roll'] / 2) * np.cos(df['Pitch'] / 2) \
    - np.sin(df['Heading'] / 2) * np.sin(df['Roll'] / 2) \
    * np.sin(df['Pitch'] / 2)
print df[0:1]
np.savetxt(r'44_Corrected_SBET_lc1_ref.txt', df[[
    'GpsTimeOfWeek',
    'East',
    'North',
    'Height',
    'qx',
    'qy',
    'qz',
    'qw',
    ]], delimiter=' ', fmt='%1.7f')
```

C. RILA ORB-SLAM3 Caliberation Settings

```
%YAML: 1.0

#--------------------------------------------------------------
# Camera Parameters. Adjust them!
#--------------------------------------------------------------
Camera.type: "PinHole"

# Camera calibration and distortion parameters (OpenCV)
Camera.fx: 1499.84104007175
Camera.fy: 1499.8475095955
Camera.cx: 1014.58125785923
Camera.cy: 1021.26199168825

Camera.k1: -0.171918629792281
Camera.k2:  0.118149760774721
Camera.k3: -0.0233476114366727
Camera.p1: -0.000448848328024488
Camera.p2: -2.29280010419978E-05

Camera.width: 2016
Camera.height: 2016

# Camera frames per second
Camera.fps: 15.0

# Color order of the images (0: BGR, 1: RGB. It is ignored if images are grayscale)
Camera.RGB: 1

# Transformation from camera to body-frame (imu)
Tbc: !!opencv-matrix
   rows: 4
   cols: 4
   dt: f
   data: [ -0.99999563183187290, -0.00278327429017821, 0.00099483737316777, -0.1770,
          -0.00099337606308907, -0.00052636588898528, -0.99999936807127443, -0.1968,
           0.00278379617980556, -0.99999598815354085, 0.00052359875167370, 0.2777,
           0.0, 0.0, 0.0, 1.0]

# IMU noise
IMU.NoiseGyro: 0.00029888
IMU.NoiseAcc:  0.0001000000
IMU.GyroWalk:  0.000000001
IMU.AccWalk:   0.000031620
IMU.Frequency: 300

#--------------------------------------------------------------
# ORB Parameters
#--------------------------------------------------------------

# ORB Extractor: Number of features per image
```

```
ORBextractor.nFeatures: 6500

# ORB Extractor: Scale factor between levels in the scale pyramid
ORBextractor.scaleFactor: 1.2

# ORB Extractor: Number of levels in the scale pyramid
ORBextractor.nLevels: 8

# ORB Extractor: Fast threshold
# Image is divided in a grid. At each cell FAST are extracted imposing a minimum response.
# Firstly we impose iniThFAST. If no corners are detected we impose a lower value minThFAST
# You can lower these values if your images have low contrast
ORBextractor.iniThFAST: 20
ORBextractor.minThFAST: 7

#--------------------------------------------------------------------------------------------
# Viewer Parameters
#--------------------------------------------------------------------------------------------
Viewer.KeyFrameSize: 0.05
Viewer.KeyFrameLineWidth: 1
Viewer.GraphLineWidth: 0.9
Viewer.PointSize:2
Viewer.CameraSize: 0.08
Viewer.CameraLineWidth: 3
Viewer.ViewpointX: 0
Viewer.ViewpointY: -0.7
Viewer.ViewpointZ: -3.5 # -1.8
Viewer.ViewpointF: 500
```

Bibliography

[1] URL: https://docs.novatel.com/Waypoint/Content/Home.htm.

[2] Amani Ben Afia. "Development of GNSS/INS/SLAM Algorithms for Navigation in Constrained Environments". In: (2017).

[3] Antonio Angrisano et al. "GNSS/INS integration methods". In: *Dottorato di ricerca (PhD) in Scienze Geodetiche e Topografiche Thesis, Universita'degli Studi di Napoli PARTHENOPE, Naple* 21 (2010).

[4] B Bah et al. "Odometry for train location". In: *European Consortium for Mathematics in Industry (ECMI), Tech. Rep.* (2009).

[5] Omar Al-Bayari. "Mobile mapping systems in civil engineering projects (case studies)". In: *Applied Geomatics* 11.1 (2019). ISSN: 1866928X. DOI: 10.1007/s12518-018-0222-6.

[6] Michael Bloesch et al. "Robust Visual Inertial Odometry Using a Direct EKF-Based Approach". In: (2015), pp. 298–304.

[7] Cesar Cadena et al. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332. ISSN: 15523098. DOI: 10.1109/TRO.2016.2624754. arXiv: 1606.05830.

[8] Michael Calonder et al. "BRIEF: Binary robust independent elementary features". In: *Springer* 6314 LNCS.PART 4 (2010), pp. 778–792. ISSN: 16113349. DOI: 10.1007/978-3-642-15561-1_56.

[9] *Camera calibration and 3D reconstruction*. URL: https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html.

[10] Carlos Campos et al. "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM". In: (2020), pp. 1–15. arXiv: 2007.11898. URL: http://arxiv.org/abs/2007.11898.

[11] Carlos Campos et al. "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM". In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890.

[12] Chang Chen et al. "A review of visual-inertial simultaneous localization and mapping from filtering-based and optimization-based perspectives". In: *Robotics* 7.3 (2018), pp. 1–20. ISSN: 22186581. DOI: 10.3390/robotics7030045.

[13] Lee E. Clement et al. "The Battle for Filter Supremacy: A Comparative Study of the Multi-State Constraint Kalman Filter and the Sliding Window Filter". In: *Proceedings -2015 12th Conference on Computer and Robot Vision, CRV 2015* (2015), pp. 23–30. DOI: 10.1109/CRV.2015.11.

[14] Andrew J Davison. "Real-Time Simultaneous Localisation and Mapping with a Single Camera 1 Introduction 2 Repeatable Localisation". In: *IEEE* (2003), p. 1403.

[15] dorian3d. *DORIAN3D/DBOW2: Enhanced hierarchical bag-of-word library for C++*. URL: https://github.com/dorian3d/DBoW2.

[16] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct Sparse Odometry". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.3 (2018), pp. 611–625. ISSN: 01628828. DOI: 10.1109/TPAMI.2017.2658577. arXiv: 1607.02565.

[17] Jakob Engel, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-Scale Direct monocular SLAM". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8690 LNCS.PART 2 (2014), pp. 834–849. ISSN: 16113349. DOI: 10.1007/978-3-319-10605-2_54.

[18] Zheng Fang and Yu Zhang. "Experimental evaluation of RGB-D visual odometry methods". In: *International Journal of Advanced Robotic Systems* 12 (2015), pp. 1–16. ISSN: 17298814. DOI: 10.5772/59991.

[19] Georgios Giannakaras. "MSc THESIS Monocular Visual Inertial Odometry for Underwater Vehicle Navigation , Optimized on Embedded System". In: (2019).

[20] Mohinder S Grewal, Lawrence R Weill, and Angus P Andrews. *Global positioning systems, inertial navigation, and integration*. John Wiley & Sons, 2007.

[21] C. Harris and M. Stephens. "A Combined Corner and Edge Detector". In: *Procedings of the Alvey Vision Conference 1988*. Vol. 15. Citeseer, 1988, pp. 23.1–23.6.

[22] Richard Hartley and René Vidal. "The multibody trifocal tensor: Motion segmentation from 3 perspective views". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 1 (2004). ISSN: 10636919. DOI: 10.1109/cvpr.2004.1315109.

[23] Jan Hartmann, Jan Helge Klussendorff, and Erik Maehle. "A comparison of feature descriptors for visual SLAM". In: *2013 European Conference on Mobile Robots, ECMR 2013 - Conference Proceedings* (Sept. 2013), pp. 56–61. DOI: 10.1109/ECMR.2013.6698820. URL: http://ieeexplore.ieee.org/document/6698820/.

[24] Carsten Hasberg, Stefan Hensel, and Christoph Stiller. "Simultaneous localization and mapping for path-constrained motion". In: *IEEE Transactions on Intelligent Transportation Systems* 13.2 (2011), pp. 541–552.

[25] Peter Henry, Evan Herbst, and Dieter Fox. "RGB-D Mapping : Using Kinect-Style Depth Cameras for Dense 3D Modeling of RGB-D Mapping : Using Depth Cameras for Dense 3D Modeling of Indoor Environments". In: *The International Journal of Robotics Research* 31.May 2016 (2012), pp. 647–663. DOI: 10.1177/0278364911434148.

[26] Wolfgang Hess et al. "Real-time loop closure in 2D LIDAR SLAM". In: *Proceedings - IEEE International Conference on Robotics and Automation* 2016-June (2016), pp. 1271–1278. ISSN: 10504729. DOI: 10.1109/ICRA.2016.7487258.

[27] Antti Hietanen et al. "A comparison of feature detectors and descriptors for object class matching". In: *Neurocomputing* 184 (2016), pp. 3–12. ISSN: 18728286. DOI: 10.1016/j.neucom.2015.08.106. URL: http://dx.doi.org/10.1016/j.neucom.2015.08.106.

[28] Berthold Horn, Hugh Hilden, and Shahriar Negahdaripour. "Closed-Form Solution of Absolute Orientation using Orthonormal Matrices". In: *Journal of the Optical Society of America A* 5 (July 1988), pp. 1127–1135. DOI: 10.1364/JOSAA.5.001127.

[29] Niklas Karlsson et al. "The vSLAM Algorithm for Robust Localization and Mapping". In: (2005).

[30] Christian Kerl and Daniel Cremers. “Dense Visual SLAM for RGB-D Cameras”. In: *IEEE* (2013), pp. 2100–2106.

[31] Alif Ridzuan Khairuddin, Mohamad Shukor Talib, and Habibollah Haron. “Review on Simultaneous Localization and Mapping (SLAM)”. In: November (2015), pp. 27–29.

[32] Sunhyo Kim and Se-young Oh. “SLAM in Indoor Environments using Omni-directional Vertical and Horizontal Line Features”. In: *Journal of Intelligent and Robotic Systems* (2008), pp. 31–43. DOI: 10.1007/s10846-007-9179-0.

[33] Nicola Krombach, David Droeschel, and Sven Behnke. “Combining Feature-Based and Direct Methods for Semi-dense Real-Time Stereo Visual Odometry”. In: *Intelligent Autonomous Systems 14*. Ed. by Weidong Chen et al. Cham: Springer International Publishing, 2017, pp. 855–868. ISBN: 978-3-319-48036-7.

[34] Rainer Kümmerle et al. “G2o: A general framework for graph optimization”. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2011), pp. 3607–3613. ISSN: 10504729. DOI: 10.1109/ICRA.2011.5979949.

[35] Stefan Leutenegger et al. “Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization”. In: (2016). DOI: 10.15607/rss.2013.ix.037.

[36] Chu-Tak Li and Wan-Chi Siu. “Fast Monocular Vision-based Railway Localization for Situations with Varying Speeds”. In: *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE. 2018, pp. 2006–2013.

[37] David G Lowe. “Object recognition from local scale-invariant features”. In: *IEEE* 2 (1999), pp. 1150–1157. ISSN: 00721077. DOI: 10.1130/2011.2482(04).

[38] *Main page*. URL: https://eigen.tuxfamily.org/index.php?title=Main_Page.

[39] Michael J Milford and Gordon F Wyeth. “SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights”. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 1643–1649.

[40] Anastasios I. Mourikis and Stergios I. Roumeliotis. “A multi-state constraint Kalman filter for vision-aided inertial navigation”. In: *Proceedings - IEEE International Conference on Robotics and Automation* April (2007), pp. 3565–3572. ISSN: 10504729. DOI: 10.1109/ROBOT.2007.364024.

[41] Shafiq Muhammad. “GNSS/INS Integration in Urban Areas”. PhD thesis. 2014.

[42] Raul Mur-Artal, J. M.M. Montiel, and Juan D. Tardos. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163. ISSN: 15523098. DOI: 10.1109/TRO.2015.2463671. arXiv: 1502.00956.

[43] Raul Mur-Artal and Juan D. Tardos. “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras”. In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262. ISSN: 15523098. DOI: 10.1109/TRO.2017.2705103. arXiv: 1610.06475.

[44] Raul Mur-Artal and Juan D. Tardos. “Visual-Inertial Monocular SLAM with Map Reuse”. In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 796–803. ISSN: 23773766. DOI: 10.1109/LRA.2017.2653359. arXiv: 1610.05949.

[45] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. “DTAM: Dense tracking and mapping in real-time”. In: *Proceedings of the IEEE International Conference on Computer Vision* (2011), pp. 2320–2327. DOI: 10.1109/ICCV.2011.6126513.

[46] D. Nister. “An efficient solution to the five-point relative pose problem”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.6 (June 2004), pp. 756–770. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2004.17. URL: http://ieeexplore.ieee.org/document/1288525/.

[47] OpenCV. *OpenCV*. Oct. 2022. URL: https://opencv.org/.

[48] *ORB-SLAM3 Calibration Tutorial*. Available at https://github.com/UZ-SLAMLab/ORB_SLAM3/blob/master/Calibration_Tutorial.pdf. 2021.

[49] Cheng Pan et al. "A Robust Adaptive Cubature Kalman Filter Based on SVD for Dual-Antenna GNSS/MIMU Tightly Coupled Integration". In: *Remote Sensing* 13.10 (May 2021), p. 1943. ISSN: 2072-4292. DOI: 10.3390/rs13101943. URL: http://dx.doi.org/10.3390/rs13101943.

[50] Dan Pritsker. "Hybrid implementation of Extended Kalman Filter on an FPGA". In: *2015 IEEE Radar Conference (RadarCon)*. 2015, pp. 0077–0082. DOI: 10.1109/RADAR.2015.7130974.

[51] I. Puente et al. "Review of mobile mapping and surveying technologies". In: *Measurement: Journal of the International Measurement Confederation* 46.7 (2013), pp. 2127–2145. ISSN: 02632241. DOI: 10.1016/j.measurement.2013.03.006. URL: http://dx.doi.org/10.1016/j.measurement.2013.03.006.

[52] Tong Qin, Peiliang Li, and Shaojie Shen. "Vins-mono: A robust and versatile monocular visual-inertial state estimator". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

[53] Meixiang Quan et al. "Accurate Monocular Visual-Inertial SLAM Using a Map-Assisted EKF Approach". In: *IEEE Access* 7 (2019), pp. 34289–34300. ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2904512. arXiv: 1706.03648.

[54] RainerKuemmerle. *Rainerkuemmerle/G2O: G2o: A general framework for graph optimization*. URL: https://github.com/RainerKuemmerle/g2o.

[55] Alejandro Rituerto and Luis Puig. "Visual SLAM with an omnidirectional camera". In: (2010), pp. 0–3. DOI: 10.1109/ICPR.2010.94.

[56] Edward Rosten and Tom Drummond. "Erickson - 2008 - Methods of treatment using anti-erbb antibody-maytansinoid conjugates.pdf". In: (2006), pp. 430–443.

[57] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *Proceedings of the IEEE International Conference on Computer Vision* (2011), pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[58] Marta Salas et al. "Trajectory alignment and evaluation in SLAM: Horn's method vs alignment on the manifold". In: 2015.

[59] Davide Scaramuzza. "Visual Odometry and SLAM: past, present, and the robust-perception age". In: ETH, 2016. URL: https://www.rsj.or.jp/databox/international/iros16tutorial_2.pdf.

[60] Davide Scaramuzza and Friedrich Fraundorfer. "Visual Odometry [Tutorial]". In: December (2011).

[61] Thomas Schneider et al. "maplab: An open framework for research in visual-inertial mapping and localization". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1418–1425.

[62] Mubarak Shah. "Fundamentals of Computer Visions". In: *Orlando: University of Central Florida* (1997).

[63] N El-Sheimy. "An overview of mobile mapping systems". In: *FIG Working Week* April (2005), pp. 1–24. URL: http://scholar.google.com/scholar?hl=en%7B%5C&%7DbtnG=Search%7B%5C&%7Dq=intitle:An+Overview+of+Mobile+Mapping+Systems%7B%5C#%7D0%7B%5C%7D5Cnhttp://scholar.google.com/scholar?hl=en%7B%5C&%7DbtnG=Search%7B%5C&%7Dq=intitle:An+overview+of+mobile+mapping+systems%7B%5C#%7D0.

[64] Tomasi Shi, Jianbo and Carlo. "Good Features to Track". In: *IEEE* 178.December (1993), pp. 593–600.

[65] Geraldo Silveira et al. "An Efficient Direct Approach to Visual SLAM". In: 24.5 (2008), pp. 969–979.

[66] Cyrill Stachniss, John J. Leonard, and Sebastian Thrun. "Simultaneous Localization and Mapping". In: *Springer Handbook of Robotics*. Cham: Springer International Publishing, 2016, pp. 1153–1176. ISBN: 9783319325521. DOI: 10.1007/978-3-319-32552-1_46. URL: http://link.springer.com/10.1007/978-3-319-32552-1%7B%5C_%7D46.

[67] Stevenlovegrove. *Stevenlovegrove/pangolin: Pangolin is a lightweight portable rapid development library for managing opengl display / interaction and abstracting video input.* URL: https://github.com/stevenlovegrove/Pangolin.

[68] Jürgen Sturm et al. "A benchmark for the evaluation of RGB-D SLAM systems". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 573–580. DOI: 10.1109/IROS.2012.6385773.

[69] Taek Lyul Song. "Observability of target tracking with bearings-only measurements". In: *IEEE Transactions on Aerospace and Electronic Systems* 32.4 (Oct. 1996), pp. 1468–1472. ISSN: 1557-9603. DOI: 10.1109/7.543868.

[70] Taek Lyul Song. "Observability of target tracking with range-only measurements". In: *IEEE Journal of Oceanic Engineering* 24.3 (July 1999), pp. 383–387. ISSN: 1558-1691. DOI: 10.1109/48.775299.

[71] Sebastian Thrun. "Probabilistic robotics". In: *Communications of the ACM* 45.3 (2002), pp. 52–57. ISSN: 00010782. DOI: 10.1145/504729.504754.

[72] Pekka Toivanen, Vandad Imani, and Keijo Haataja. "Three main paradigms of simultaneous localization and mapping (SLAM) problem". In: April 2018 (2018), p. 74. ISSN: 1996756X. DOI: 10.1117/12.2310094.

[73] Xin Tong et al. "A double-step unscented Kalman filter and HMM-based zero-velocity update for pedestrian dead reckoning using MEMS sensors". In: *IEEE Transactions on Industrial Electronics* 67.1 (2019), pp. 581–591.

[74] Florian Tschopp et al. "Experimental Comparison of Visual-Aided Odometry Methods for Rail Vehicles". In: *IEEE Robotics and Automation Letters* PP (Feb. 2019), pp. 1–1. DOI: 10.1109/LRA.2019.2897169.

[75] Florian Tschopp et al. "Experimental comparison of visual-aided odometry methods for rail vehicles". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1815–1822. ISSN: 23773766. DOI: 10.1109/LRA.2019.2897169. arXiv: 1904.00936.

[76] Shinji Umeyama. "Least-squares estimation of transformation parameters between two point patterns". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 13.04 (1991), pp. 376–380.

[77] Lukas Von Stumberg, Vladyslav Usenko, and Daniel Cremers. "Direct sparse visual-inertial odometry using dynamic marginalization". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 2510–2517.

[78] Johan Wahlström and Isaac Skog. "Fifteen years of progress at zero velocity: A review". In: *IEEE Sensors Journal* 21.2 (2020), pp. 1139–1151.

[79] H Wang, J Berkers, and Fugro NL Land BV. "ABSOLUTE AND RELATIVE TRACK GEOMETRY: CLOSING THE GAP". In: ().

[80] Jürgen Wohlfeil. "Vision based rail track and switch recognition for self-localization of trains in a rail network". In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2011, pp. 1025–1030.

[81] Rinara Woo, Eun-Ju Yang, and Dae-Wha Seo. "A Fuzzy-Innovation-Based Adaptive Kalman Filter for Enhanced Vehicle Positioning in Dense Urban Environments". In: *Sensors* 19 (Mar. 2019), p. 1142. DOI: 10.3390/s19051142.

[82] Li Xu, Jiaya Jia, and Yasuyuki Matsushita. "Motion detail preserving optical flow estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.9 (2011), pp. 1744–1757.

[83] Nan Yang, Rui Wang, and Daniel Cremers. "Feature-based or Direct : An Evaluation of Monocular Visual Odometry Feature-based or Direct : An Evaluation of Monocular Visual Odometry arXiv : 1705 . 04300v1 [cs . CV] 11 May 2017". In: June (2017). arXiv: arXiv:1705.04300v1.

[84] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. "An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics". In: *Intelligent Industrial Systems* 1.4 (2015), pp. 289–311. ISSN: 2363-6912. DOI: 10.1007/s40903-015-0032-7.

[85] Feng Zhang et al. "Algorithms analysis of mobile robot SLAM based on Kalman and particle filter". In: *2017 9th International Conference on Modelling, Identification and Control (ICMIC)*. IEEE. 2017, pp. 1050–1055.

[86] Zichao Zhang, Guillermo Gallego, and Davide Scaramuzza. "On the comparison of gauge freedom handling in optimization-based visual-inertial state estimation". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2710–2717.

[87] Zichao Zhang and Davide Scaramuzza. "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 7244–7251.